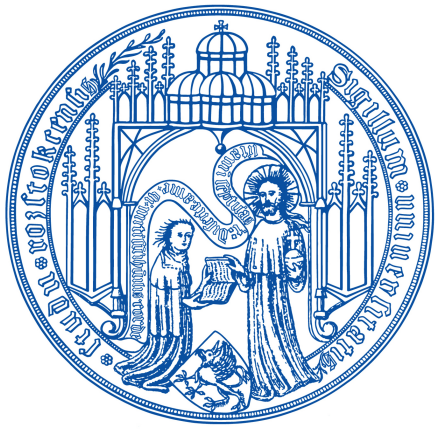
Vertrauenswürdige, adaptive Anfrageverarbeitung in dynamischen Sensornetzwerken zur Unterstützung assistiver Systeme

Dissertation zur Erlangung des akademischen Grades Doktor-Ingenieur (Dr.-Ing.)

Universität Rostock
Fakultät für Informatik und Elektrotechnik
Institut für Informatik



vorgelegt von: M.Sc. Hannes Grunert
geboren am: 30.08.1987 in Ribnitz-Damgarten
Gutachter: Prof. Dr. rer. nat. habil. Andreas Heuer, Universität Rostock
Prof. Dr. rer .nat. habil. Clemens H. Cap, Universität Rostock
Prof. Dr. rer. nat. habil. Sven Groppe, Universität zu Lübeck
Datum der Abgabe: 02.03.2022
Datum der Verteidigung: 15.09.2022

https://doi.org/10.18453/rosdok_id00004022

„Der Teleschirm war Sende- und Empfangsgerät zugleich. Jedes von Winston verursachte Geräusch, das über ein gedämpftes Flüstern hinausging, würde registriert werden; außerdem konnte er, solange er in dem von der Metallplatte kontrollierten Sichtfeld blieb, ebenso gut gesehen wie gehört werden. Man konnte natürlich nie wissen, ob man im Augenblick gerade beobachtet wurde oder nicht. Wie oft oder nach welchem System sich die Gedankenpolizei in jede Privatleitung einschaltete, darüber ließ sich bloß spekulieren. Es war sogar denkbar, dass sie ständig alle beobachtete. Sie konnte sich jedenfalls jederzeit in jede Leitung einschalten. Man musste folglich in der Annahme leben – und tat dies auch aus Gewohnheit, die einem zum Instinkt wurde –, dass jedes Geräusch, das man verursachte, gehört und, außer bei Dunkelheit, jede Bewegung beäugt wurde.“

George Orwell: „1984“ [Orw49]

„To be always 'on' would destroy the human organism.“

Alan Westin: „Privacy and Freedom“ [WR67]

Abstrakt

Assistenzsysteme in smarten Umgebungen sammeln durch den Einsatz verschiedenster Sensoren viele Daten, um die Intentionen und zukünftigen Aktivitäten der Nutzer zu berechnen. In den meisten Fällen werden dabei mehr Informationen gesammelt als für die Erfüllung der Aufgabe des Assistenzsystems notwendig sind.

Das Ziel dieser Dissertation ist die Konzeption und Implementierung von datenschutzfördernden Algorithmen für die Weitergabe sensibler Sensor- und Kontextinformationen zu den Analysewerkzeugen der Assistenzsysteme. Die Datenschutzansprüche der Nutzer werden dazu in Integritätsbedingungen der Datenbanksysteme transformiert, welche die gesammelten Informationen speichern und auswerten.

Ausgehend vom Informationsbedarf des Assistenzsystems und den Datenschutzbedürfnissen des Nutzers werden die gesammelten Daten so nahe wie möglich am Sensor durch Selektion, Reduktion, Kompression oder Aggregation durch die Datenschutzkomponente des Assistenzsystems verdichtet. Sofern nicht alle Informationen lokal verarbeitet werden können, werden Teile der Analyse an andere, an der Verarbeitung der Daten beteiligte Rechenknoten ausgelagert.

Das Konzept wurde im Rahmen des PArADISE-Frameworks (Privacy-AwaRe Assistive Distributed Information System Environment) umgesetzt und u. a. in Zusammenarbeit mit dem DFG-Graduiertenkolleg 1424 (MuSAMA – Multimodal Smart Appliances for Mobile Application) anhand eines Beispielszenarios getestet.

Abstract

Assistive systems in smart environments collect a lot of data by using a wide variety of sensors, in order to predict the intentions and future activities of their users. In most cases, more information is collected than necessary for the assistance system to complete its purpose.

The aim of this dissertation is the conception and implementation of privacy enhancing algorithms for the transfer of sensitive sensor and context information to the analysis tools of the assistive systems. For this purpose, the privacy requirements of the users are transformed into integrity constraints of the database systems that store and analyze the collected information.

Based on the information demands of the assistive system and the data privacy needs of the user, the collected data is condensed as close as possible to the sensor by selection, reduction, compression or aggregation by the data privacy component of the assistance system. If not all information can be processed locally, parts of the analysis are outsourced to other computing nodes involved in processing the data.

The concept was implemented within the framework of the PArADISE framework (Privacy-AwaRe Assistive Distributed Information System Environment) and tested, among other projects, in cooperation with the DFG Research Training Group 1424 (MuSAMA – Multimodal Smart Appliances for Mobile Application).

Einordnung in das 2012er ACM Computing Classification System

Das *ACM Computing Classification System*[1] gilt in der Informatik als de-facto-Standard zur Klassifikation von wissenschaftlichen Beiträgen. Die vorliegende Dissertationsschrift umfasst die folgenden Konzepte:

- Information systems
  - Data management systems
    - Database management system engines
      - Integrity checking
      - Data exchange
- Security and privacy
  - Database and storage security
    - Data anonymization and sanitization
    - Information accountability and usage control
  - Human and societal aspects of security and privacy
    - Privacy protections
- Human-centered computing
  - Ubiquitous and mobile computing
    - Ubiquitous and mobile computing theory, concepts and paradigms
      - Ubiquitous computing
      - Mobile computing
      - Ambient intelligence
  - Social and professional topics
    - Computing / technology policy
      - Privacy policies

Schlüsselwörter

Datenschutz, Datensparsamkeit, Ubiquitous Computing, Query-Containment-Problem, Anfragetransformation

Keywords

privacy, privacy-awareness, ubiquitous computing, query containment problem, query transformation

[1] `https://www.acm.org/publications/class-2012`, zuletzt aufgerufen am 12.08.2021.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Im Informationszeitalter wird es für den Menschen immer wichtig, stetig vernetzt zu sein. Mobiltelefone erinnern uns an dringende Termine, die wir am Tag zuvor an unserem Arbeitsplatzrechner eingegeben haben. Wenn wir hungrig sind, weisen sie uns per GPS den kürzesten Weg zum nächsten Currywurststand. Im Jahr 2001 wurde durch den Film *Startup* die Vision einer weltweiten, multimedialen und mobilen Kommunikation skizziert. Wie wir heute, nur wenige Jahre später, sehen, blieb es nicht nur bei einer Vision. Das Informationszeitalter wirkt sich jedoch nicht nur auf unsere Kommunikation, sondern auch auf viele weitere Bereiche unseres Lebens aus. Die vom Assistenzsystem erfassten Daten werden im System gespeichert und mit weiteren Daten, beispielsweise mit dem Facebook-Profil des Nutzers, verknüpft. Durch die so gewonnenen Informationen lassen sich unsere Vorlieben und Verhaltensmuster, aber auch zukünftige Ereignisse berechnen. Die Einsatzmöglichkeiten von Informationssystemen scheinen dabei unbegrenzt.

Assistenzsysteme im Einsatz

Die Automobilindustrie verbaut in modernen Autos zunehmend mehr eingebettete Systeme, inklusive Sensorik [Eic18]. Abstandswarner verhindern Unfälle, indem sie per Kameras oder Schall ihre Umgebung abtasten; Spurhalteassistenten sorgen auf der Autobahn für ein entspannteres Fahren. Einige Firmen, wie Tesla[1], Google[2] oder Apple[3] arbeiten, neben traditionellen Automobilherstellern wie VW[4] oder BMW[5], an Technologien für selbstfahrende Automobile. Dabei wird jede Fahrentscheidung auf Basis von großen Datenmengen, die im Intervall von wenigen Millisekunden durch Hunderte von Sensoren erfasst werden, getroffen.

Auch unser Alltag wird zunehmend durch Assistenzsysteme bestimmt. In unserem Smart Home [Wei91] erfassen Sensoren jeden Augenblick unseres Lebens, erkennen was wir tun und berechnen die wahrscheinlichste Folgehandlung. Ausgehend von diesen Berechnungen stellt sich das Assistenzsystem auf unsere Gewohnheiten ein und steuert Heizung, Beleuchtung und die Musikanlage. Auf Arbeit sorgen Smart Meeting Rooms [HK05] für reibungslose Abläufe bei Besprechungen. Präsentationen werden automatisch geladen, sobald ein Redner an das Rednerpult tritt; Leinwände und Verdunkelung fahren automatisiert herunter.

Ein Großteil dieser smarten, ubiquitären Umgebungen basieren auf Cloud-Architekturen, bei denen unsere Geräte lediglich als Client dienen. Die eigentliche Berechnung, die mitunter aus komplexen Funktionen besteht und auf großen Datenmengen arbeitet, wird von Cloud-Diensten, wie Amazons Web Services[6] oder IBMs Plattform

[1] `https://www.tesla.com/autopilot`, zuletzt aufgerufen am 05.01.2022

[2] `https://www.google.com/selfdrivingcar/`, zuletzt aufgerufen am 05.01.2022

[3] `https://www.macwelt.de/a/apple-auto-icar-geruechte-ausstattung-releasedatum-preis,3292706`, zuletzt aufgerufen am 05.01.2022

[4] `https://www.volkswagen.de/de/elektrofahrzeuge/elektromobilitaet-erleben/elektroauto-technologie/autonomes-fahren-level-und-gesetzliche-regelungen.html`, zuletzt aufgerufen am 05.01.2022

[5] `https://www.bmwgroup.com/en/innovation/technologies-and-mobility/autonomes-fahren.html`, zuletzt aufgerufen am 05.01.2022

[6] `https://aws.amazon.com`, zuletzt aufgerufen am 05.01.2022

IBM Cloud[7] (ehemals *Bluemix*), übernommen. Hinter diesen Diensten verbergen sich große Rechenzentren, die durch die Verwendung von Clustersystemen eine massiv-parallele und effiziente Datenverarbeitung in Echtzeit erreichen. Viele Endgeräte sind aufgrund ihrer beschränkten Leistungsfähigkeit dazu nicht in der Lage, wodurch eine Auslagerung der Berechnungen in die Cloud unabdingbar ist.

Assistenzsysteme und Datenschutz

Dieses Outsourcing ist aber mit einem Problem verbunden: Durch die Herausgabe unserer persönlichen Informationen geben wir die Datenhoheit an den Dienstleister bzw. den Cloudanbieter ab. Dabei kann nicht sichergestellt werden, dass die Grundsätze des Datenschutzes eingehalten werden, obwohl Vertraulichkeit und Anonymität zu den wichtigsten Sicherheitsbedürfnissen bei der Informationsverarbeitung gehören. Nach einer Studie [TNS13, BN14] von TNS Infratest im Auftrag des IT-Planungsrates sehen befragte IKT-Experten aus Wissenschaft, Wirtschaft und Verwaltung die Thematik *IT-Sicherheit und Datenschutz* zu 74% als das dringendste Thema im Bereich der Digitalisierung an. Damit liegt es noch vor dem *Breitbandausbau* (63%) und der *Digitalen Souveränität* (im Sinne von Medienkompetenz; 46%).

In Assistenzsystemen kommt es häufig vor, dass mehr Informationen gesammelt als benötigt werden. Außerdem hat der Nutzer meist nur einen sehr geringen Einfluss auf die Speicherung und Verarbeitung seiner personenbezogenen Daten, auch wenn innerhalb der letzten Jahre viele Datenschutzrichtlinien in Wirtschaft und Politik zugunsten der Bürger abgeändert wurden [Deu18, Eur16]. Meist ist jedoch das Recht auf informationelle Selbstbestimmung weiterhin verletzt, da vertrauliche Informationen teilweise ungefragt an einen unbefugten Anwenderkreis weitergegeben werden. Nach einer Studie der Europäischen Kommission [Eur11] fühlen sich 69% der Internetnutzer bei der Nutzung von kostenlosen Suchmaschinen und kostenlosen E-Mail-Anbietern unwohl, da personenbezogene Daten zur Anzeige von relevanter Werbung genutzt werden. Zudem geben nur 18% der befragten Personen an, dass sie das Gefühl haben, die vollständige Kontrolle über ihre Daten in sozialen Netzwerken zu besitzen.

Fast täglich werden wir von Meldungen überschwemmt, dass Datenschutzverletzungen in unterschiedlichen Lebensbereichen stattfinden. Dazu zählen nicht nur die Enthüllungen von Edward Snowden[8], sondern auch alltägliche Dinge, die uns aufzeigen, dass sich in den letzten Jahren nicht viel verändert hat. Viele Cloud-Anbieter verweisen in ihren Datenschutzrichtlinien darauf, dass Abkommen wie Safe Harbour [Mar06] bzw. dessen Nachfolger Privacy Shield [WA16] nicht unbedingt eingehalten werden. Häufig werden die Daten zudem für die zielgruppengerichtete Schaltung von Werbung oder für andere, nicht näher spezifizierte Aufgaben verwendet.

Datenschutz wird zudem als Hemmschwelle bzw. Fortschrittsbremse für die Einführung von innovativen Softwareprodukten und als Hindernis für die Wissensverbreitung gesehen, obwohl Datenschutz als nichttechnischer Faktor in vielen Anforderungen an die Entwicklung von Software steckt [DDH$^+$15]. So scheiterte der Elektronische Entgeltnachweis (ELENA) im Jahr 2011 nach neun Jahren Entwicklung. Das von der deutschen Bundesregierung in Auftrag gegebene Vorhaben sollte für Bürokratieabbau und Einsparungen für Arbeitnehmer, Arbeitgebern und bei den abrufenden Stellen sorgen, wurde aber nach mehrmaligen Verschiebungen wegen Datenschutzbedenken verschoben [Sch11]. Wären die Bedenken und Lösungsvorschläge bezügliches des Datenschutzes [Sch10] in die Entwicklungs- und Konzeptionsphase von ELENA mit eingeflossen, so hätte eine datenschutzkonforme Lösung in der Projektlaufzeit gefunden werden können.

Datenschutz ohne Informationsverlust

Im Bundesdatenschutzgesetz [Deu18], den untergeordneten Landesdatenschutzgesetzen (z. B. das Landesdatenschutzgesetz für Mecklenburg-Vorpommern [Lan02]) und vielen weiteren Gesetzestexten sind zwei Hauptforderungen des Datenschutzes die Aspekte der Datensparsamkeit und Datenvermeidung. Informationssysteme sollen möglichst wenig Daten erheben und verarbeiten, sowie die Daten, sofern mit geringem Aufwand verbunden, geeignet anonymisieren bzw. pseudonymisieren. Dies stellt jedoch einen Widerspruch zu den Grundsätzen des Cloud Computing dar: Die Systeme brauchen für die komplexen Berechnungen alle Rohdaten, um ein möglichst genaues Ergebnis zurückliefern zu können. Durch die Anonymisierung verfälschte bzw. unvollständige Daten führen

[7] `https://cloud.ibm.com`, zuletzt aufgerufen am 05.01.2022

[8] Ein guter Überblick ist unter `https://www.heise.de/extras/timeline-2013` zu finden (zuletzt aufgerufen am 05.01.2022).

zu abweichenden Ergebnissen und somit zu veränderten Handlungen des Assistenzsystemes. Stellen Sie sich ein selbstfahrendes Auto vor, welches nur die ungefähre Position eines Fußgängers, der unvorhergesehen die Straße kreuzt, kennt. Kein schöner Gedanke, wenn der Wagen versucht auszuweichen, um einen Unfall zu vermeiden, aber in Wirklichkeit die eigentliche Position des Fußgängers ansteuert.

Die Datenverarbeitung ist durch die bestehende Gesetzeslage [Eur16, Deu15] abgedeckt, solange die Verarbeitung zweckgebunden und im Einverständnis mit dem Betroffenen erfolgt. Eine Anonymisierung der Daten ist notwendig, wenn lediglich vorher definierte Analysen auf den personenbezogenen Informationen ausgeführt werden. Um zu verhindern, dass weitere Verarbeitungen auf den gespeicherten Daten ausgeführt werden, muss eine Datenminimierung erfolgen. Werden nur solche Informationen aufbewahrt, die zur gewollten Analyse benötigt werden, sind die Möglichkeiten zur Ausführung anderer Analysemethoden beschränkt.

Im Rahmen dieser Dissertation wird eine neuartige Form der Datenverarbeitung vorgestellt, welche den scheinbaren Widerspruch zwischen *Datensparsamkeit* und *Big Data* auflöst. Dazu werden die aus der Datenbanktheorie bekannten *Techniken zur Anfrageoptimierung* neu aufbereitet und mit dem Konzept des *Edge Computing* verbunden. Der daraus resultierende Ansatz *Query Rewriting by Contract* bezieht alle bei der Datenerzeugung und -verarbeitung beteiligten Hardwarekomponenten, angefangen bei Sensoren, über mobile Geräte bis hin zu Rechnern in großen Datenzentren, in die Anfrageverarbeitung mit ein.

Edge Computing und dessen Automatisierung

In dem eingangs erwähnten Film *Startup* stehen die Entwickler des Kommunikationssystems vor einem Problem: Die an der Übertragung der Videodaten beteiligten Geräte verfügen nicht über genug Rechenleistung, um die Komprimierung der Videodaten für eine stabile Übertragung in Echtzeit durchzuführen. Nach einer langen Nacht kommt ihnen die zündende Idee: Die Antwort für ihr Problem liegt nicht in der Verarbeitung auf den Endgeräten, sondern in der Ausführung im *Netz* selbst. Dabei sollen die Geräte, die an der Kommunikation beteiligt sind, zu der Verarbeitung der Daten beitragen. Zwar ist aus dieser Fiktion bereits in Form des Edge Computings [SD16] Wirklichkeit geworden, jedoch werden Edge-Anwendungen gegenwärtig „per Hand" auf die einzelnen Geräte in einer bekannten Systemumgebung verteilt. Wünschenswert ist ein Verfahren, um beliebige Analysen automatisiert in einer dynamischen Umgebung mit den unterschiedlichsten, lose gekoppelten Geräten zu verteilen. Dadurch kann besser auf die Dynamik von Big Data- und Data Science-Anwendungen reagiert werden.

Durch die ständig, nach Moore's Law [Moo65], steigende Rechenleistung bzw. mit den in der Post-Moore-Ära [Mei03] aufkommenden Technologien, sind heutige Großrechner problemlos in der Lage, tausende von Anfragen und Datenströmen gleichzeitig zu bearbeiten. Heute dienen viele Geräte, wie Router, Repeater und Netzwerkknoten, lediglich zum Transport der Daten zwischen Endgeräten und der Cloud. Wird eines dieser Geräte nicht vollständig ausgelastet, wird viel potenzielle Rechenleistung verschwendet.

Die vorliegende Dissertation zeigt auf, wie diese Rechenleistung genutzt werden kann, um den durch Sensoren generierten Datenstrom schrittweise vorzuverarbeiten. Dabei werden die Charakteristika der verschiedenen Geräteklassen optimal ausgenutzt: Sensoren führen einfache Selektionsbedingungen aus, während lokale Server bereits einen Teil der komplexen Anfragen vorberechnen und somit ein hochverdichtetes Ergebnis an die Cloud weiterleiten. Aus diesen Teilberechnungen lässt sich in der Cloud die restliche Anfrage vollständig und, sofern keine zusätzliche Anonymisierung der Zwischenergebnisse erfolgte, ohne Informationsverlust ausführen.

Ein angepasster Anfrageprozessor, welcher sowohl die Anfrage als auch das vorläufige Zwischenergebnis modifiziert, fördert die Einhaltung der Datenschutzbedürfnisse des Nutzers. Durch eine datensparsame Weitergabe der Sensor- und Kontextinformationen an Analysewerkzeuge des Assistenzsystems wird nicht nur der Datenschutz des Systems verbessert, sondern auch dessen Effizienz durch die parallele Datenverarbeitung gesteigert [MH15a]. Durch eine verringerte Bearbeitungszeit auf den Zwischenknoten und der geringeren Datenemission von Sensoren bzw. Knoten wird zudem der Energieverbrauch verringert [SD16].

1.1 Konkrete Problemstellung

Das Übertragen sämtlicher Informationen, die in einem Sensornetzwerk gesammelt werden, ist in vielerlei Hinsicht problematisch: Es kann nicht sichergestellt werden, dass es sich bei den von den Sensoren gesammelten Informationen ausschließlich um die eigenen, personenbezogenen Daten handelt. So können beispielsweise Bewegungsmelder und Überwachungskameras auch die Aktivitäten von Personen in der näheren Umgebung aufnehmen.

Zudem hat die Übermittlung aller Daten zur Folge, dass der Dienstanbieter, welcher die Daten erhält, zusätzliche Auswertungen, seien diese gewollt oder ungewollt, ausführen kann. Dadurch wird der, in der Datenschutz-Grundverordnung [Eur16] geforderte, Grundsatz der *Zweckbindung* verletzt. Zusätzlich könnten Dritte, wie beispielsweise kriminelle Organisationen oder staatliche Überwachungsorgane, die Möglichkeit haben, Daten während der Übertragung mitzuschneiden und für ihre eigenen Zwecke zu missbrauchen. Liegen Daten hingegen bereits vorselektiert und voraggregiert vor, ist das Risiko des Datenmissbrauchs reduziert. Die Einhaltung der *Integrität und Vertraulichkeit* stellt einen weiteren Grundsatz der Datenschutz-Grundverordnung dar.

Problematisch wird es, wenn die Daten zu stark anonymisiert vorliegen. In diesem Fall kann es vorkommen, dass der eigentliche Dienst des Anbieters nicht funktioniert bzw. stark verfälsche Ergebnisse zurückliefert. Um diesem Problem vorzubeugen, müssen Techniken entwickelt werden, die nur die minimale Menge an notwendigen Daten übertragen. Außerdem müssen die Daten so anonymisiert werden, dass der Informationsverlust für die gestellte Anfrage möglichst gering ist, während der Informationsverlust für alle anderen Auswertung möglichst hoch ist.

Hinsichtlich des Funktionsumfanges unterscheiden sich die beteiligten Rechenknoten extrem. Dadurch ist es erforderlich, dass die gerätespezifischen Eigenschaften mit untersucht werden, da nicht jeder Rechenknoten in der Lage ist, Anfragen vollständig zu verarbeiten. Je nach Verfügbarkeit müssen Teile von komplexen Selektionen, Projektionen und Aggregationen abgespalten und getrennt voneinander ausgeführt werden, um die sensornahe, datensparsame Verarbeitung der Daten zu ermöglichen. Zudem muss die Anfrage muss zudem semantisch äquivalent zu der Originalanfrage sein, da ansonsten die verwendeten Analysemodelle zu Auswertung großer Datenmengen ggf. falsche oder zu ungenaue Ergebnisse liefern. Die in der Datenschutz-Grundverordnung [Eur16] geforderte *Datenminimierung* wird durch dieses vorgehen bei der Erhebung, Verarbeitung und Nutzung der Daten auf allen beteiligten Rechenknoten eingehalten.

Im Themenfeld *Privacy by Design* besteht nach der Europäischen Agentur für Netz- und Informationssicherheit (European Union Agency For Network And Information Security, ENISA [DDK+15][9]) ein großer Bedarf zur Erforschung von dezentralen und datenschutzgerechten Analysemethoden. Nach [PS17] und [Deu17c] wurde insbesondere dem Aspekt der Datensparsamkeit bisher nur wenig Beachtung geschenkt. Eine Studie [TTG14] hat aufgezeigt, dass die meisten smarten Systeme, die zwar datenschutzfördernde Techniken integriert haben, meist die Aspekte der Zweckbindung und Datensparsamkeit außen vor gelassen haben. Dies zeigt, dass insbesondere diese Aspekte einer erhöhte Aufmerksamkeit bzgl. der Entwicklung und Erforschung neuer Lösungsansätze erfordern. Ziel dieser Arbeit ist es, diese Lücke zu schließen.

1.1.1 Fragestellungen und Zielsetzungen der Arbeit

Bereits in der Grundschule wird erklärt, wie sich im Brandfall verhalten werden soll. Bei der Meldung des Brandes sollen dabei folgende fünf Fragen beantwortet bzw. eingehalten werden:

1. Wer meldet?
2. Was ist passiert?
3. Wie viele sind betroffen oder verletzt?
4. Wo ist es passiert?
5. Warten auf Rückfragen.

[9] Die Agentur wurde am 28. Juni 2019 in *Agentur der Europäischen Union für Cybersicherheit* umbenannt.

Ähnlich zur Feuerwehr, die für die Verhinderung bzw. das Löschen von Bränden zuständig ist, kann der Datenschutz als Beschützer des Menschen vor der missbräuchlichen Nutzung seiner personenbezogenen Daten angesehen werden. Davon ausgehend ist die Zielsetzung dieser Arbeit von drei bzw. fünf Fragestellungen motiviert:

Fragestellung 1: *Wie lässt sich Datenschutz in Informationssystemen, insbesondere in sensorgestützten Assistenzsystemen, realisieren?*

Diese Frage wird primär in Kapitel 3 dieser Arbeit beantwortet. Neben einen allgemeinen Lösungskonzept werden zwei Ansätze zur Lösung skizziert.

Unter personenbezogenen Daten werden meist Informationen zum Alter und Geschlecht einer Person gezählt. In sensorgestützten Systemen werden hingegen Positionsdaten, die anliegende Stromspannung oder einfache An/Aus-Werte erfasst. Auf den ersten Blick ist nicht klar, welche Daten geschützt werden sollen bzw. die Identifikation einer Person ermöglichen. Daraus ergibt sich die zweite Forschungsfrage:

Fragestellung 2: *Durch welche Daten lassen sich in Assistenzsystemen einzelne Personen bzw. deren Handlungen identifizieren? Wie lassen sich diese Daten bestimmen und so verändern, dass der Datenschutz gewahrt wird?*

Kapitel 4 geht näher auf die Bestimmung dieser sogenannten Quasi-Identifikatoren ein. Das Kapitel endet dabei mit einem negativen Ergebnis: In realen Anwendungsszenarien existieren zu viele Attributkombinationen, sodass eine ausreichende Anonymisierung mit einem nur unverhältnismäßig hohen Informationsverlust möglich ist. Diese Erkenntnis führt zur dritten und letzten[10] zentralen Fragestellung:

Fragestellung 3: *Wie lässt sich durch gezielte Vorverarbeitung der Daten und mit geringem Informationsverlust der Personenbezug aus den Daten entfernen, sodass eine Anonymisierung nicht mehr oder nur selten nötig ist?*

In Kapitel 5 wird zunächst auf bestehende Techniken aus der Grundlagenforschung im Bereich Datenbanken und angrenzenden Disziplinen der Informatik eingegangen. Dabei wird überprüft, wie diese Techniken zur Lösung der obigen Fragestellungen eingesetzt werden können. Aufbauend darauf wird in Kapitel 6 ein Konzept erarbeitet, welches sich dieser Problemstellung annimmt und löst.

Im Rahmen dieser Dissertation werden datensparsame Verfahren zur adaptiven Anfrageverarbeitung in Sensornetzwerken weiterentwickelt. Dabei wird insbesondere die Dynamik der beteiligten Sensoren und der technischen Umgebung für die Adaptivität der Abfragebearbeitung untersucht.

Die vorliegende Arbeit bietet eine Lösung für die obige Problemstellung, indem der erforderliche Datenschutz nicht vorrangig durch eine, mit Informationsverlust behaftete, Anonymisierung der Daten erfolgt. Es werden vielmehr Lösungsstrategien aufgezeigt, wie durch eine gezielte Anfragetransformation und -verteilung ein hohes Datenschutzniveau erreicht werden kann. Die folgenden Zielsetzungen wurden dafür definiert:

- Erarbeitung eines Konzeptes für die datenschutzkonforme Anfrageverarbeitung in Assistenzsystemen, welches die Aspekte der Datensparsamkeit und Datenvermeidung umsetzt.
- Spezifikation und Erkennung von schützenswerten Daten, die als Ausgangsbasis für die Verteilung der Anfragen dienen.
- Erarbeitung einer Strategie, bei der Daten durch schrittweise Vorberechnungen bereits lokal und ohne Informationsverlust anonymisiert werden.

Das Hauptziel ist ein integriertes Gesamtkonzept für eine datenschutzfreundliche Anfrageverarbeitung, beginnend bei der eingehenden Anfrage bis hin zur Anonymisierung des Analyseergebnisses. Dabei soll ein Ausgleich zwischen den Forderungen nach Datenschutz seitens der Nutzer und dem Informationsbedarf des Informationssystems gefunden werden.

[10] Als weitere Fragestellungen verbleiben die Fragen *Wer kümmert sich um die Einhaltung des Datenschutzes? – Das Informationssystem* und *Wann wird der Datenschutz eingehalten? – Jederzeit.*

1.1.2 Schwerpunkte der Arbeit

Ausgehend von der Problemstellung wurden drei Schwerpunkte erarbeitet, welche in der vorliegenden Arbeit näher betrachtet werden. Dazu gehören:

- *Allgemeines Lösungskonzept* (Kapitel 3 – Konzept)
  - Abänderung der Anfrageverarbeitung in Datenbanksystemen
  - Vorverarbeitung und Modifikation von Anfragen
  - Anonymisierung von Analyseergebnissen
- *Erkennung von schützenswerten Daten* (Kapitel 4 – Erkennung von Quasi-Identifikatoren)
  - Charakteristik von schützenswerten Daten
  - Effiziente Berechnung von Quasi-Identifikatoren in hochdimensionalen Datensätzen
  - Nutzen von Quasi-Identifikatoren als Parameter zur Anfragetransformation und Ergebnisanonymisierung
- *Vertikale Anfrageverteilung* (Kapitel 6 – Verteilung von Anfragen)
  - Theoretischer Hintergrund
  - Verteilungsstrategien für ressourcenbeschränkte Umgebungen
  - Verteilte Berechnung von komplexen Aggregatfunktionen

1.1.3 Abgrenzung

Ausgehend von den Zielsetzungen wurden die Schwerpunkte der Arbeit abgeleitet. Datenschutz und datenschutzfördernde Techniken umfassen jedoch viele weitere Aspekte, die im Rahmen dieser Arbeit nicht abgedeckt werden. Ausgehend von den acht *Datenschutz Design Pattern* nach Koops et. al [KHL13] (Details siehe Abschnitt 2.4) werden insbesondere die Aspekte des *Informierens* und des *Demonstrierens* im vorgestellten Konzept nicht näher betrachtet.

Zudem wird die Verschlüsselung der Daten, sowohl in Datenbanksystemen als auch beim Transport, außer Acht gelassen. Zwar stellt die Vermeidung bzw. Verhinderung des Ausspähens, Abhörens und Abfangens von Daten, sowie deren Löschung und Vervielfältigung ohne Kenntnisnahme (siehe §201 und §202 StGB [Deu17a] sowie §100 Telekommunikationsgesetz [Deu17b]) eine wichtige Forderung an Informationssysteme dar, fällt jedoch vielmehr unter den Aspekt der Datensicherheit bzw. des allgemeinen Strafrechtes. Dies betrifft ebenso die unbefugte Datenveränderung durch Unbrauchbarmachung oder inhaltlicher Umgestaltung von Daten (§303a StGB [Deu17a]).

Auch die Aspekte der Onion Encryption [GKK04], speziell bei der Aggregation von Sensordaten [TH20], und der Secure Multi-Party Computation (SMC) [Yao86] werden nicht näher betrachtet. Für beide Aspekte existieren bereits weitreichende Konzepte für die Anwendung in Kombination mit relationalen Anfragesprachen [PRZB11, Sep13, SCD14] und Aggregatfunktionen [BE15] bzw. im Speziellen auch für Systeme wie das Smart Grid [LCC14]. Zudem existieren Konzepte, mit denen sich viele Operatoren der relationalen Algebra, wie Selektion, Verbund und Aggregation, auch auf verschlüsselten Daten ausführen lassen [ABE+13]. Die genannten Techniken lassen sich zusätzlich zu dem von mir in Kapitel 6 eingeführten Konzept anwenden und ergänzen einander – sie sind somit nicht als direkte Konkurrenz anzusehen.

Datensicherheit und dessen Umsetzung durch Verschlüsselung, Passwortschutz und Zugriffskontrollsysteme werden in dieser Arbeit vielmehr als Voraussetzung für die Gewährleistung von Datenschutz angesehen (vgl. IT-Grundschutz-Kataloge [Mün16, Baustein 1.5: Übergreifende Aspekte – Datenschutz]). Dies betrifft neben der Speicherung der Daten auch den Transport der Daten zwischen verschiedenen Computersystemen bzw. anderen Formen der elektronischen Datenübermittlung, beispielsweise per Telefon oder Fax.

Zudem wird in dieser Arbeit von einem *honest-but-curious*-Angreifer ausgegangen, d. h. einem Nutzer, der legitimen Zugang zu dem Datenbestand hat, jedoch ein hohes Missbrauchspotenzial aufweist, da er die Daten auch

für andere Zwecke als vorgesehen, sei es absichtlich oder unabsichtlich, nutzen könnte. In Anlehnung an die in [AAA+21] skizzierte Privacy-Leakage-Attacke betrifft dies im Internet der Dinge insbesondere die Nutzung von Datenknoten, die nicht unter der Kontrolle der Nutzer stehen.

Andere Arten von Angreifern, welche es gezielt auf das Ausspähen, die Manipulation oder die Zerstörung der Daten aus politischen, sozialen oder monetären Gründen abgesehen haben, werden in dieser Arbeit nicht betrachtet. Insbesondere werden auch keine Angriffe durch SQL-Injection, Folgen von Anfragen [DDS79] oder verwandte Verfahren untersucht.

1.2 Publikationen und Präsentationen

Einige der Schlüsselkonzepte dieser Dissertation wurden im Vorfeld auf nationalen und internationalen Konferenzen bzw. Workshops sowie in Fachzeitschriften vorgestellt:

- Der Ansatz zur Überprüfung von Datenschutzmetriken für Folgen von Anfragen wurde 2014 im Rahmen des Doktorandenseminares auf der *INFORMATIK* vorgestellt [Gru14].
- Der Algorithmus zur effizienten Berechnung von Quasi-Identifikatoren wurde auf dem Workshop *Grundlagen von Datenbanken* im Jahr 2014 vorgestellt [GH14].
- Ebenfalls auf dem Workshop *Grundlagen von Datenbanken* wurde 2015 die Umsetzung des Slicing-Ansatzes zur Anonymisierung von Daten auf Datenbankebene gezeigt [GH15c].
- Auf der *8th International Conference on PErvasive Technologies Related to Assistive Environments* wurde 2015 die erste Version der Auszeichnungssprache PP4SE für die Formulierung von Datenschutzansprüchen in smarten Umgebungen präsentiert [GH15a].
- Im Rahmen der *19th International Conference on Extending Database Technology* wurde 2016 der Ansatz zur vertikalen Anfrageverteilung erstmals vorgestellt [GH16b].
- In der Zeitschrift *Datenbank-Spektrum* wurde im Jahr 2016 das Gesamtkonzepts des zu entwickelten Anfrageprozessors und dessen einzelner Bestandteile vorgestellt [GH16a].
- Auf der Konferenz *Lernen, Wissen, Daten, Analysen* wurde 2016 die Anwendbarkeit der entwickelten Konzepte im Internet der Dinge gezeigt [GKB+16].
- Im *Open Journal of Internet of Things (OJIOT)* wurden die theoretischen Grundlagen für die Anfrageverteilung und deren Anwendung auf komplexen Aggregatfunktionen präsentiert [GH17]. Der Beitrag wurde zudem 2017 auf dem Workshop *Very Large Internet of Things (VLIoT)* im Rahmen der Konferenz *Very Large DataBases (VLDB)*[11] vorgestellt.
- Die Verfeinerung des zuletzt genannten Konzeptes wurde in der Folgepublikation [GH18] ebenfalls im *OJIOT* und auf dem *VLIoT*-Workshop der *VLDB* 2018 vorgestellt.
- Als Ausblick wurde auf dem *VLIoT*-Workshop der *VLDB* 2020 [HG20] die Anwendung des Konzepts, bei dem das *gesamte* Internet der Dinge als hochgradig verteilte Datenbankmaschine genutzt wird, skizziert.

1.3 Gliederung & Roter Faden durch die Arbeit

Der Kern dieser Arbeit wurde in den Kapiteln 3, 6 und 7 niedergeschrieben. Nachdem in diesem Abschnitt die Frage aufgeworfen wurde, wie Datenschutz in assistiven Systemen umgesetzt werden kann, gibt Kapitel 3 eine Übersicht über das dafür entwickelte Konzept zur datenschutzkonformen Anfrageverarbeitung. Dabei wird

[11] Die VLDB zählt, neben der *International World Wide Web Conference*, der *IEEE Transactions on Knowledge and Data Engineering* und der *ACM SIGMOD International Conference on Management of Data* zu den wichtigsten Konferenzen im Bereich der Datenbankforschung [Goo21].

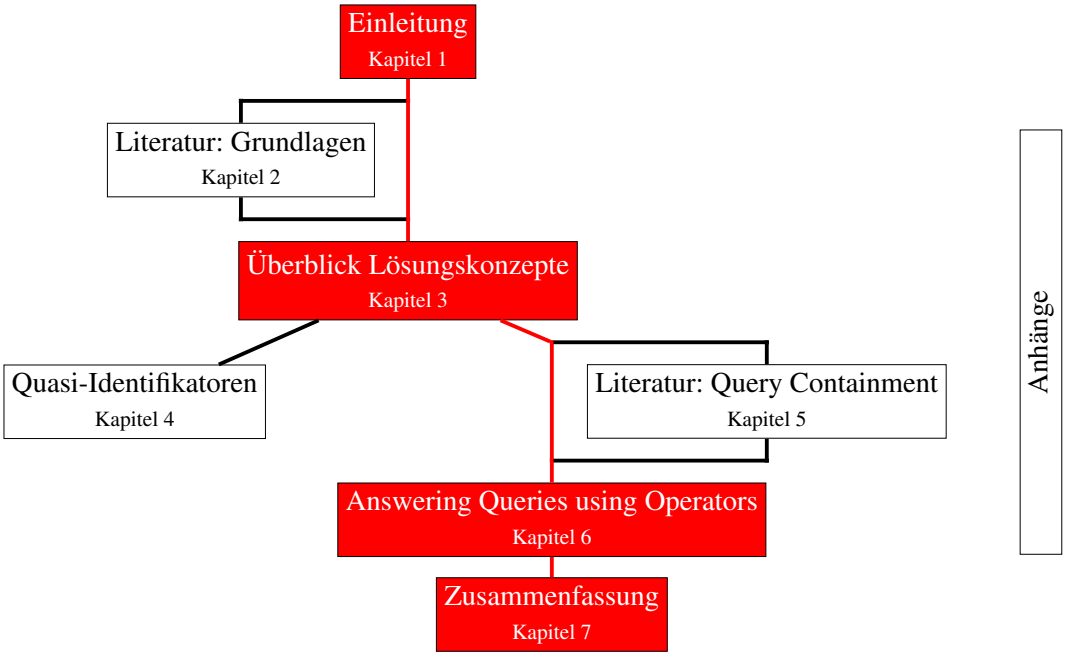


Abbildung 1.1: Der rote Faden durch die Kapitel dieser Arbeit.

gezeigt, dass bei der Anfrageverarbeitung im Wesentlichen an zwei Zeitpunkten eingegriffen werden kann: ***VOR*** und ***NACH*** dem Ausführen einer Anfrage auf dem Datenbanksystem.

Die darauf folgenden Kapitel 4 und Kapitel 6 konzentrieren sich auf zwei Teilaspekte des Gesamtkonzeptes: Kapitel 4 stellt einen effizienten Ansatz zur Erkennung von Quasi-Identifikatoren in hochdimensionalen Daten vor. Die ermittelten Ergebnisse dienen zur Parametrisierung vieler Komponenten des Anfrageprozessors, u. a. für die Anonymisierung der Daten ***NACH*** dem Ausführen einer Anfrage. Kapitel 4 endet dabei mit einem negativen Ergebnis: In hochdimensionalen Daten, wie sie beispielsweise in assistiven Systemen vorkommen, existieren zu viele Möglichkeiten zur Identifizierung von Personen und deren Aktivitäten, sodass eine ausreichende Anonymisierung der Daten nur mit einem sehr hohen Informationsverlust möglich ist – was meist zur Lasten der Funktionsfähigkeit des Systems führen würde.

In Kapitel 6 wird mit *Answering Queries using Operators* eine Technik vorgestellt, die mittels vertikaler Anfrageverteilung einen Eingriff ***VOR*** dem Ausführen der Anfrage vornimmt. Das Verfahren wahrt den Datenschutz durch gezielte, lokale Vorverarbeitungen der gesammelten Rohdaten eines Assistenzsystems, ohne dabei Daten anonymisieren zu müssen. Dieses Kapitel stellt zugleich den Kern der Arbeit dar. Die Ausarbeitung schließt mit einer Zusammenfassung in Kapitel 7 ab.

Definitionen, Schreibweisen und Modelle bezüglich Datenbanken, Datenschutz und smarter Umgebungen werden in Kapitel 2 eingeführt. Dieses Grundwissen dient als Ausgangslage für die folgenden Kapitel. In den einzelnen Kapiteln wird gesondert auf spezielle Definitionen und den Stand der Technik verwiesen. Das Kapitel 5 gibt aufgrund seines Umfangs einen gesonderten Überblick zu Techniken im Forschungsbereich des Query Containments und den damit verwandten Ansätzen. Zusätzliche Informationen zu den Kapiteln 2 und 3 werden in den Anhängen A bzw. B beschrieben.

Technische Hinweise

Im Rahmen dieser Arbeit werden Definitionen, Zitate und Beispiele durch speziell hervorgehobene Umgebungen gekennzeichnet:

Zitat: Lorem ipsum

Zitate sind blau markiert. Die Quelle des Zitats wird in der Überschrift der Box als Link zum Literaturverzeichnis angegeben.

Definition: Lorem ipsum

Definitionsboxen sind orange markiert. Sie führen neu definierte Begriffe ein bzw. übernehmen diese aus der Literatur. Erfolgt die Definition des Begriffs durch Formeln, so wird dies i. d. R. durch die Relationenalgebra oder das relationale Modell umgesetzt.

Beispiel: Lorem ipsum

Beispielboxen sind grün markiert. Sie enthalten u. a. SQL-Anfragen, Ausschnitte aus Relationen und Rechenwege.

Im Fließtext werden ausgewählte Textstellen vom normalen Text hervorgehoben. Neue bzw. wichtige Begriffe werden durch *Kursivschrift* betont, `Bezeichner` hingegen in einer nicht-proportionalen Schriftart.

Quellcodeausschnitte werden, sofern nicht anders angeführt, als PostgreSQL-Dialekt oder in Java angegeben. Gegebenenfalls kann der Quellcode zur Steigerung der Lesbarkeit gekürzt abgedruckt sein. Der vollständige Quellcode und die notwendigen externen Bibliotheken sind dem Begleitmaterial (siehe Anhang G) entnehmbar.

Kapitel 2

Grundlagen

Ziel dieses Kapitels ist es, die für diese Arbeit notwendigen Grundlagen in komprimierter Form aufbereitet wiederzugeben. Ausgangspunkt bilden Assistenzsysteme bzw. Big Data-Anwendungen und die dabei möglichen Verletzungen der Privatsphäre in Abschnitt 2.1. Mit den Themen Datenbanken und Cloud Computing werden in den Abschnitten 2.2 bzw. 2.3 die beiden wichtigsten Technologien dieser Arbeit eingeführt. Abschließend werden in den Abschnitten 2.4 bzw. 2.5 die rechtlichen und technischen Grundlagen des Datenschutzes angesprochen.

2.1 Assistenzsysteme und smarte Umgebungen

Technologie durchdringt immer mehr Bereiche unseres Lebens: In den Fabrikhallen vieler Autohersteller sorgt Industrie 4.0 für die Herstellung von individuellen, nach Kundenwünschen angepassten Automobilen, noch während diese am Fließband zusammengesetzt werden [WR17]. Auf der Straße verwenden diese Autos verschiedene Fahrerassistenzsysteme, um Routen zu planen, die Spur zu halten und Verkehrsunfälle zu verhindern. Während der Fahrt kommuniziert das Fahrzeug ohne weiteres Zutun mit unserem vernetzten Zuhause: Die Heizung wird so geregelt, dass bei unserer Ankunft (inkl. Einberechnung von Verkehrsbehinderungen) die Wohnung unserem persönlichem Empfinden nach temperiert ist [OW08].

Eine Studie des Bundesverbandes der Energie- und Wasserwirtschaft [Bun16] hat in einer repräsentativen Umfrage ermittelt, welche Smart-Home-Anwendungen von der Bevölkerung am nützlichsten angesehenen werde. Mit 92% werden medizinische Assistenzsysteme deutlich häufiger genannt als verschiedene Arten von fernsteuerbaren Geräten (46% bis 68%). Mit lediglich 21% Zustimmung ist die Automatisierung von Haushaltsgeräten weit weniger erwünscht.

Die Fernsehwerbung (siehe Abbildung 2.1) suggeriert ein anderes Bild von *smart*: Die Fernsteuerung der Heizung via Mobiltelefon oder der Einsatz von Bewegungsmeldern gilt noch nicht als smart – die Intelligenz der Geräte liegt hier beim Menschen, da er die Steuerung übernimmt bzw. ist im Fall des Bewegungsmelders nicht vorhanden, da er als reaktives System nur die Bewegung erkennt, nicht jedoch die eigentliche Anwesenheit einer Person. Die Intelligenz kommt vielmehr erst dann zustande, wenn die Geräte eigenständig Informationen austauschen und auf die Bedürfnisse der Nutzer reagieren. Die erwähnte Heizungssteuerung würde als intelligent gelten, wenn sie eigenständig erkennt, dass der Nutzer auf dem Weg nach Hause ist und die (automatisch erkannte) gewünschte Temperatur einstellt.

In [Wei91] skizziert Mark Weiser die Vision der allgegenwärtigen elektronischen Intelligenz, die unsichtbar und unaufdringlich Unterstützung im Alltag bieten soll:

> **Zitat:** Mark Weiser zu Ubiquitous Computing [Wei91]
>
> Ubiquitous Computing enhances the computer use by making computers available throughout the physical environment, while making them effectively invisible to the user. [...] The most profound technologies are

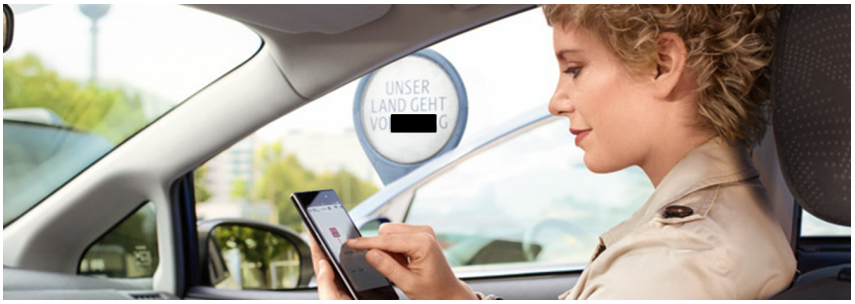


Abbildung 2.1: Ausschnitt aus einer Werbeanzeige eines in Deutschland ansässigen Energieunternehmens (Name geschwärzt). Das Unternehmen wirbt mit der Vernetzung des smarten Zuhauses von unterwegs. Die eigentliche Intelligenz liegt jedoch nicht beim System, sondern beim Benutzer, welcher die Fernsteuerung der smarten Umgebung übernimmt[1].

> those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it.

Aus dieser Forderung ergeben sich neue Fragestellungen und Probleme: Wie wird mit unsichtbaren Rechnern interagiert bzw. wie wird die Funktionalität genutzt? Wie werden diese Funktionen realisiert, wenn mehrere hunderte bzw. tausende (oder gar Milliarden?) Recheneinheiten an der Informationsverarbeitung beteiligt sind? Welche neuen, nicht-funktionalen Anforderungen, besonders hinsichtlich des Datenschutzes, entstehen daraus?

Insbesondere die ethischen Fragestellungen bedürfen einer genaueren Untersuchung. Eine Studie [MWRF13] identifizierte ethische Probleme beim Einsatz von Assistenzsystemen: Einerseits schränkt die Erhebung und Verarbeitung der Daten die informationelle Selbstbestimmung stark ein. Andererseits ermöglichen Assistenzsysteme in der eigenen Wohnung ein selbstbestimmteres Leben, insbesondere bei kognitiv beeinträchtigten Patienten, die ansonsten in einem Pflegeheim untergebracht werden müssten. Basierend auf der Studie wurden hinsichtlich der Privatheit und des Datenschutzes drei Leitlinien erarbeitet:

> **Zitat:** Ethische Leitlinien für den Einsatz von altersgerechten Assistenzsystemen [MWRF13]
>
> - **Leitlinie 6 – Privatheit:** Altersgerechte Assistenzsysteme sollen die persönliche Lebensgestaltung nicht negativ beeinträchtigen.
> - **Leitlinie 7 – Datenschutz:** Personenbezogene und sonstige vertraulich zu behandelnden Daten, die im Kontext von altersgerechten Assistenzsystemen erhoben, dokumentiert, ausgewertet oder gespeichert werden, sollen vor dem Zugriff unbefugter Dritter sowie vor Missbrauch bestmöglich geschützt werden.
> - **Leitlinie 8 – Aufklärung & informationelle Selbstbestimmung:** Nutzer von altersgerechten, technischen Assistenzsystemen sollen vollständig über die Funktion und Erhebung der sie betreffenden Daten und die Funktion des Systems informiert werden und erst auf dieser Basis eine informierte Einwilligung geben.

Neuere Publikationen [BSS19, PSHSG21] zeigen ähnliche ethische Aspekte und Herausforderungen auf, insbesondere in Hinblick auf den Punkt *Aufklärung*. Der folgende Unterabschnitt geht näher auf die technischen Details und den Aufbau eines Assistenzsystems ein. Anschließend werden einige Umsetzungen für diese Art von Systemen vorgestellt. Zum Schluss werden ausgehend von den ethischen Leitlinien [MWRF13] tiefergreifende Anforderungen an den technischen Datenschutz für Assistenzsysteme diskutiert.

[1] Quelle: RWE Smarthome, `http://www.rwe-smarthome.de`; letzter Aufruf am 16.10.2017. Die Website ist in dieser Form nicht mehr verfügbar. Eine Kopie der Website ist unter `https://web.archive.org` aufrufbar.

2.1.1 Aufbau eines Assistenzsystemes

Ziel dieses Unterabschnittes ist es, einen Überblick über den Aufbau und die Heterogenität von Assistenzsystemen zu vermitteln. Nach [HKHT06] bestehen Assistenzsysteme aus den folgenden fünf aufeinander aufbauenden Komponenten:

1. Sensorik und Ortung
2. Heterogene Vernetzung und spontane Gerätekooperation
3. Verteilte Datenverwaltung
4. Situations-, Handlungs- und Intentionserkennung
5. Multimodale Interaktion durch Ein- und Ausgaben

Im Folgenden werden die einzelnen Komponenten näher vorgestellt. Dabei wird auch auf aktuelle Entwicklungen, Technologien und Forschungsthemen eingegangen.

Sensorik und Ortung

Sensoren messen kontinuierlich Daten zur Lage, Drehung und Beschleunigung von Personen und Objekten oder erfassen die Daten der Umgebung, wie Luftdruck, Temperatur und die Konzentration von Gasen. Kameras nehmen in unterschiedlichen Detailstufen (Standardaufnahme, 3D-Tiefeninformationen, ...) Bewegtbilder auf und können ggf. Gesten und Mimik der Personen erkennen. Mikrofone nehmen Sprache, aber auch Töne und weitere Geräusche auf. Für die Ortung kommen, je nach Einsatzszenario, Systeme zur Indoor-Ortung, wie z. B. WLAN, oder zur Outdoor-Ortung (GPS, Galileo) zum Einsatz.

Für die Datengrundlage in einem Assistenzsystem werden häufig Daten von Sensoren und weiteren mobilen Geräten, wie Smartphones, Tablets und Notebooks, erhoben. In smarten Umgebungen kommen beispielsweise aktive und passive RFID-Chips [WTRK12, WST13], intelligente Fußböden mit Drucksensoren [SL08] und Kameras zum Einsatz, um Personen und Gegenstände zu orten und um ihre Bewegungsabläufe zu verfolgen. Mikrofone nehmen jedes Geräusch im Raum auf, z. B. um auf Kommando den Raum zu verdunkeln und um Filme abzuspielen. Auch weitere Geräte, die auf den ersten Blick nicht als *smart* angesehen werden, sammeln Daten: Steckdosen erkennen, ob ein Gerät angeschlossen ist und wenn ja, wie viel Strom es verbraucht, Leinwände erkennen ihre aktuelle Position, ebenso die Jalousien zur Verdunkelung des Raumes.

Daten in smarten Umgebungen werden häufig kontinuierlich und im Millisekundenbereich generiert. Ein UbiSense-Tag [RG17] zur Lokalisierung von Personen und Objekten erzeugt mit einer Frequenz von etwa 120 Hertz neue Daten – pro Gerät. In einem Szenario von acht oder mehr Nutzern werden so Tausende von Datensätzen pro Sekunde generiert, die gespeichert und ausgewertet werden müssen. Die erste Verarbeitung solcher Daten erfolgt in der Regel über Stromdatenbankmanagementsysteme, wie TinyDB [MFHH05], welche die Daten in Echtzeit analysieren.

Ergänzt werden intelligente Umgebungen durch eine Vielzahl weiterer Sensoren, die für sich allein genommen nicht als smart gelten, jedoch im Zusammenspiel ein smartes System bilden. Beispielsweise sind Bewegungsmelder allein betrachtet nicht smart, da sie lediglich *Bewegung* registrieren, jedoch nicht die *Anwesenheit* einer Person im Raum – dies wird erst aus einem Netz von Bewegungsmeldern im Zusammenspiel mit einer zentralen Komponente, die alle Bewegungen erfasst und auswertet, realisiert.

Neben der Vielfalt von Sensoren erzeugt bereits ein einzelnes Sensorsystem eine große Menge an Daten, die es zu filtern und zu analysieren gilt. Beispielsweise erzeugt ein Motion-Capturing-Anzug mehr als 700 verschiedene Messwerte, die mit einer Abtastrate von mehr als 120 Hz erzeugt werden [HK10]. Um die Datenmenge gering zu halten, wird in [HK10] der Einsatz von Clustering-Verfahren basierend auf Stromdaten vorgeschlagen.

Ein weiteres Einsatzgebiet von Sensorik ist in der Medizin zu finden. Um Patienten mit neuromuskulären Erkrankungen längerfristig ambulant zu untersuchen, können beispielsweise Sensoriksysteme am Bein des Patienten angebracht werden [FML+15], wodurch sich Bewegungsabläufe und die dabei auftretenden Belastungen erfassen lassen. Die Analyse der Daten kann wahlweise live erfolgen, um z. B. auf Stürze des Patienten zu reagieren, oder nachträglich, um Diagnosen aufzustellen.

Heterogene Vernetzung und Gerätekooperation

Sensorik- und Ortungssysteme arbeiten, wie oben am Beispiel erklärt, nicht unabhängig voneinander. Vielmehr erfolgt die Vernetzung mit verschiedenen Ein- und Ausgabegeräten, die bereits im bestehenden System installiert sind. Beispielsweise können Feuchtigkeitssensoren mit dem Verdeck eines Cabrios verbunden sein, damit dieses bei aufkommenden Regen automatisch schließt. Unter Umständen können auch verschiedene Geräte zusammenarbeiten, damit ein gemeinsames Assistenzziel erfüllt werden kann. Bei der Unfallverhinderung erkennen Kameras und Infrarotsensoren Objekte auf der Fahrbahn und betätigen die Bremsen und die Lenkung, damit das Objekt nicht berührt wird. In smarten Assistenzsystemen erfolgt zudem eine spontane Kooperation der Geräte. Fällt ein Gerät durch einen technischen Defekt aus bzw. kommt ein neues mobiles Gerät hinzu, so passt sich das Assistenzsystem an die neue Situation an und verteilt die Aufgaben gemäß den Eigenschaften der verfügbaren Geräte [MBK14].

Die Vernetzung der Geräte erfolgt in viele Fällen über eine Middleware, wie z. B. die in [BN12] vorgestellte und auf Java basierende Helferlein-Middleware. Middlewares eignen sich zur Abstraktion von der eigentlichen Hardware, die ansonsten über verschiedenste Protokolle angesprochen werden müsste. Durch eine vereinheitlichte Sicht auf alle Geräte und Daten wird durch den Einsatz einer Middleware die Programmierung und Steuerung einer smarten Umgebung vereinfacht.

Verteilte Datenverwaltung

Die von den Sensoren erzeugten Daten werden zwar direkt erfasst und analysiert, um unverzüglich auf die aktuellen Aktivitäten zu reagieren. Zusätzlich ist es auch notwendig, die Daten zu verdichten und langfristig zu speichern. Die Ursache liegt im maschinellen Lernen der Algorithmen zur Situations- und Intentionserkennung: Viele Data Mining-Verfahren zur Klassifikation und Kategorisierung, aber auch das Hidden-Markov-Modell [KYHK13], nutzen bestehende Daten, um neue Erkenntnisse zu gewinnen.

Die erhobenen Daten können dabei im System horizontal verteilt gespeichert vorliegen: Sensoren speichern, aufgrund ihres begrenzten Speichers, nur wenige bis gar keine Daten. Der in der Verarbeitungskette nächstverfügbare Rechner, meist ein Kleinstgerät wie beispielsweise ein Raspberry Pi, ein Smartphone oder ein Home Media Center, übernimmt die dauerhafte Speicherung der Rohdaten. Ein zu diesem Thema passender Überblick zu verschiedenen Speicherverwaltungen in gebetteten Systemen ist in [RLAS07] zu finden.

Werden Daten mehrerer Quellen zusammengeführt, so geschieht dies in der Regel cloudbasiert auf einem Cluster mehrerer hochperformanter Rechner. Zusätzlich zu den systeminternen Daten des Assistenzsystems können auch externe Daten für die verschiedenen Algorithmen genutzt werden, indem andere Systeme Daten über Schnittstellen bereitstellen. Zur Verarbeitung der dabei anfallenden großen Datenmengen kommen parallele und verteilte Datenbanksysteme, wie Postgres-XL[2], zum Einsatz. Dies können, abhängig vom konkreten Anwendungsfall, beispielsweise elektronische Patientenakten, Wetterdaten des Deutschen Wetterdienstes oder Verkehrsinformationen zu Baustellen sein.

Situations-, Handlungs- und Intentionserkennung

Unter dem Begriff *Situation* [Nyo19] werden die aktuellen Umgebungsinformationen, d. h. die Position und der Zustand einzelner Objekte innerhalb der smarten Umgebung, zusammengefasst. Die *Situationserkennung* beschäftigt sich mit der Erfassung dieser Umgebungsinformationen. Dabei werden nicht nur Detailinformationen (z. B. Luftfeuchtigkeit 93 %, Geschwindigkeit 30 km\h, Verdecköffnung 85 %) erfasst, sondern diese auch in abstraktere Zustände (z. B. Regen, langsame Fahrt, Verdeck offen) überführt.

Eine *Handlung* [Nyo19] ist die Aktivität einer Person, welche durch das Assistenzsystem unterstützt wird. Aktivitäten umfassen bewusste Handlungen, wie Armbewegungen und Gestik, Positions- und Lagewechsel sowie komplexere Handlungsabläufe, wie das Zubereiten von Mahlzeiten oder Handlungen, die zum Einparken eines Fahrzeuges führen. Handlungen können auch unbewusst erfolgen: Ein schläfriger Blick kann von Kamerasystemen erfasst werden oder ein nervöses, sich ständig wiederholendes Zucken der Füße wird durch einen smarten Fußboden registriert. Ziel der *Handlungserkennung* ist das Erkennen von komplexen Aktivitäten aus einer Folge

[2] `https://www.postgres-xl.org/`, zuletzt aufgerufen am 05.01.2022.

von einfachen Bewegungen. Dabei werden vielfältige Methoden, wie Hidden Markov-Modelle [KYHK13] oder Bayessche Filter [NKY+15], eingesetzt, um Aktivitäten zielgerichtet innerhalb eines großen Zustandsraumes zu erkennen. Die Handlungserkennung dient, zusammen mit der Situationserkennung, als Ausgangsbasis für die Intentionserkennung.

Die *Intentionserkennung* [Nyo19] wird zur Handlungsvorhersage in smarten Systemen eingesetzt. Ziel ist es, durch das Ensemble von smarten Geräten zu erkennen, was eine oder mehrere Personen, die das System nutzen, in naher Zukunft vorhaben zu erledigen bzw. welche Ziele verfolgt werden. Ausgehend von der aktuellen Situation in der smarten Umgebung und den letzten Handlungen bzw. den bekannten Handlungsabläufen des Nutzers wird durch die Intentionserkennung erkannt, wie das Assistenzsystem reagieren muss. Die Geräte in der Umgebung werden entsprechend automatisiert instrumentalisiert, ohne dass der Nutzer mit diesen interagieren oder seine Aufmerksamkeit auf sie lenken muss.

Dazu können beispielsweise Positions-, Beschleunigungs- und Bewegungsdaten einzelner Personen ausgewertet und mit Metainformationen, beispielsweise den zu der jeweiligen Person gehörenden Terminkalender, verknüpft werden. Fährt beispielsweise eine Person die gewohnte Route von seiner Arbeitsstelle zurück nach Hause (Situations- und Handlungserkennung), so kann die intelligente Heizungsanlage angewiesen werden, die Wohnungstemperatur so zu regeln, dass die gewünschte Temperatur beim Erreichen der Wohnung eingestellt ist. Durch die Methoden der Intentionserkennung wird versucht, aus den Sensordaten und Metainformationen auf die Handlungsziele der Personen zu schließen. Die zu lösende Problematik aus Sicht der Forschung und Entwicklung existiert darin, die *passende* Methodik für den jeweiligen Anwendungsfall zu finden.

Die Methoden zur Handlungs- und Situationserkennung basieren häufig auf einer großen Menge annotierter Trainingsdaten oder nutzen Vorwissen, wie die räumliche Anordnung von Geräten und Aktionsbereichen, Tagesordnungen und im Vorfeld bekannte Aktionsfolgen aus. Aktionsfolgen können beispielsweise Handlungssequenzen sein, die zur Durchführung einer bestimmten Intention notwendig sind. Diese können beispielsweise durch Plansynthese unter Ausnutzung von a-priori-Wissen aus der Anwendungsdomäne automatisiert generiert werden [Yor14]. Durch die Plansynthese wird sichergestellt, dass alle zielführenden Pläne berücksichtigt werden.

Die Algorithmen zur Situations- und Intentionserkennung basieren auf komplexen mathematischen Funktionen, wie (linearen) Regressionsanalysen, Autokorrelationen und Matrix-Matrix-Multiplikationen [MH15b]. Diese Funktionen werden, teilweise auch rekursiv, in einer Folge von mehreren Anweisungen ausgeführt. In vielen Fällen werden diese Arten von Algorithmen mittels statistischer Programmiersprachen wie R bzw. mit statistischen Softwareprogrammen wie *MATLAB*[3] oder *Mathematica*[4] formuliert. Diese Algorithmen lassen sich allerdings auch in deklarativen Programmiersprachen, wie beispielsweise SQL bzw. mit den dahinterstehenden theoretischen Modellen (Relationenalgebra bzw. Aussagenlogik), darstellen [MH17].

Unter *maschinellem Lernen* werden *Veränderungen im System, die adaptiv sind in dem Sinn, dass sie dem System ermöglichen, dieselbe oder eine ähnliche Aufgabe beim nächsten Mal effizienter zu lösen* [Sim13, eigene Übersetzung] erfasst. Ziel von maschinellen Lernverfahren ist es, durch die Nutzbarmachung von Erfahrungen eine Verbesserung der Leistungsfähigkeit des Systems zu erreichen. Dabei wird das bestehende Wissen vergrößert, Ähnlichkeiten zwischen bestehenden Fakten werden erkannt und es erfolgt eine Adaption an die veränderte Umgebung. Durch die Modellierung adaptiven Verhaltens ist es möglich, (teil-)autonome Systeme für intelligente Umgebungen zu realisieren. Dabei werden, ausgehend von Wahrscheinlichkeiten und modellierten Annahmen, durch die Algorithmen selbstständig Abwägungen zur intelligenten Steuerung des Systems getroffen.

Die Data Mining-Verfahren werden häufig nicht als alleiniges Mittel zur Datenanalyse eingesetzt, sondern dienen häufig zur Vorbereitung anderer Algorithmen, welche die eigentlichen Schlüsse aus den Daten ziehen. Zur Intentionserkennung werden häufig auch Ansätze aus der Wahrscheinlichkeitstheorie, wie Bayessche Netze und Filter, verwendet [NK15]. Als Kern wird hier speziell der Satz von Bayes zur Berechnung der bedingten Wahrscheinlichkeit genutzt, um von einer gegebenen Beobachtung B auf die Wahrscheinlichkeit p einer bestimmten Wirkung W_i zu schließen.

Als Beobachtungen werden häufig die zuletzt berechneten bzw. wirklich eingetretenen Wirkungen benutzt, wodurch sich die Intentionen im Voraus über einen längeren Zeitraum langfristig planen lassen, sofern keine unge-

[3] `https://de.mathworks.com/products/matlab.html`, zuletzt aufgerufen am 05.01.2022.
[4] `https://www.wolfram.com/mathematica/`, zuletzt aufgerufen am 05.01.2022.

wöhnlichen Ereignisse eintreten. Die Wirkungen W_i entsprechen den möglichen Intentionen der Nutzer, von denen die wahrscheinlichsten unter der Anwendung weiterer Techniken und aktueller (Sensor-)Daten zur Bestimmung der genauen Intention verwendet bzw. korrigiert werden.

Ausgehend von den erkannten Zielen der Nutzer wird anschließend eine Strategie entwickelt, welche bestimmt, *wie* die Ziele realisiert werden sollen. Die so erzeugten Aktionen werden an das jeweilige zuständige System weitergeleitet und dort ausgeführt.

Multimediale Interaktion

Die *multimediale Interaktion* dient als Schnittstelle zwischen dem Assistenzsystem und dessen Nutzern – in beide Richtungen. Als Eingabemöglichkeiten stehen dem Benutzer viele Möglichkeiten offen: Neben den klassischen Wegen (Maus, Tastatur, ...) stehen Touchscreens (vom Smartphone bis zum interaktiven Tisch) und Spracherkennungssysteme bzw. -assistenten (Siri, Cortana, ...) sowie Gesten-erkennende Geräte [HHK09] zur Verfügung. Ebenso ist auch die Steuerung der smarten Umgebung durch die Beobachtung der Blickrichtung der Nutzer möglich [MSS14]. Ziel von assistiven Systemen ist, dass das System möglichst unentdeckt im Hintergrund arbeitet. Moderne Systeme sind meist in den Möbeln, dem Auto oder dem Gebäude selbst integriert, wodurch sie teilweise kaum erkennbar sind.

Anders sieht es bei den Ausgabemedien aus. Dort versucht das Assistenzsystem, sofern es zwingend eingreifen muss, dem Nutzer auf verschiedenen Wegen Informationen zu vermitteln. Eine Türsprechanlage kann beispielsweise mit dem Fernseher verbunden sein [SBMH09]. Überhört der Bewohner mehrfach die Türklingel während er vor dem Fernseher sitzt, so kann beispielsweise die Kamera der Türsprechanlage das Bild auf den Fernseher legen. Gerät eine Person in eine Gefahrensituation, wie z. B. ein eingeschaltet gelassener Herd, können über Lautsprecher Alarmsignale und Ansagen akustisch wiedergegeben werden. In Notfallsituationen, bei denen die Person nicht mehr reagieren kann, können auch Meldungen an externe Personen herausgegeben werden. Im Bereich der häuslichen Pflege wird dies u. a. bei der Sturzerkennung von Demenzpatienten eingesetzt: Wird ein Sturz erkannt, wird eine Meldung an den zuständigen Betreuer bzw. einen Angehörigen geschickt [RGM+12].

Im folgenden Abschnitt werden Einsatzszenarien für Assistenzsysteme vorgestellt. Es folgen dazu Beispiele aus der Pflege, der Heimautomatisierung und der Automobilindustrie.

2.1.2 Assistenzsysteme im Einsatz

Zu dem bekanntesten Anwendungsgebiet von Assistenzsystemen zählt die Heimautomatisierung. Bevor konkrete Beispiele vorgestellt werden, wenden wir uns zunächst der Definition dieser Systeme zu:

Definition: Ambient Assisted Living[a]

Unter *Ambient Assisted Living* (AAL) werden Konzepte, Produkte und Dienstleistungen verstanden, die neue Technologien und soziales Umfeld miteinander verbinden und verbessern mit dem Ziel, die Lebensqualität für Menschen in allen Lebensabschnitten, vor allem im Alter, zu erhöhen. Übersetzen könnte man AAL am besten mit *Altersgerechte Assistenzsysteme für ein gesundes und unabhängiges Leben*.

[a] Quelle: `http://www.aal-deutschland.de/`, zuletzt aufgerufen am 05.01.2022

Ausgehend von der obigen Definition zählen zu altersgerechten Assistenzsystemen Systeme, welche die Haushalts- und Lebensführung sowie die allgemeine Gesundheit unterstützen. In diesem Abschnitt werden einige umfangreiche Beispiele für den Einsatz von Assistenzsystemen beschrieben. Dabei wird für jedes System eine Motivation, ein Überblick zu den verwendeten Geräten und Algorithmen und ein Fazit zu den Verbesserungen durch das Assistenzsystem gegeben.

Intelligente Besprechungsräume

Intelligente Besprechungsräume sollen den Ablauf von Meetings, Diskussionsrunden und Vorträgen erleichtern. Ein Anwendungsfall ist beispielsweise, dass die Verdunklung des Raumes automatisch erfolgt, sobald jemand

dauerhaft in der Nähe des Präsentationsbereiches steht. Das integrierte Projektorsystem fährt parallel dazu hoch und startet die Präsentation des Vortragenden. All dies geschieht ohne weiteres Eingreifen; es muss im Idealfall keine Rednerreihenfolge im Vorfeld angegeben noch Einstellungen getätigt werden.

Am Graduiertenkolleg MuSAMA[5] wurden Assistenzsysteme am Beispiel eines intelligenten Besprechungsraumes erforscht. Die Forschung gliederte sich in folgende vier Schwerpunkte:

- **Forschungsschwerpunkt 1: Kontexterkennung und -analyse.** Durch die Nutzung verteilter und vernetzter Sensorik wird der aktuelle Kontext in der smarten Umgebung bestimmt [NYK15]. Dies betrifft insbesondere die Position von Nutzern und mobilen Geräten im Raum. Zudem wurden Modelle für die Repräsentation von Kontext und für die Beschreibung von quantitativen Aussagen über das Eintreffen und die Erwünschtheit von Situationen entworfen.
- **Forschungsschwerpunkt 2: Multimodale Interaktion und Visualisierung.** In diesem Forschungsbereich wurden Verfahren zur Darstellung von Informationen in verteilten Infrastrukturen und auf ubiquitären Displays entworfen [ENSS15]. Es wurde zudem untersucht, wie sich adaptive Interaktionen mit diesen Geräten durch explizite Aufgabenmodelle umsetzen lassen.
- **Forschungsschwerpunkt 3: Intentionserkennung und Strategieentwicklung.** Auf der Grundlage von Kontextdaten, Nutzerpräferenzen, expliziter Interaktion und Wissen über die Handlungsabläufe der Nutzer werden Vorhersagen über deren Intention getroffen. Auf Basis aller Intentionen wird eine gemeinsame, koordinierte Strategie zur Umsetzung der Nutzerziele entwickelt und umgesetzt [HK05].
- **Forschungsschwerpunkt 4: Datenhaltung, Ressourcen- und Infrastrukturmanagement.** Für die Realisierung der in Punkt 3 entwickelte Strategie müssen Ensemble-weite Basisdienste bzw. eine Infrastruktur bereitgestellt werden. Im folgenden Abschnitt wird näher auf diesen Forschungsschwerpunkt eingegangen.

Im Forschungsschwerpunkt 4 wurde die Dynamik der Vernetzung von den beteiligten Geräten sowie deren Kommunikationsbeziehungen untersucht (siehe z. B. [Sei17]). Bestehende Ansätze zur komplexen Analyse von Daten in hochvernetzten Umgebungen, wie Publish / Subscribe-Systeme oder Analysen in einer zentralen Cloud, ermöglichen die adaptive Kopplung der einzelnen Knoten. Diese Systeme besitzen jedoch nur eingeschränkte Möglichkeiten, den Datenschutz für der Nutzer zu gewährleisten, da meist alle Nutzerdaten weitergeleitet werden.

Im Rahmen dieser Dissertation wird eine adaptive und vertrauenswürdige Anfrageverarbeitung in verteilten Informationssystemen konzipiert und entwickelt. Dabei ist zu beachten, dass die verwendeten Systeme extrem heterogen sind: In cloudbasierten Systemen werden Datenbank mit vollständigen Funktionsumfang, d. h. mit kompletter Umsetzung des SQL-Standards und erweiterten Analysefunktionen, genutzt, während auf den mobilen Geräten und Sensoren lediglich Kleinstdatenbanken mit sehr beschränkter Funktionalität eingesetzt werden. Dazwischen existieren eine Vielzahl von unterschiedlichen Systemen mit differenzierten Funktionsumfang. Im eigenen entwickelten Ansatz wird untersucht, wie eine verteilte Berechnung der Daten auf diese unterschiedlichen Systeme unter der Ausnutzung ihrer spezifischen Eigenschaft erfolgen kann, sodass nur diejenigen Daten verarbeitet und weitergeleitet werden, welche für die Erfüllung der festgelegten Ziele des unterstützenden Assistenzsystems zwingend notwendig sind.

Weitere Beispiele für Assistenzsysteme werden im Anhang A.1 aufgeführt. Diese umfassen verschiedene Einsatzbereiche wie die häusliche Pflege und Fahrerassistenzsysteme.

2.1.3 Privatsphäre in Assistenzsystemen

Wie den oben genannten, einzelnen Beispielen und der Motivation dieser Dissertationsschrift zu entnehmen, stellt Datenschutz eine wichtige Anforderung an den Entwurf und den Betrieb von Informationssystemen dar. Im Folgenden wird – in knapper Form – der Zusammenhang zwischen Assistenzsystemen und Datenschutzaspekten hergestellt.

[5] Multimodal Smart Appliance Ensembles for Mobile Applications, `http://musama.informatik.uni-rostock.de`, zuletzt aufgerufen am 05.01.2022

Angst vor neuen Technologien

In [BGS05] werden die Ängste seitens der Verbraucher gegenüber der RFID-Technologie, insbesondere in ubiquitären Umgebungen, thematisiert. Zu diesen Ängsten zählen

- der Kontrollverlust über die eigenen Geräte und die damit verbundene unbemerkte Überwachung,
- das Erstellen von Bewegungsprofilen,
- die Entdeckung von Fehlverhalten sowie
- die allgemeine Sammlung von Information die damit verbundene Bildung von Nutzerprofilen.

Zudem werden in dem Artikel mögliche technische Schutzansätze, wie Verschlüsselung, Kill-Funktionen und Abschirmung, genannt, aber auch die Integration von Datenschutzprofile wie die *Platform for Privacy Preferences (P3P)* [W3C07] als möglicher Lösungsansatz vorgestellt.

Motti und Caine führen in ihrer Studie [MC15] weitere Ängste von Konsumenten tragbarer Geräte auf. Sie unterscheiden dabei zwischen Geräten, die am Arm angebracht werden, wie Smart Watches oder Fitnessarmbänder, und am Kopf angebrachte Geräte, wie das Head-Up-Display Google Glass. Bezüglich der ersten Gerätegruppe befürchten die Verbraucher insbesondere die ungewollte Preisgabe ihrer Position und Aktivitäten an Dritte, die durch mangelnde Zugriffskontrolle, dem fehlenden Recht auf Vergessenwerden und der zunehmenden Überwachung durch *Big Brother* entsteht. Hinsicht Head-Up-Displays existieren nach [MC15] ähnliche Ängste; hier kommen jedoch weitere Faktoren, wie Sprach- und Gesichtserkennung, sowie die Verknüpfung mit sozialen Medien hinzu.

Auch bezüglich des Internets der Dinge (Internet of Things; IoT) existieren seitens der Nutzer Bedenken hinsichtlich der Einhaltung des Datenschutzes. Laut der Studie von Babun et al. [BCMU21] haben 83 % der Befragten allgemeine Bedenken, wenn es um die Nutzung von IoT-Systemen geht. Dabei sehen 72 % der Befragten bereits die Sammlung der Daten als kritisch an, während 53 % der Befragten die Auswertung ihres Verhaltens und ihrer Gewohnheiten auf Basis dieser Daten als bedenklich empfinden. 97 % der Befragten wünschen sich im Hintergrund automatisch agierende Prozesse, welche die Analysen der IoT-Anwendungen so abändern, dass ihre Privatsphäre gewahrt wird.

Soziale Auswirkungen von ubiquitären Systemen

In [Lan01] werden vier Kriterien diskutiert, welche die Unterschiede bzgl. sozialer Auswirkungen zwischen dem normalen Einsatz von Informationssystemen und den spezielleren Assistenzsystemen hervorheben:

- **Allgegenwart:** Die Entscheidungen des Assistenzsystems betreffen jeden Lebensbereich einer Person; sogar alltägliche Handlungen, wie das Überqueren einer Straße.
- **Unsichtbarkeit:** Durch die Integration der Hardware eines Assistenzsystems in Möbel und Kleidung bzw. Automobile und Infrastruktur erkennt der Nutzer nicht, ob und mit welchen Geräten er interagiert bzw. ob er überwacht wird.
- **Wahrnehmung:** Bei der Verwendung von Sensoriksystemen besteht die Gefahr, dass aus den erhobenen Daten unerwünschte Rückschlüsse auf Emotionen, wie Stress oder Angst, gezogenen werden können.
- **Datenspeicherung:** Durch die Erhebung der Daten besteht die Gefahr, dass diese dauerhaft gespeichert werden und für andere Zwecke missbraucht werden.

Auf Basis dieser Befürchtungen müssen Assistenzsysteme so entworfen werden, dass sie den Alltag einer Person nicht negativ beeinflussen. Bei der Konzeption von assistiven Systemen müssen Datenschutzbedenken bereits früh beachtet und entsprechende Lösungen in die zu entwickelnde Hard- und Software integriert werden.

Datenschutzanforderungen an Smarte Systeme für die Altenpflege

In einer Studie [Kol15] wurden konkrete Vorschläge für die Ausgestaltung von Assistenzsystemen für die Altenpflege entwickelt. Zu den wichtigsten Aspekten gehören:

- **Proaktiv, nicht reaktiv; Vorbeugend, nicht behebend:** Die Funktion des Systems vor den Datenschutz stellen, sonst ist die Sicherheit der Älteren gefährdet.
- **Datenschutz als Standard**: Verschlüsselung und die Einschränkung von Datenzugriffen sollen standardmäßig vorgegeben werden, da ältere Menschen Probleme haben, selbst die richtigen Einstellungen zu treffen.
- **Datenschutz eingebettet in das Design:** Das Assistenzsystem kümmert sich eigenständig um die Zertifizierung sowie die Verschlüsselung der Daten und speichert Daten lokal ab.
- **Funktionalität bewahren:** Datensparsamkeit umsetzen, außer es entstehen dadurch Nachteile für den Nutzer, z. B. durch die fehlende Überwachung von Vitalparametern.
- **Lebenszyklus der Daten:** Die Zweckbindung der Auswertung muss über den gesamten Verarbeitungszeitraum sichergestellt werden.
- **Sichtbarkeit und Transparenz:** Es muss klar formuliert werden, wer welche Daten zu einem bestimmten Zweck verarbeitet. Die Nutzer sind direkt bei der Einrichtung des Systems mit einzubinden und es muss Ihnen der Zweck des Systems erklärt werden. Zudem sollte eine schriftliche Betriebserlaubnis eingefordert werden.
- **Respekt für die Privatsphäre der Benutzer:** Das System muss alle erforderlichen zutreffenden Gesetze bzw. Regelungen einhalten, unabhängig davon, ob diese formal oder informal festgehalten wurden.

In den späteren Abschnitten 2.4 und 2.5 wird noch genauer auf die gesetzlichen Grundlagen des Datenschutzes und deren technischen Umsetzungen eingegangen. Zuvor werden in den Abschnitten 2.2 und 2.3 mit Datenbanksystemen und Cloud, Fog, Dew bzw. Edge Computing die wichtigsten Technologien zur Umsetzung von Assistenzsystemen vorgestellt.

2.2 Datenbanken

Datenbanken sind Teil komplexer Informationssysteme. Sie dienen vorrangig der Speicherung großer Datenmengen, können aber in Verbindung mit einer Anfragesprache ebenso komplexe Auswertungen auf dem Datenbestand vornehmen. Datenbanksysteme können vielfältig eingesetzt werden: In Cloud-basierten Systemen als Teil von großen Datenbank-Clustern, wie beispielsweise Postgres-XL, als eigenständiges (Hauptspeicher-) Datenbanksystem auf einem einzelnen Großrechner, wie der Einsatz von SAPs HANA-DB bei PayPal mit 48 TB Speicher[6], oder als Standard-Datenbanksystem auf einfachen Rechnern, um Webseiten und private Datenbanken zu verwalten. Datenbanksysteme können auch in entschlackter Form, beispielsweise als Datenbank im Browser[7] oder auf Kleinstrechnern, wie TinyDB [MFHH05], in ressourcenbeschränkten Umgebungen zum Einsatz kommen. Daneben existieren auch Stromdatenbanksysteme, welche Datenströme filtern, vorverdichten und analysieren können.

In diesem Abschnitt werden die Konzepte relationaler Datenbanken an einem durchgängigen Beispiel erklärt, das im nächste Unterabschnitt 2.2.1 vorgestellt wird. Nach einer kurzen Einführung in den Aufbau von Datenbankmanagementsystemen wird das zugrunde liegende relationale Modell (Unterabschnitt 2.2.2) eingeführt, auf welchem die in Unterabschnitt 2.2.3 vorgestellten Anfragesprachen verwendet werden.

Nach E. F. Codd [Cod82] müssen Systeme zur Datenspeicherung diverse Anforderungen, wie Integritätsbedingungen, Transaktionskontrolle und Datenkonsistenz, erfüllen, damit diese als *Datenbankmanagementsysteme*

[6] SAP Bank Analyzer powered by SAP HANA: `https://www.facebook.com/watch/?v=1331697070222416`, zuletzt aufgerufen am 05.01.2022

[7] Siehe beispielsweise SQLite Browser für Chrome: `https://chrome.google.com/webstore/detail/sqlite-manager/njognipnngillknkhikjecpnbkefclfe`, zuletzt aufgerufen am 05.01.2022

bezeichnet werden können. Die für diese Arbeit wichtigste Anforderung ist die Bereitstellung von Autorisierungskomponenten, damit der Zugriff auf den Datenbestand bzw. die Manipulation der Daten durch zuvor festgelegte Regeln erfolgt.

Es muss sichergestellt werden, dass kein unautorisierter Zugriff auf die Datenbank bzw. einzelne Relationen, Tupel oder Attributwerte erfolgt. Durch die Bereitstellung von Sichten auf die Daten wird sichergestellt, dass Nutzer bzw. Nutzerrollen nur auf die Daten zugreifen dürfen, über welche sie die entsprechenden Zugriffsrechte verfügen.

Die Details zu den Zugriffskontrollverfahren werden später im Anhang A.2 weiter ausgeführt. Zunächst erfolgt eine kurze Einführung in das relationale Modell und die dazugehörigen Anfragesprachen, bevor darauf aufbauend die Verfahren zur Zugriffskontrolle vorgestellt werden.

Die verschiedenen Basistechniken der relationalen Datenbanken dienen als Ausgangsbasis für weiterführende Konzepte. Diese werden vorerst nicht in diesem Kapitel vorgestellt, sondern als State-of-the-Art-Abschnitt in den jeweiligen Detailkapiteln behandelt.

2.2.1 Durchgängiges Beispiel

Als durchgängiges Beispiel wird eine Musikdatenbank verwendet. Das Schema beruht auf dem Programm Amarok[8], einer Software zur Verwaltung von Musik und zum Erstellen von Wiedergabelisten. Neben der Wiedergabe von lokal gespeicherter Musik erlaubt Amarok die Integration verschiedener Web Services, wie beispielsweise die Einbindung von last.fm[9], bzw. die Anbindung selbst programmierter Programme über eine API.

Die Struktur der Datenbank ist in Abbildung 2.2 als Entity-Relationship-Modell abgebildet. Der Aufbau der Musikdatenbank ist an das aus dem Bereich des Data Warehousing bekannte Starschema angelehnt.

Im Zentrum befindet sich eine Faktentabelle – `Tracks` – mit Detailinformationen über die Musikstücke. Die Faktentabelle wird um mehrere Dimensionen, wie das dazugehörige Album, das Erscheinungsjahr oder das Genre, ergänzt, die in einer 1:n-Beziehung stehen. Vom Starschema ergeben sich lediglich zwei Abweichungen: Zum einen verwendet die Faktentabelle einen eigenen künstlichen Primärschlüssel anstatt eines zusammengesetzten Schlüssels aus den Fremdschlüsselattributen der Dimensionen. Außerdem existiert eine weitere Beziehung zwischen `Albums` und `Artists`.

Die Dimensionstabellen sind alle gleichartig aufgebaut: Jede Dimension verfügt über eine eindeutige `id` (mit dem jeweils ersten Buchstaben des Relationennamens vorangestellt) und dem dazugehörigen Wert, der im Attribut `name` gespeichert wird. Für die `Albums`-Relation wird zudem die `artid` des dazugehörigen Künstlers als Fremdschlüssel gespeichert. Zusätzlich zum ursprünglichen Kern der Datenbank wurde ein User-Entitätstyp hinzugefügt, welcher die Nutzer der Musikdatenbank repräsentiert.

In diesem laufenden Beispiel soll die Privatsphäre der Nutzer, die in der `Users`-Relation eingetragen wurden, gewahrt werden. Niemand ohne Befugnis soll nachvollziehen können, wie die Beziehung zwischen dem einzelnen Nutzer und seinen getätigten Käufen aussieht. Ein Nutzer soll stets abstreiten können, dass er in der Datenbank eingetragen ist bzw. bestimmte Musikstücke gehört hat. Es sollen hingegen weiterhin statistische Auswertungen auf der Datenbank möglich sein, um beispielsweise Empfehlungssysteme zu realisieren, damit die zum Musikgeschmack des jeweiligen Nutzers passende Musik abgespielt wird.

Die personenbezogenen Daten scheinen zunächst leicht überschaubar, da nur wenige Daten in der `Users`-Relation gespeichert werden. Jedoch lassen sich aus den getätigten Einkäufen und den Damit verbunden Daten über die Musikstücke Rückschlüsse auf einzelne Personen ziehen. Ein ähnlicher Fall ist im Kontext der Erkennung von Medien-Piraterie [EGKP11] bekannt.

2.2.2 Das relationale Modell

Als Grundlage für den Datenbankentwurf müssen die Struktur der Daten in einem Datenbankmodell festgelegt werden. Nach Brodie [Bro84] muss ein Datenbankmodell aus den folgenden Komponenten bestehen:

[8] Webseite zum Amarok Music Player: `https://amarok.kde.org/de`, zuletzt aufgerufen am 05.01.2022

[9] Webseite von last.fm: `https://www.last.fm/`, zuletzt aufgerufen am 05.01.2022

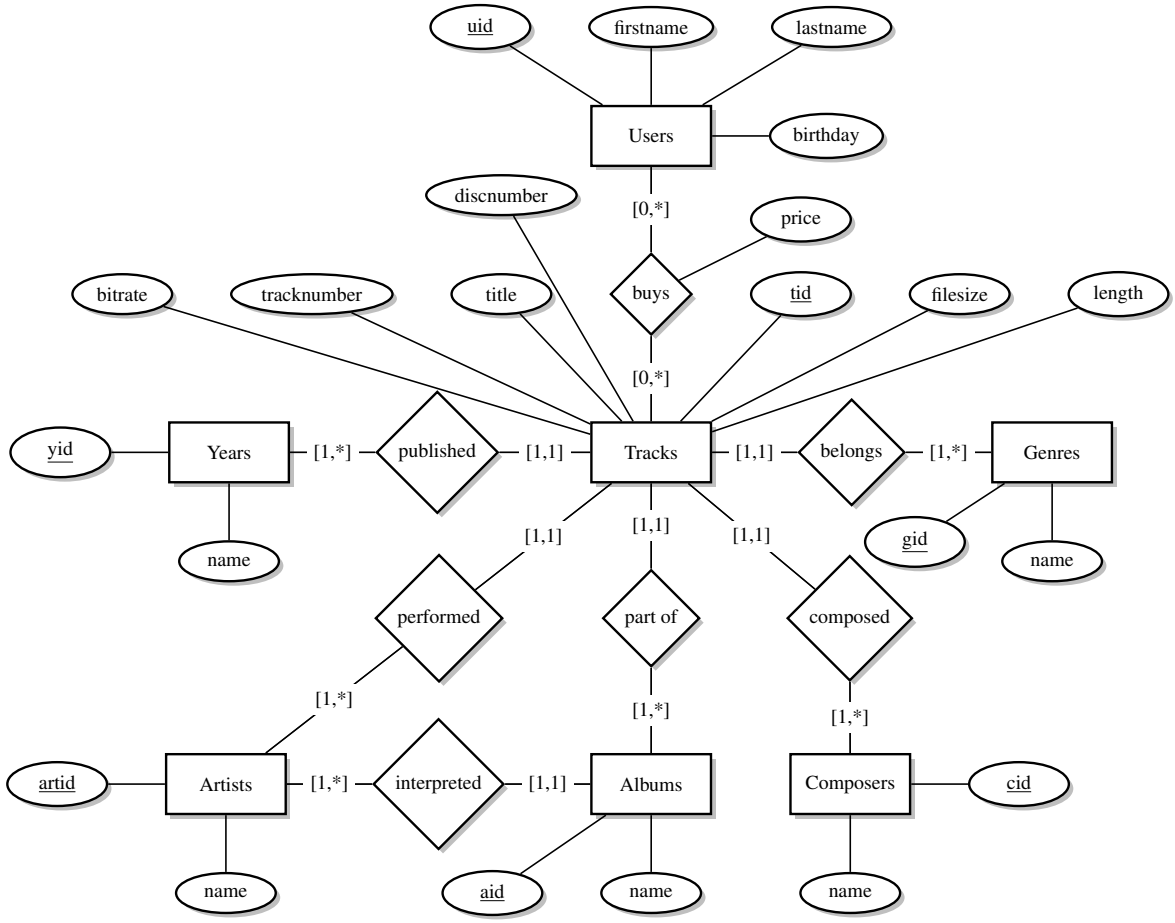


Abbildung 2.2: Das Entity-Relationship-Diagramm für die Musikdatenbank.

Definition: Datenbankmodell nach Brodie [Bro84]

A data model is a collection of mathematically well defined concepts that help one to consider and express the static and dynamic properties of data intensive applications. [...] It is generally assumed that an application can be characterized by:

- Static properties such as objects, object properties [...], and relationships amongst objects [...]
- Dynamic properties such as operations on objects, operation properties, and relationships amongst operations [...]
- Integrity rules over objects [...] and operations [...]

Zur Darstellung von strukturierten Daten wurden in der Vergangenheit Konzepte wie das hierarchisches Datenbankmodell oder das Netzwerkdatenbankmodell entwickelt, in der neueren Zeit aber auch Konzepte für objektrelationale oder gar objektorientierte Datenbanken. In der Praxis hat sich hingegen das relationale Datenbankmodell etabliert, welches an dieser Stelle kurz vorgestellt wird.

Das relationale Modell lässt sich mit einer Tabelle vergleichen. Eine Relation besteht aus einer Menge von Tupeln, wobei jedes Tupel aus den gleichen Attributen besteht. Anders als in den Tabellen dürfen in Relationen keine Tupel doppelt vorkommen. Ebenso ist die Reihenfolge der Tupel nicht von Bedeutung, da diese als Mengen betrachtet werden. Zudem sind die einzelnen Attributwerte atomar, d. h., ein Attributwert kann weder tupel-, noch mengenwertig sein.

Verwendete Notation

In dieser Arbeit wird die gebräuchliche Notation für das relationale Modell eingesetzt, wie sie in vielen Lehrbüchern, beispielsweise in [SSH18], verwendet wird. Die in diesem Unterabschnitt definierten Definitionen basieren, sofern nicht anders angegeben, auf dem oben genannten Lehrbuch bzw. den darauf aufbauenden Vorlesungsfolien. Wir beginnen dabei bei der kleinsten Einheit, dem Attribut, und arbeiten uns bis zum Datenbankschema vor. Eine Übersicht zu funktionalen und Inklusionsabhängigkeiten schließt die Definitionsübersicht ab.

Attribut: Ein Attribut ist das kleinste Schemaelement im relationalen Modell. In den dazugehörigen Attributwerten, die einem bestimmten Wertebereich angehören, werden die atomaren Informationen abgelegt. Attribut, Attributwert und Wertebereich sind wie folgt formal definiert:

Definition: Universum, Attribut, Wertebereich nach [SSH18]

Sei U eine nichtleere, endliche Menge, genannt Universum. Mit $A \in U$ wird ein Attribut bezeichnet, dem ein konkreter Datentyp DT, beispielsweise eine Fließkommazahl, ein Zeitstempel oder eine Zeichenfolge fester Länge, zugeordnet wird. Der Wertebereich von A ($dom(A)$) wird durch eine total definierte Funktion $dom : U \rightarrow DT$ bestimmt. $a_i \in dom(A)$ bezeichnet einen konkreten Attributwert von A.

Beispiel: Attribut

Zwei Attribute aus dem laufenden Beispiel sind `title` und `length`. Das Attribut `title` ist der Datentyp *Zeichenkette variabler Länge* (character varying) zugewiesen, während `length` eine *ganze Zahl* ist. Ein möglicher Attributwert für `title` ist *'Torn from the Heavens'*, für `length` der Wert *267* (Sekunden).

Relationenschema: Ein Relationenschema fasst Attribute zu logischen, zusammengehörenden Einheiten zusammen. Relationenschemata werden i. d. R. durch einen Namen gekennzeichnet. Das Relationenschema ist wie folgt formal definiert:

Definition: Relationenschema nach [SSH18]

Eine Teilmenge $R \subseteq U$, $R \neq \emptyset$ wird als Relationenschema bezeichnet. Ein Relationenschema besteht folglich aus einem oder mehreren Attributen: $R = \{A_1, \ldots A_n\}$.

Beispiel: Relationenschema

Im laufenden Beispiel besteht die Relation `Users` aus den Attributen `uid`, `firstname`, `lastname` und `birthdate`: $Users = \{uid, firstname, lastname, birthdate\}$.

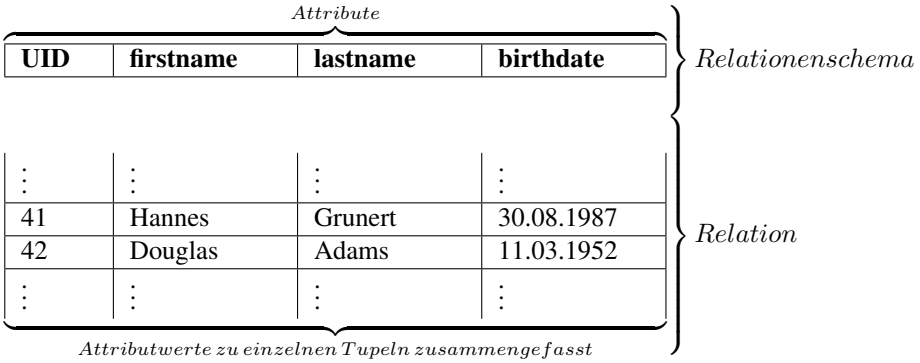
Tupel und Relation: Das Relationenschema dient somit als Vorlage zur Definition einzelner zusammenhängender Informationen, den sogenannten *Tupeln*. Eine Menge von Tupeln, die dem gleichen Relationenschema zugeordnet sind, werden als *Relation* bezeichnet. Tupel und Relation sind wie folgt formal definiert:

Definition: Relation nach [SSH18]

Eine Relation r über $R = \{A_1, \ldots, A_n\}$, $r(R)$, ist eine endliche Menge von totaldefinierten Abbildungen $t : R \rightarrow \bigcup_{A_i} dom(A_i)$, die Tupel genannt werden. Pro Tupel wird jedem Attribut ein Attributwert zugewiesen.

Beispiel: Relation

Eine Relation und das dazugehörige Relationenschema lassen sich anschaulich durch Tabellen darstellen. Die folgende Abbildung zeigt einen Ausschnitt aus der Relation über dem Relationenschema `Users`:



Schlüssel: Für jede Relation existiert im relationalen Modell mindestens eine Menge von Attributen, welche die Tupel eindeutig identifiziert. Diese identifizierende Attributmenge ist wie folgt formal definiert:

Definition: Identifizierende Attributmenge, Schlüssel nach [SSH18]

Eine identifizierende Attributmenge für eine Relation R ist eine Menge $K := \{A_1, \ldots, A_n\} \subseteq R$ mit $\forall t_1, t_2 \in R : (t_1 \neq t_2 \Rightarrow \exists A_1 \in K : t_1(A_i) \neq t_2(A_i))$.

Als Schlüssel wird bzgl. $\subseteq$ eine minimale, identifizierende Attributmenge bezeichnet. Eine Relation kann einen oder mehrere Schlüssel enthalten. Als Primärschlüssel wird ein speziell ausgezeichneter Schlüssel bezeichnet.

Beispiel: Schlüssel

Der Primärschlüssel der obigen Beispielrelation ist das Attribut `uid`. Abhängig vom konkreten Datenbestand könnte die Attributkombination aus `firstname, lastname` und `birthdate` als alternativer Schlüssel verwendet werden.

Datenbankschema: In relationalen Datenbanken wird in der Regel mehr als ein Relationenschema angelegt. Eine Menge von Relationenschemata wird als Datenbankschema bezeichnet, welches wie folgt formal definiert ist:

Definition: Datenbankschema nach [SSH18]

Als Datenbankschema wird eine Menge von Relationenschemata $S := \{R_1, \ldots, R_n\}$ mit $n \in \mathbb{N}$ bezeichnet.

Beispiel: Datenbankschema

Im laufenden Beispiel existieren insgesamt acht Relationenschemata: Für jeden der sieben Entitätstypen existiert je ein Relationenschemata, zusätzlich wird für den Beziehungstypen `buys` ein eigenes Relationenschema erzeugt. Die einzelnen Relationenschemata bilden zusammen das Datenbankschema `amarok`.

Datenbankinstanz: Die Relationen, deren dazugehörige Relationenschemata zum gleichen Datenbankschema gehören, werden zusammengefasst als Datenbankinstanz bezeichnet. Diese lässt sich wie folgt formal definieren:

Definition: Datenbankinstanz nach [SSH18]

Eine Datenbankinstanz (kurz: Datenbank) d über dem Datenbankschema $S = \{R_1, \ldots, R_n\}$ ist eine Menge von Relationen $d := \{r_1, \ldots, r_n\}$, wobei $\forall i \in \{1, \ldots, n\} : r_i$ die Relation über dem Relationenschema R_i ist.

Des Weiteren existieren in relationalen Datenbanken lokale und globale *Integritätsbedingungen*, welche die Menge der erlaubten Relationen einschränken. Eine bereits vorgestellte Bedingung ist die Schlüsselbedingung, welche verhindert, dass eine bestimmte Belegung einer identifizierenden Attributmenge mehrfach auftauchen kann. Mit den *funktionalen Abhängigkeiten* und den *Inklusionsabhängigkeiten* werden im Folgenden zwei weitere, wichtige Konzepte relationaler Datenbanken vorgestellt.

Funktionale Abhängigkeit: Eine funktionale Abhängigkeit liegt vor, wenn durch die Werte einer Attributmenge `X` die Werte einer Attributmenge `Y` eindeutig bestimmt werden können. Für die gleiche Kombination der `X`-Werte müssen folglich stets die gleichen `Y`-Werte zurückgegeben werden. Die funktionale Abhängigkeit zwischen `X` und `Y` wird wie folgt definiert:

Definition: Funktionale Abhängigkeit nach [SSH18]

Eine funktionale Abhängigkeit (engl.: functional dependency; FD) ist ein Ausdruck der Form $X \rightarrow Y$, mit $X, Y \subseteq R$ und $r(R)$. r genügt $X \rightarrow Y$, wenn $\forall c \in X : |\pi_Y(\sigma_{X=c}(r))| \leq 1$ gilt. π_Y kennzeichnet dabei die Projektion der Attributmenge auf die Attribute aus Y, $\sigma_{X=c}$ hingegen die Selektion der Tupel nach der Bedingung $X = c$. Die Operatoren Projektion und Selektion werden im nächsten Unterabschnitt näher betrachtet.

Beispiel: Funktionale Abhängigkeit

In der Relation *tracks* existiert, neben der Schlüsselabhängigkeit, eine funktionale Abhängigkeit von der Dateigröße (*filesize*) und der Bitrate (*bitrate*) auf die Länge (*length*) des Musikstückes; es gilt folglich:
$\{filesize, bitrate\} \rightarrow \{length\}$[a].

[a] In der verwendeten Datenbank gilt dies nicht für 185 der insgesamt 22556 Tupel. In diesen Fällen unterscheidet sich die Laufzeit um eine Sekunde. Zur Veranschaulichung der funktionalen Abhängigkeit lassen wir dies an dieser Stelle außer Acht. In Kapitel 4 wird näher auf *quasi-funktionale Abhängigkeiten*, auch bekannt als *Partial* bzw. *Approximate Functional Dependencies*, eingegangen.

Inklusionsabhängigkeit: Inklusionsabhängigkeiten sind verallgemeinerte Fremdschlüsselbedingungen der Form $X \rightarrow Y$, wobei die Y-Werte nicht unbedingt ein Primärschlüssel einer Relation sein müssen. Formal lassen sich Inklusionsabhängigkeiten wie folgt definieren:

Definition: Inklusionsabhängigkeit nach [SSH18]

Seien r_1 und r_2 Relationen über R_1 bzw. R_2. Die Inklusionsabhängigkeit $R_1[X] \subseteq R_2[Y]$ ist für zwei Attributmengen $X \subseteq R_1$ und $Y \subseteq R_2$ und eine Instanz d gültig, wenn $\pi_X(r_1) \subseteq \pi_Y(r_2)$ gilt.

Beispiel: Inklusionsabhängigkeit

In der Beispieldatenbank befinden sich neben den Fremdschlüsselbedingungen keine weiteren Inklusionsabhängigkeiten. Eine Fremdschlüsselbedingung fordert beispielsweise, dass die IDs der Komponisten (cid) in der Relation $tracks$ eine Teilmenge der IDs aus der Relation $composers$ sind:
$\pi_{cid}(tracks) \subseteq \pi_{cid}(composers)$.
Somit können in der Datenbank Komponisten erfasst werden, auch wenn Sie (noch) keinen Musikstücken zugeordnet sind.

Das relationale Modell für das laufende Beispiel

Das durchgängige Beispiel ist im Relationenmodell wie folgt umgesetzt:

$$
\begin{aligned}
&Tracks(\{tid, tracknumber, title, discnumber, bitrate, filesize, length,\\
&\quad yid, gid, artid, aid, cid\}, \{\{tid\}\})\\
&\quad Tracks(yid) \rightarrow Years(yid)\\
&\quad Tracks(gid) \rightarrow Genres(gid)\\
&\quad Tracks(artid) \rightarrow Artists(artid)\\
&\quad Tracks(aid) \rightarrow Albums(aid)\\
&\quad Tracks(cid) \rightarrow Composers(cid)\\
&Years(\{yid, name\}, \{\{yid\}\})\\
&Genres(\{gid, name\}, \{\{gid\}\})\\
&Artists(\{artid, name\}, \{\{artid\}\})\\
&Albums(\{aid, name, artid\}, \{\{aid\}\})\\
&\quad Albums(artid) \rightarrow Artists(artid)\\
&Composers(\{cid, name\}, \{\{cid\}\})\\
&Users(\{uid, firstname, lastname, birthday\}, \{\{uid\}\})\\
&listens(\{uid, tid\}, \{\{uid, tid\}\})\\
&\quad listens(uid) \rightarrow Users(uid)\\
&\quad listens(tid) \rightarrow Tracks(tid)
\end{aligned}
\tag{2.1}
$$

Jede Relation erhält eine eindeutige ID mit vorangestellten, gekürzten Relationsnamen, damit diese im Schema einzigartig ist. Die ID ist zeitgleich alleiniger Schlüssel der entsprechenden Relation. Zudem werden alle Attribute des jeweiligen Entitätstypen in die Relation übertragen. Auf eine eindeutige Benennung der jeweiligen `name`-Attribute wird zugunsten einer einfacheren Anfrageformulierung für spätere Beispiele verzichtet.

Die Relation `tracks` enthält zusätzlich die Schlüssel der verschmolzenen Beziehungstypen, d. h. alle IDs der anderen Relationen als Fremdschlüssel. Gleiches gilt für die Relation `albums`; hier wird die ID des zugehörigen `artists` als Fremdschlüssel aufgenommen. Da mit Ausnahme des Beziehungstypens `listens` immer eine 1:n-Beziehung vorliegt, braucht für die Beziehungstypen keine eigene Relation angegeben werden.

2.2.3 Anfragesprachen

Als *Anfrage* wird eine Abfolge von Operationen bezeichnet, die eine oder mehrere Ausgangsrelationen in eine Ergebnisrelation überführen. Benannte Anfragen werden als *Sicht* bezeichnet. Anfragesprachen müssen nach [HS91] verschiedene Kriterien, wie

- **Abgeschlossenheit:** Das Anfrageergebnis ist wieder eine Relation,
- **Deskriptivität:** Die Formulierung der Anfrage erfolgt deklarativ,
- **Optimierbarkeit:** Es existieren Optimierungsregeln für die Anfragesprache, und
- **Vollständigkeit:** Die Anfragesprache unterstützt alle Konstrukte der zugrunde liegenden Algebra.

erfüllen.

Für relationale Strukturen wurden diverse Anfragesprachen und Algebren entwickelt. In diesem Abschnitt stellen wir im Folgenden die drei wichtigsten Sprachen, die relationale Algebra, SQL und Datalog, vor. Während SQL insbesondere zur Formulierung von Anfragen in Datenbanksystemen eingesetzt wird, erfolgt die Verwendung der relationalen Algebra insbesondere in der logischen Anfrageoptimierung. Datalog wird zudem in vielen Publikationen zur Datenbanktheorie, insbesondere zum Query-Containment-Problem, verwendet, welches in Kapitel 5 näher betrachtet wird.

Relationale Algebra

In Kapitel 6 wird ein neuer Ansatz zur Lösung eines *Query Containment Problems* vorgestellt. Im Gegensatz zu den meisten Publikationen in diesem Bereich verwendet der Ansatz als zugrunde liegende Anfragesprache nicht Datalog, sondern setzt im Kern auf die relationale Algebra. Zu den betrachteten Operatoren der relationalen Algebra werden in dieser Arbeit die Projektion (π), die Selektion (σ), der Verbund ($\bowtie$), die Umbenennung (β), sowie die Mengenoperatoren Vereinigung ($\cup$), Durchschnitt ($\cap$) und Differenz ($-$) betrachtet. Außerdem wird die Aggregation und die Gruppierung mit dem Operator γ abgedeckt. Wie bereits im vorherigen Unterabschnitt wird, sofern nicht anders angegeben, die Notation aus [SSH18] zur Definition der relationalen Operatoren verwendet.

Projektion: Die Projektion dient zum Ausblenden von Attributen. Es werden Attribute einer Relation bestimmt, die in der Ergebnisrelation erhalten bleiben sollen. Dabei werden alle Attribute aus der Ergebnisrelation entfernt, die nicht in der Projektionsliste vorhanden sind.

Definition: Projektion nach [SSH18]

Die Syntax der Projektion ist durch $\pi_X(R)$ gegeben, wobei $X \subseteq R$ eine Teilmenge der Attribute des Relationsschemas R ist. In die Ergebnisrelation werden dabei alle Attributwerte übernommen, welche in der auf X eingeschränkten Tupeln t aus der Relation r über R vorkommen: $\pi_X(r) := \{t(X) | t \in r\}$.

Beispiel: Projektion in der relationalen Algebra

Der folgende algebraische Ausdruck gibt die Attributwerte für die Attribute *title* und `length` der Relation `tracks` aus:

$$\pi_{title,length}(tracks)$$

title	**length**
The Man With The Machine Gun	216000
Answers	429000
Somnus	148000
...	...

Selektion: Die Selektion dient zum Filtern von Tupeln. Durch eine Menge von Selektionsprädikaten C wird bestimmt, welche Tupel einer Relation in das Ergebnis einfließen. Die Syntax der Selektion ist durch $\sigma_C(R)$ gegeben. C ist eine Menge von Bedingungen über den Attributen der Relation R, die durch boolesche Operatoren $\wedge$ und $\vee$ miteinander verknüpft bzw. durch den booleschen Operator $\neg$ negiert werden.

Typische Selektionsbedingungen sind Attribut-Konstanten-Selektionen $x_i \theta c$ und Attribut-Attribut-Selektionen $x_i \theta x_j$, wobei der Vergleichsoperator θ aus der Menge $\{<, \leq, =, \neq, \geq, >\}$ stammt und x_i, x_j Attribute von R sind. Als Kurzform und zur besseren Unterscheidung der beiden Selektionsarten verwenden wir im Folgenden θ und θ_c für die Attribut-Attribut-Selektion bzw. Attribut-Konstanten-Selektion. Verallgemeinert lässt sich die Selektion auch auf tupelwertige Attributkombinationen anwenden. Im allgemeinen Fall ist die Semantik der Selektion durch $\sigma_C(r) := \{t | t \in r \wedge C(t) = true\}$ bestimmt. Die aussagenlogische Bedingung C lässt sich dabei auf die oben erwähnten Konstanten- und Attributselektionen zurückführen:

Definition: Konstantenselektion nach [SSH18]

Sei $X \subseteq R$, c Attributwert von X und und θ Einer der Vergleichsoperatoren $<, \leq, =, \geq, >$ oder $\neq$ (Kurform: $\theta \in \{<, \leq, =, \geq, >, \neq\}$) ist. Die Konstantenselektion $X \theta c$ auf der Relation r ist definiert als $\sigma_{X \theta c}(R) := \{t | t \in r \wedge t(X) \theta c\}$.

Die Attributselektion lässt sich auf Basis der Konstantenselektion wie folgt definieren:

Definition: Attributselektion nach [SSH18]

Seien $X \subseteq R, Y \subseteq R$ mit $|X| = |Y|$ und θ Einer der Vergleichsoperatoren $<, \leq, =, \geq, >$ oder $\neq$ ist. Die Attributselektion $X \theta Y$ auf der Relation r ist definiert als $\sigma_{X \theta Y}(R) := \{t | t \in r \wedge t(X) \theta t(Y)\}$.

Voraussetzung dabei ist, dass die einzelnen Attributpaare aus X und Y den Vergleichsoperator θ unterstützen. Das nachfolgende Beispiel zeigt eine einfache Selektion mit zwei durch ein logisches Und verknüpfte Konstantenselektionen:

Beispiel: Selektion in der relationalen Algebra

Der folgende algebraische Ausdruck gibt nur diejenigen Tupel aus der Datenbank aus, bei denen der Attributwert für das Attribut `length` kleiner als `150000` ist und der Attributwert für das Attribut `discnumber` größer als `1` ist:

$$\sigma_{length<150000 \wedge discnumber>1}(tracks)$$

tid	**discnumber**	**tracknumber**	**title**	**length**	...
20	2	7	Victory Theme	12000	...
24	2	11	Prima Vista Orchestra	93000	...
41	4	6	Apocalypsis Noctis	131000	...
...	...	...	...	...	...

Umbenennung: Die Umbenennung ermöglicht die Änderung des Namens einzelner Attribute. Voraussetzung für die mengentheoretischen Operationen Vereinigung, Durchschnitt und Differenz ist, dass die beteiligten Relationen über das gleiche Relationenschema verfügen. Besitzen die Relationen eine gleiche Struktur, haben jedoch unterschiedliche Bezeichnungen für die Attribute, so können durch die Umbenennung die Relationenschemata vereinheitlicht werden. Gleiches gilt auch für den natürlichen Verbund. Die Umbenennung ist wie folgt definiert:

Definition: Umbenennung nach [SSH18]

Die Umbenennung β des Attributes A zu B in der Relation r über dem Relationenschema R ist für $A \in R, B \notin R - \{A\}, R' := (R - \{A\}) \cup \{B\}$ und $dom(A) = dom(B)$ definiert durch $\beta_{B \leftarrow A}(r) := \{t' | \exists t \in r : t'(R - \{A\}) = t(R - \{A\}) \wedge t'(B) = t(A)\}$.

Beispiel: Umbenennung in der relationalen Algebra

Durch die folgende Anfrage wird das Attribut `name` der Relation `artists` in `Musiker` umbenannt. Das Attribut `artid` bleibt dabei erhalten.

$\beta_{Musiker \leftarrow name}(artists)$

artid	**Musiker**
2016	Susan Calloway
2017	Benyamin Nuss
2620	Lenny Kravitz
...	...

Verbund: Der Verbund $\bowtie$ dient zur Verknüpfung der Tupel zweier Relationen über eine festgelegte Verbundbedingung. Als spezieller Verbund kann das Kreuzprodukt $R \times S$ angesehen werden. Sind keine gleich benannten Attribute in den Relationen R und S vorhanden ($R \cap S = \emptyset$), so wird in die Ergebnisrelation die Kombination jedes Tupels aus R mit jedem Tupel aus S übernommen. Gilt $R = S$, so wird der Verbund zum mengentheoretischen Durchschnitt. Beim natürlichen Verbund werden die Tupel über gleich benannte Attribute miteinander zu einer Ergebnisrelation verknüpft. Der natürliche Verbund ist formal wie folgt definiert:

Definition: Natürlicher Verbund nach [SSH18]

Der natürliche Verbund der Relationen $r_1 \bowtie r_2$ über den Relationenschemata R_1 bzw. R_2 ist bestimmt durch $r_1 \bowtie r_2 := \{t | t \text{ Tupel über } R_1 \cup R_2 \wedge (\forall i \in \{1, 2\} : \exists t_i \in r_i : t_i = t(R_i))\}$.

Beispiel: Natürlicher Verbund in der relationalen Algebra

Die folgende Anfrage verknüpft die Relationen `tracks` und `albums` über das gemeinsame Attribut `aid`:

$tracks \bowtie albums$

...	**title**	**aid**	**name**	...
...	Victory Theme	4711	Distant Worlds II	...
...	Prima Vista Orchestra	4711	Distant Worlds II	...
...	Broken Down	2342	Final Fantasy XV OST	...
...	...	...	...	...

Vereinigung: Durch die Vereinigung werden die Tupel zweier Relationen r_1 und r_2 über dem gleichen Relationenschema R in eine gemeinsame Relation überführt. Bei der Vereinigung werden, sofern nicht explizit unter Multimengensemantik betrachtet, Duplikate entfernt. Die Vereinigung ist wie folgt formal definiert:

Definition: Mengentheoretische Vereinigung nach [SSH18]

Seien r_1 und r_2 zwei Relationen über dem Relationenschema R. Die Vereinigung $r_1 \cup r_2$ wird bestimmt durch $r_1 \cup r_2 := \{t | t \in r_1 \vee t \in r_2\}$.

Beispiel: Mengentheoretische Vereinigung in der relationalen Algebra

Die folgende Anfrage gibt die Namen aller Komponisten und Musiker aus den Relationen `composers` bzw. `artists` aus. Da die Relationenschemata nicht identisch sind, erfolgt zuvor jeweils eine Projektion auf das Attribut `name`.

$$\pi_{name}(composers) \cup \pi_{name}(artists)$$

name
Bob Dylan
Lenny Kravitz
Foo Fighters
...

Durchschnitt: Der Durchschnitt zweier Relationen r_1 und r_2 über dem gleichen Relationenschema R enthält alle Tupel, die sowohl in r_1 als auch in r_2 vorkommen. Die formale Definition des mengentheoretischen Durchschnitts wird wie folgt festgelegt:

Definition: Mengentheoretischer Durchschnitt nach [SSH18]

Seien r_1 und r_2 zwei Relationen über dem Relationenschema R. Der Durchschnitt $r_1 \cap r_2$ wird bestimmt durch $r_1 \cap r_2 := \{t | t \in r_1 \wedge t \in r_2\}$.

Beispiel: Mengentheoretischer Durchschnitt in der relationalen Algebra

Die folgende Anfrage gibt die Namen aller Komponisten aus, die auch Musiker sind. Da die Relationenschemata `composers` und `artists` nicht identisch sind, erfolgt zuvor jeweils eine Projektion auf das Attribut `name`.

$$\pi_{name}(composers) \cap \pi_{name}(artists)$$

name
Lenny Kravitz
Foo Fighters
Masayoshi Soken
...

Differenz: Durch die Differenz werden von einer Relation r_1 diejenigen Tupel abgezogen, die in einer zweiten Relation r_2 über dem gleichen Relationenschema R vorkommen. Die Differenz ist wie folgt formal definiert:

Definition: Mengentheoretische Differenz nach [SSH18]

Seien r_1 und r_2 zwei Relationen über dem Relationenschema R. Die Differenz $r_1 - r_2$ wird bestimmt durch $r_1 - r_2 := \{t | t \in r_1 \wedge t \notin r_2\}$.

Beispiel: Mengentheoretische Differenz

Die folgende Anfrage gibt die Namen aller Komponisten aus, die nicht gleichzeitig Musiker sind. Da die Relationenschemata `composers` und `artists` nicht identisch sind, erfolgt zuvor jeweils eine Projektion auf das Attribut `name`.

$$\pi_{name}(composers) - \pi_{name}(artists)$$

name
Nobuo Uematsu
Steve Wonder
Carlos Santana
...

Gruppierung und Aggregation: Durch die Aggregation werden die Werte eines Attributes zu einem einzelnen Wert zusammengefasst. Mittels einer vorherigen Gruppierung einer Relation nach Werten einer festgelegten Attributmenge liefern Aggregatfunktionen zu jeder Gruppe einen eigenen aggregierten Wert.

Ihrer Definition nach sind Aggregatfunktionen kein zulässiger Operator in der Relationenalgebra, da die Anwendung der Aggregation einen einzelnen Wert zurückliefert. Durch einen Einbettungsoperator, γ, können die einzelnen Werte in eine neu erzeugte Relation eingebettet werden. Dies führt dazu, dass Aggregatfunktionen orthogonal zu anderen Operatoren der Relationenalgebra genutzt werden können. Der Einbettungsoperator ist wie folgt formal definiert:

Definition: Einbettungsoperator γ in Anlehnung an die Notation von [Dit13]

Seien $X, Y_1, \ldots Y_n$ Attributmengen aus der Relation r über dem Relationenschema R und f_i Aggregatfunktionen. Der Einbettungsoperator $\gamma_{[s;]f_1(Y_1),\ldots,f_n(Y_n)}(r)$ ist eine Abbildung, welche für die Ergebnisse der Aggregatfunktionen f_i eine neue Relation s erzeugt und für dessen einziges Tupel t die Gleichung $t = (f_1(Y_1), \ldots, f_n(Y_n))$ erfüllt ist. Die Benennung der Relation in s ist dabei optional:

$\gamma_{[s;]f_1(Y_1),\ldots,f_n(Y_n)}(r) := \{t|t := (f_1(r.Y_1), \ldots, f_n(r.Y_n))\}$

Werden zusätzlich Gruppierungsattribute X angegeben, so wird für jede Belegung von X ein eigenes Tupel mit den dazugehörigen Ergebnissen der Aggregatfunktionen erzeugt:

$\gamma_{[s;]X;f_1(Y_1),\ldots,f_n(Y_n)}(r) := \{t|t := (t.X, \gamma_{f_1(Y_1),\ldots,f_n(Y_n)}(\sigma_{r.X=t.X}(r)))\}$

Im Folgenden betrachten wir die grundlegenden Aggregatfunktionen. Diese sind formal wie folgt definiert:

Definition: Minimum, Maximum und Summe

Sei r eine Relation über dem Relationenschema R und $A \in R$ ein Attribut aus R.

Das Minimum von A ist wie folgt definiert: $\gamma_{MIN(A)}(r) := MIN(\pi_A(r)) := min(\{t.A|t \in r\})$.

Das Maximum von A ist wie folgt definiert: $\gamma_{MAX(A)}(r) := MAX(\pi_A(r)) := max(\{t.A|t \in r\})$.

Die Summe von A ist wie folgt definiert: $\gamma_{SUM(A)}(r) := SUM(\pi_A(r)) := \sum(\{t.A|t \in r\})$.

Definition: Anzahl

Sei r eine Relation über dem Relationenschema R und $A \subseteq R$ eine Attributmenge aus R. Bezeichne $\perp$ den Nullwert.

Die Anzahl der Elemente von A ist wie folgt definiert:

$\gamma_{COUNT(A)}(r) := COUNT(\pi_A(r)) := |\{t.A|t \in r \wedge t.A \neq \perp\}|$.

Nullwerte bzw. Nulltupel und Duplikate werden entsprechend nicht mitgezählt. Die Anzahl kann auch über die gesamte Relation ermittelt werden, wobei die Kardinalität der gesamten Relation als Ergebnis zurückgegeben wird: $COUNT(r) := |\{t|t \in r\}|$.

Einige Aggregatfunktionen lassen sich aus der Kombination von mehreren Aggregaten zusammen mit arithmetischen Funktionen herleiten. Beispielsweise lässt sich der *Durchschnitt* aus den Aggregatfunktionen *Summe* und *Anzahl*, sowie der *Division* berechnen:

Definition: Durchschnitt

Sei r eine Relation über dem Relationenschema R und $A \in R$ ein Attribut aus R, auf dessen Wertebereich $dom(A)$ der Summenoperator und die Division definiert sind. Der Durchschnitt der Attributwerte von A ist wie folgt definiert: $\gamma_{AVG(A)}(r) := AVG(\pi_A(r)) := \frac{\gamma_{SUM(A)}(r)}{\gamma_{COUNT(A)}(r)}$.

Analog zum Durchschnitt lassen sich auf Basis der Standardaggregatfunktionen weitere Aggregate, wie *Standardabweichung* und *Varianz*, definieren. Auf die analytischen Funktionen wird im nächsten Unterabschnitt näher eingegangen. Das folgende Beispiel zeigt den Einsatz von Aggregatfunktionen und dem Einbettungsoperator im Zusammenspiel mit weiteren Operatoren der relationalen Algebra:

Beispiel: Anwendung des Einbettungsoperators bei der Gruppierung und Aggregation

Die folgende Anfrage ermittelt den Namen aller Komponisten, die mit mehr als zehn Tracks in der Musikdatenbank vertreten sind:

$$\pi_{name,Anzahl}(\sigma_{Anzahl>10}(\beta_{Anzahl \leftarrow COUNT(tid)}(\gamma_{R;COUNT(tid);cid}(tracks))) \bowtie composers)$$

name	**Anzahl**
The Beatles	84
R. E. M.	73
Nobuo Uematsu	47
...	...

Eine wichtige Eigenschaft der relationalen Algebra ist die Abgeschlossenheit, d. h., das Ergebnis jedes relationalen Operators, der auf eine Relation angewendet wird, ist wieder eine Relation. Dadurch können Folgen von Operatoren betrachtet werden. Diese spielt insbesondere bei der Optimierung von relationalen Ausdrücken eine wichtige Rolle. Die Optimierungskonzepte werden später in Kapitel 6 näher vorgestellt. Nachfolgend werden die Konzepte der relationalen Algebra auf die Anfragesprache SQL übertragen.

SQL

SQL (Structured Query Language, de.: strukturierte Anfragesprache) dient zur Formulierung von Definitionen, Manipulationen und Anfragen auf relationalen Daten. Im Folgenden betrachten wir nur den Anfrageteil aus dem Foundation-Abschnitt des SQL:2016-Standards [Mel16]. Der folgende Quellcodeausschnitt zeigt die allgemeine Syntax einer einfachen SQL-Anweisung:

```
1 SELECT <attributliste>
2 FROM <relationsliste>
3 [WHERE <bedingungen>]
4 [GROUP BY <attributliste>]
5 [HAVING <bedingungen>]
6 [ORDER BY (<attribut [ASC|DESC])+]
```

Quellcodeausschnitt 2.1: Aufbau einer SQL-Anfrage (vereinfachte Darstellung)

Im Folgenden werden die einzelnen Klauseln kurz vorgestellt. Die dazugehörigen Beispielanfragen beziehen sich dabei, sofern äquivalent darstellbar, auf die in der Relationenalgebra formulierten Anfragen aus dem vorherigen Unterabschnitt.

SELECT-Klausel: Die `SELECT`-Klausel dient vorrangig der Projektion von Attributen der in der `FROM`-Klausel vorkommenden Attribute der gegebenen Relationen. Für gleichbenannte Attribute aus verschiedenen Relationen

kann zusätzlich der Relationenname (bzw. dessen Umbenennung) als Präfix zur eindeutigen Identifizierung angegeben werden. Die auszugebenden Attribute werden nacheinander und durch Komma getrennt angegeben. Zusätzlich können in der `SELECT`-Klausel arithmetische Operationen, Aggregationen und Umbenennungen (`AS`) auf den Attributen angegeben werden. Durch das Schlüsselwort `DISTINCT` erfolgt eine Duplikateliminierung auf der Ergebnismenge, wodurch ein Wechsel von der Multimengensemantik auf die Mengensemantik erfolgt. Bei Aggregationen ist darauf zu achten, dass neben den Aggregaten nur Attribute in der Projektionsliste auftauchen dürfen, die auch in der `GROUP BY`-Klausel auftreten.

FROM-Klausel: Die `FROM`-Klausel gibt an, aus welchen Relationen Daten gelesen werden sollen. Dabei können durch Verbundbedingungen bzw. das kartesische Produkt mehrere Relationen miteinander verknüpft werden. Die Relationen können optional mittels der `AS`-Klausel umbenannt werden.

Beispiel: Beispiel für die SELECT- und FROM-Klausel

```
1 SELECT title, length
2 FROM tracks
```

Statt eines Relationsnamens kann auch ein neuer `SELECT-FROM-WHERE`-Block (SFW-Block) angegeben werden, da mit dem SQL-92-Standard [Mel92] die orthogonale Schachtelung ermöglicht wurde. Werden Daten aus mehr als einer Relation benötigt, so müssen diese durch eine Verbundbedingung verknüpft werden. SQL stellt dafür fünf verschiedene Operatoren bereit:

1. **Kartesisches Produkt:** Das kartesische Produkt von zwei Relationen wird mittels eines Kommas bzw. dem Schlüsselwort `CROSS JOIN` gebildet.
2. **Verbundbedingungen:** Innerhalb der `WHERE`-Klausel lassen sich eine oder mehrere Selektionsbedingungen zwischen Attributen verschiedener Relation formulieren. Auf diese Weise lassen sich ebenso die nachfolgenden drei Arten von Verbundoperatoren realisieren.
3. **θ-Verbund:** Im θ-Verbund lässt sich die Verbundbedingung zwischen zwei Relationen innerhalb der `FROM`-Klausel angeben. Die dazugehörige Syntax lautet:
   `Relation1 JOIN Relation2 ON (Relation1.AttrX = Relation2.AttrY)`
4. **Gleichverbund:** Der Gleichverbund verknüpft zwei Relationen über ein gemeinsames, gleich benanntes Attribut. Die dazugehörige Syntax lautet:
   `Relation1 JOIN Relation2 USING (Attribut)`
5. **Natürlicher Verbund:** Über den natürlichen Verbund werden zwei Relationen über ihre gemeinsamen, gleich benannten Attribute miteinander verknüpft. Die dazugehörige Syntax lautet:
   `Relation1 NATURAL JOIN Relation2`

Beispiel: Natürlicher Verbund

```
1 SELECT *
2 FROM tracks NATURAL JOIN albums
```

Neben dem inneren Verbund (Schlüsselwort `INNER JOIN`), der nur diejenigen Tupel, für welche ein entsprechender Verbundpartner existiert, mit in die Ausgaberelation übernimmt, können durch den äußeren Verbund (Schlüsselwort `OUTER JOIN`) auch die Tupel mit in das Ergebnis übernommen werden, für die kein passender Partner existiert. Dabei wird zwischen den Varianten `LEFT OUTER JOIN`, `RIGHT OUTER JOIN` und `FULL OUTER JOIN` unterschieden. Dabei werden entsprechend alle Tupel der ersten bzw. zweiten Relation in das Ergebnis übernommen und ggf. Attributwerte durch fehlende Verbundpartner mittels Nullwerte aufgefüllt.

WHERE-Klausel: Die `WHERE`-Klausel dient zur Formulierung von Selektions- und Verbundbedingungen aus der Relationenalgebra. Zudem können geschachtelte Anfragen, in Form eines erneuten `SFW`-Blockes, als Relation übergeben werden, um geschachtelte Unteranfragen zu realisieren.

Einzelne Selektionsbedingungen lassen sich in die Konstantenselektion (`attr` θ `konst`), die Attributselektion (`attr1` θ `attr2`), die Bereichsselektion (`attr BETWEEN konst1 AND konst2`) und die Ungewissheitsselektionen (`attr LIKE konst`) untergliedern. Die Ungewissheitsselektion erlaubt die Selektion auf Zeichenketten, deren Wert nicht exakt bekannt ist. Das Zeichen % erlaubt eine beliebige Anzahl (≥ 0) an Zeichen an dessen Position, während das Zeichen _ für genau ein unbekanntes Zeichen steht.

Durch einen Vergleichsoperator θ ($<$, $\leq$, $=$, $\neq$, $\geq$, $>$) lässt sich ein Attribut mit einer Konstanten bzw. einem weiteren Attribut vergleichen. Stammen im Falle des Attributvergleiches die Attribute aus zwei verschiedenen Relationen, wird dadurch eine Verbundbedingung formuliert. Die Bereichsselektion lässt sich zudem durch eine einfache Umschreibung auf zwei `AND`-verknüpfte Konstantenselektionen abbilden.

Als weitere Selektionsbedingungen können Attributwerte gegen den Nullwert geprüft werden (`IS NULL`) bzw. als quantifizierte Bedingung (Allquantor `ALL`, Existenzquantor: `ANY`, `SOME`, Test auf Existenz eines Wertes: `EXISTS`) formuliert werden, wenn der Vergleich auf einer Menge von Attributwerten bzw. Tupeln erfolgt. Die quantifizierten Bedingungen werden insbesondere für geschachtelte Unteranfragen genutzt, um ein Attribut gegen die in der Unteranfrage erzeugte Wertemenge zu testen. Für Unteranfragen kann zudem das `IN`-Prädikat genutzt werden, wobei die Attributmenge als Argument des `IN`-Prädikates auch manuell angegeben werden kann.

Jede Selektionsbedingung kann als boolescher Ausdruck aufgefasst werden und liefert somit als Ergebnis stets einen booleschen Wert (`TRUE` bzw. `FALSE`) zurück. Mehrere Selektionsbedingungen können mit den booleschen Konnektoren `AND`, `OR` und `NOT` miteinander verbunden werden, um komplexere Selektionsbedingungen zu formulieren. Das folgende Beispiel enthält eine Anfrage mit mehreren Selektionsbedingungen:

Beispiel: Anfrage mit mehreren Selektionsbedingungen

```
1 SELECT *
2 FROM tracks
3 WHERE length < 150000
4 AND discnumber > 1
```

Neben dem Vergleich von einzelnen Attributwerten erlauben sogenannte Tupelkonstruktoren den Vergleich von einzelnen Tupeln, die ggf. aus Konstanten gebildet wurden, mit dem Ergebnis einer SQL-Unteranfrage. Zusätzlich zu den Attributen können in der `WHERE`-Klausel und in der `SELECT`-Klausel skalare Ausdrücke verwendet werden. Dies schließt numerische Operatoren, wie die Addition und die Multiplikation, aber auch Operatoren auf Zeichenketten, wie die Berechnung der Länge einer Zeichenfolge, die Bildung von Teilzeichenketten und die Konkatenation von zwei Zeichenketten ein. Auf Datums- und Zeitangaben lassen sich arithmetische Operationen ausführen und das aktuelle Datum bzw. die Zeit ausgeben.

Beispiel: Vergleich mit Tupelkonstruktor

Die folgende Anfrage liefert als Ausgabe alle Musiker, die auch Nutzer der Musikdatenbank sind und deren Kombination aus Vor- und Nachnamen mit dem Musikernamen übereinstimmen. Der Name des Musikers wird dazu in zwei Teilstrings aufgeteilt. Sowohl Syntax als auch die verwendeten Built-in-Funktionen stammen aus PostgreSQL.

```
1 SELECT name
2 FROM Artists
3 WHERE (
4   SUBSTRING(name, 1, strpos(name, ' ')),
5   SUBSTRING(name, strpos(name, ' ')+1, LENGTH(name)+1))
6 IN (
```

```
7   SELECT firstname, lastname
8   FROM Users
9 )
```

Aggregatfunktionen: Die meisten Datenbanksysteme unterstützen die folgenden fünf Aggregatfunktionen: Mit `COUNT` wird die Anzahl der Attributwerte einer Spalte bzw. die Gesamtzahl aller Tupel einer Relation ausgegeben. `SUM` ermittelt die Summe der Werte einer Spalte, `AVG` dessen arithmetisches Mittel. Durch `MAX` bzw. `MIN` wird der größte bzw. kleinste Wert einer Spalte ermittelt. Neben den Attributen, die durch die Relationen aus der `FROM`-Klausel zur Verfügung stehen, können auch beliebige, gültige skalare Ausdrücke als Argument an die Aggregatfunktion übergeben werden.

Weiterführende SQL-Klauseln werden in Anhang A.3 aufgeführt. Diese behandeln u. a. die Gruppierung, Mengenoperationen und neue Klauseln, wie z. B. für Fensterfunkionen. Weiterhin wird in Anhang A.4 mit Datalog auf eine weitere Anfragesprache eingegangen, die insbesondere in der Datenbankforschung eingesetzt wird.

2.3 Cloud, Fog, Edge und Dew Computing

Im Big Data-Zeitalter findet die Speicherung und Verarbeitung von Informationen zunehmend in cloud-basierten Systemen statt. Anbieter von Cloud-Computing-Angeboten bieten eine Vielzahl von Diensten, Geräten und Softwarelösungen zur Speicherung von Daten an. Dies schließt gleichermaßen auch Dienste ein, die in smarten Umgebungen zum Einsatz kommen.

Neben reinen Cloud-Lösungen wurden mit dem Fog-, Edge- und Dew-Computing zwei weitere Architekturen zur Verarbeitung von großen Datenmengen aus einer Vielzahl von Datenquellen entwickelt. Die folgenden Unterabschnitte geben einen kurzen Überblick über diese drei Architekturparadigmen und deren verschiedene Varianten.

2.3.1 Cloud Computing

Die Verarbeitung und Speicherung von persönlichen und betriebsinternen Daten erfolgen in der heutigen Zeit nicht mehr auf dem eigenen Rechner, sondern wird zunehmend ausgelagert. Das Cloud Computing und die angrenzenden Technologien spielen für diese Art der modernen Informationsverarbeitung eine zentrale Rolle. Für die weiteren Betrachtungen gehen wir von der folgenden, allgemeinen Definition von *Cloud Computing* aus:

> **Zitat:** Definition Cloud Computing nach Alex Didier Essoh [Ess09]
>
> Von Cloud Computing wird dann gesprochen, wenn eine oder mehrere der folgenden drei IT-Dienstleistungen Infrastruktur (Rechenleistung, Hintergrundspeicher, etc.), Plattform und Anwendungssoftware aufeinander abgestimmt, schnell und dem tatsächlichen Bedarf angepasst sowie nach tatsächlicher Nutzung abrechenbar über ein Netz bereitgestellt werden.

Die Abrechnung der Nutzung wird dabei unterschiedlich gehandhabt: Es wird, je nach Service und Anbieter, nach der benötigten Rechenzeit oder nach dem gespeicherten bzw. verarbeiteten Datenvolumen abgerechnet. Ausgehend von der obigen Definition wird in der Regel zwischen drei Betriebsmodellen unterschieden:

1. **Private Cloud:** Die verfügbaren Dienste und die Speicherung der Daten werden innerhalb des eigenen Netzwerkes realisiert. Der Anbieter und der Anwender der Cloud sind somit identisch. Ein Beispiel für eine Private Cloud ist das Softwareprodukt Owncloud[10].
2. **Public Cloud:** Diese Variante ist eine von, meist kommerziellen Anbietern, angebotene Lösung, die auf eine hohe Zahl von Anwendern ausgelegt ist. Beispiele für Public Clouds sind Amazons EC2[11] und IBMs Cloud[12].

[10] `https://owncloud.org/`, zuletzt aufgerufen am 05.01.2022
[11] `https://aws.amazon.com/de/ec2/`, zuletzt aufgerufen am 05.01.2022
[12] `https://www.ibm.com/cloud`, zuletzt aufgerufen am 05.01.2022; ehemals IBM Bluemix

3. **Hybrid Cloud:** Zum Erreichen einer besseren Lastbalanzierung erfolgt bei der hybriden Cloudlösung eine Mischung der Anteile von Private und Public Cloud. Die Speicherung der Rohdaten und deren Vorverarbeitung erfolgt in der Private Cloud. Die Public Cloud dient zur Berechnung des endgültigen Ergebnisses und ggf. zur dessen Präsentation. Anbieter von hybriden Cloudlösungen ist beispielsweise Hewlett Packard Enterprises[13].

Bezüglich der angebotenen Leistungen wird ebenfalls zwischen drei Typen unterschieden:

1. **Infrastructure as a Service (IaaS):** Die Nutzer verwenden bereitgestellte virtualisierte Maschinen, um eigene Betriebssysteme und Programme zu installieren und zu verwenden. Zudem erfolgt die Speicherung der Daten auf dem entfernten Rechner.
2. **Platform as a Service (PaaS):** Zusätzlich zu IaaS wird das Betriebssystem vom Cloud-Anbieter bereitgestellt. Auf diesem werden die eigenen Anwendungen installiert und verwendet.
3. **Software as a Service (SaaS):** Der Nutzer verwendet, meist über einen Browser oder eine App, die vom Dienstleister bereitgestellten Anwendungen und Dienste.

Je mehr an den Dienstleistern von Cloudsystemen ausgelagert wird, desto weniger Kontrolle hat der Anwender über die Ressourcen. Während bei der eigenen Datenverarbeitung der Nutzer die volle Kontrolle über die Anwendung, das Betriebssystem, den Server und den verfügbaren Speicher hat, verliert er bei IaaS und PaaS die vollständige Kontrolle über die physischen Ressourcen (Speicher und Server) sowie die teilweise Kontrolle über das Betriebssystem und die Anwendungen. Bei SaaS hat der Anbieter die volle Kontrolle über sämtliche Komponenten. Der Nutzer ist somit darauf angewiesen, dass der Anbieter vertrauenswürdig mit sämtlichen hinterlegten Daten umgeht und ein hohes Datenschutzniveau garantieren kann.

Wie bei jeder Art von Datenverarbeitung und -speicherung ist Cloud Computing nur datenschutzrelevant, wenn personenbezogene Daten verarbeitet bzw. gespeichert werden. Werden personenbezogene Daten verarbeitet, so sollte der Einsatz einer hybriden Cloudlösung bevorzugt werden [BDH+13]: In der privaten Cloud werden die Daten vorverarbeitet und der Personenbezug entfernt, beispielsweise durch Anonymisierung, Pseudonymisierung oder dem Löschen der betroffenen Datensätzen. Die Public Cloud verarbeitet anschließend die Daten weiter und liefert das Ergebnis an die private Cloud zurück, die auch den Personenbezug wiederherstellt. Dieses Vorgehen ermöglicht die datenschutzkonformste Verarbeitung der Daten, wenn Daten mit Drittanbietern geteilt werden müssen.

Die Verarbeitung personenbezogener Daten ist abseits der Private Cloud datenschutzrechtlich schwierig bis kaum lösbar, da meist eine grenzüberschreitende Datenverarbeitung vorliegt. Falls die datenverarbeitende Stelle außerhalb der Europäischen Union bzw. des Europäischen Wirtschaftsraumes (siehe dazu [Deu15, §3 Absatz 8] der alten Fassung des Bundesdatenschutzgesetzes) liegt, ist die Auftragsdatenverarbeitung durch den Cloud-Anbieter nicht zulässig, außer das Land, in dem das Unternehmen registriert ist, bietet ein vergleichbares, angemessenes Datenschutzniveau (siehe dazu [Deu15, §§ 4b, 4c] der alten Fassung des Bundesdatenschutzgesetzes). Anwender sollten vorher die Datenschutzrichtlinien der Anbieter genau studieren, da viele Systeme Abkommen wie das ehemalige Safe Harbor [Mar06] oder das neuere – wenngleich auch nicht mehr gültige – Privacy Shield [WA16], welche den Datenschutz zwischen der EU und den USA regeln, nicht oder nur teilweise einhalten.

2.3.2 Fog Computing

Im Gegensatz zum Cloud Computing erfolgt beim *Fog Computing* die Auswertung der Daten auf lokaler Ebene. Dabei erfolgt die Datensammlung- und Verarbeitung in einem lokal begrenzten Areal bzw. als Teil eines Intranets. Für die weiteren Betrachtung gehen wir von folgender Definition für das Fog Computing aus:

[13] `https://www.hpe.com/de/de/solutions/cloud.html`, zuletzt aufgerufen am 05.01.2022

Definition: Fog Computing nach [Ope17a]

Fog Computing bezeichnet eine horizontal verteilte Architektur, bei der die Verarbeitung, die Speicherung, die Kontrolle und die Bereitstellung von Netzwerkfunktionen näher an den Nutzer entlang der Verarbeitungskette zwischen Cloud und Endgerät verlagert wird.

Die Vorteile des Fog Computing lassen sich nach [Ope17a] in den folgenden fünf Punkten zusammenfassen:

1. Sicherheit: Zusätzliche Sicherheitsmaßnahmen für die Umsetzung vertrauenswürdiger Transaktionen
2. Kognition: Bewusstsein für kundenorientierte Ziele, um Autonomie zu ermöglichen
3. Agilität: Schnelle Innovation und kostengünstige Skalierung unter einer gemeinsamen Infrastruktur
4. Latenz: Echtzeitverarbeitung und Kontrolle der Systeme
5. Effizienz: Dynamisches Pooling der lokalen, unbenutzten Ressourcen auf teilnehmenden Endgeräten

Als Datenbasis werden häufig Messwerte von Sensoren erhoben, beispielsweise zur Bestimmung freier Parkplätze in Großstädten [SBF+16]. Sensoren bieten ein breites Einsatzspektrum, wie die Gebäudeüberwachung und das Messen der lokalen Feinstaubbelastung[14].

Der Einsatz lokaler Sensorsysteme biete eine Vielzahl von Vorteilen: Steht einem Hochleistungsrechner ein leistungsschwacher Rechner mit der Hälfte der Taktrate gegenüber, so benötigt der langsamere Rechner zwar doppelt so lange zur Berechnung des Ergebnisses, benötigt aber nur die Hälfte der Energie. Annahme hierfür ist allerdings, dass auch die anliegende Spannung auf die Hälfte reduziert wurde [Tan09].

Geschickt konfiguriert, beträgt die Batterielaufzeit der Sensorsysteme mehrere Monate. Der größte Energiebedarf besteht im Senden und Empfangen von Nachrichten durch drahtlose Kommunikationstechnologien. Die Reduzierung des Nachrichtenaustausches und die damit verbundene Senkung des Stromverbrauches stellt einen großen Forschungsschwerpunkt dar. Der folgende Teilabschnitt motiviert die damit verbundenen Fragestellungen kurz.

Dust Computing

Als Spezialfall des Fog Computings kann das Dust Computing [WLLP01] angesehen werden. Smart Dust bezeichnet eine Klasse von Kleinstsystemen, welche autonom Daten sammeln, verarbeiten und weiterleiten. Die Systeme sind meist nur wenige Millimeter groß und eignen sich somit zum Aufbau eines großen Sensornetzwerkes.

Das Übertragen von Daten kostet einen Knoten mehr Energie als deren Verarbeitung. Werden einfache Operationen, wie das Filtern, Aggregieren oder Klassifizieren lokal durchgeführt, wird die Menge an übertragenen Daten signifikant reduziert. Als positive Folgeerscheinung ergibt sich eine längere, unabhängige Energieversorgung durch den gesunkenen Energieverbrauch.

Die neuesten Smart-Dust-Geräte sind lediglich 63 mm^3 groß und verfügen über eine integrierte Solarzelle zur eigenständigen Stromversorgung. Das in [WLLP01] vorgestellte Gerät verbraucht lediglich ca. 0,1 Nanojoule pro Bit. Werden jeden Tag 1 Millijoule durch direkte oder indirekte Sonneneinstrahlung an Energie aufgenommen, so kann das Geräte im Sekundentakt Daten erzeugen, verarbeiten und ggf. übertragen.

Die OpenFog-Referenzarchitektur

Das OpenFog Consortium, welchem u. a. verschiedene Universitäten und bekannte Firmen, wie Microsoft, Intel und Toshiba, angehören, entwickelt gegenwärtig eine Referenzarchitektur [Ope17a, Ope17b] zur Standardisierung von Fog-basierten Systemen. Die Architektur basiert auf den folgenden acht Säulen, die von jedem System eingehalten werden sollen:

[14] Siehe dazu beispielsweise das Citizen-Science-Projekt `https://luftdaten.info`, zuletzt aufgerufen am 05.01.2022.

1. Säule der Sicherheit: Vertrauen, Nachvollziehbarkeit und Privatsphäre
2. Säule der Skalierbarkeit: Lokale Kontrolle, Verarbeitung und Orchestration
3. Säule der Offenheit: Zugänglichkeit und Kontrolle der Ressourcen, *White box*-Entscheidungsfindung, Interoperabilität und Datennormierung
4. Säule der Autonomie: Flexibilität, Agilität, sowie der Wert der Daten
5. RAS-Säule: Zuverlässigkeit (engl.: Reliability), Verfügbarkeit (engl.: Availability), Wartungsfreundlichkeit (engl.: Serviceability)
6. Säule der Agilität: Taktische und strategische Entscheidungsfindung
7. Säule der Hierarchie: Vollständige Cloud-Befähigung, Autonomie auf allen Ebenen
8. Säule der Programmierbarkeit: Programmierbare Software und Hardware, Virtualisierung

Kernanliegen der Referenzarchitektur ist die Vermeidung kostspieliger, bidirektionaler Datenübertragungen zwischen Sensoren und cloud-basierten Systemen. Durch die lokale Verarbeitung ergeben sich in Anlehnung an [Ope17b] mehrere Vorteile in den Punkten Sicherheit, Latenz und Kosteneffizienz, da die Menge der übertragenen Daten beispielsweise auf gelegentliche Statusmeldungen oder auf zusammengefasste, aggregierte Daten reduziert werden kann.

Analog zur Cloud stellt die OpenFog-Referenzarchitektur verschiedene Betriebsmodi bereit, damit unterschiedliche Anforderungen bei der Entwicklung von Informationssystemen unter Einbezug der Fog abgedeckt werden. Unter dem Schlagwort *Fog as a Service* werden die Varianten *Infrastructure as a Service* (IaaS), *Platform as a Service* (PaaS) und *Software as a Service* (SaaS) zusammengefasst. Da Fog-basierte Systeme meist aus mehreren Ebenen zusammengesetzt werden, ermöglicht die Referenzarchitektur Kombinationen aus IaaS, PaaS und SaaS, um den Anforderungen an die jeweils vorliegende Infrastruktur gerecht zu werden.

Im folgenden Abschnitt wird mit *Berkeley DB* ein eingebettetes System zur Speicherung und Verarbeitung großer Datenmengen vorgestellt. Durch die Flexibilität und Skalierbarkeit des Systems bei gleichzeitiger puristischer Handhabung der zur Verfügung stehenden Ressourcen eignet sich Berkeley DB als *Fog as a Service* in den verschiedenen Ausprägungen.

Berkeley DB

Oracles Berkeley DB [Ora17] ist eine hoch hochperformante, eingebettete Datenbank, welche sowohl relationale Strukturen, als auch Java-Objekte und einfache Schlüssel-Wert-Paare abspeichern kann. Trotz der geringen Installationsgröße bietet Berkeley DB einen relativ hohen Funktionsumfang, auch wenn dieser nicht ganz an vollwertige DBMS heranreicht. Zu den unterstützten Funktionen zählen beispielsweise die Replikation und Partitionierung, die Bereitstellung verschiedener Indexstrukturen und die Unterstützung von Transaktionen [Ora18]. Der Zugriff via SQL erfolgt über bekannte Schnittstellen wie ODBC und JDBC. Zusätzlich werden auch imperative Programmiersprachen, wie C, C++ und Java, unterstützt.

Die Entwickler von BerkeleyDB sehen das System dabei nicht nur als reines System für die lokaler Speicherung, sondern vielmehr als flexibles und skalierbares System, welches auf allen Rechenknoten entlang der *Edge*, der Verarbeitungskette zwischen lokalen Systemen und der Cloud, genutzt werden kann [Ora18].

Anwendung von Fog Computing zur Realisierung von Privatsphäre

In [KSH14] wird ein Framework zur Sicherstellung der Privatsphäre in Sensor-Fog-Netzwerken vorgestellt. Die Verarbeitung der Daten wird von den Daten-erzeugenden Geräten teilweise mit übernommen. Dazu zählen neben der Erhebung und Übertragung der Daten die Extraktion der für die Verarbeitung notwendigen Features, die Transportverschlüsselung und Partitionierung der Daten. Auf der Fog-Ebene werden nach der Entschlüsselung die Daten weiterverarbeitet, beispielsweise durch Klassifikationsalgorithmen.

2.3.3 Edge Computing

Durch das Internet der Dinge wird die Verarbeitung der erzeugten Daten zunehmend von der Cloud auf das *Edge* des Netzwerks [SD16] verlagert. Der Begriff *Edge* kennzeichnet dabei diejenigen Verarbeitungseinheiten, welche zuvor nur für die Kommunikation zwischen den erzeugenden Datenquellen und den Cloud-Servern zuständig waren. Durch die gesteigerte Leistungsfähigkeit dieser Geräte kann das Problem der limitierten Bandbreite gelöst werden, welches als Flaschenhals des Cloud Computings angesehen wird [SD16].

> **Definition:** Edge Computing nach [SD16]
>
> Edge computing [...] refers to the enabling technologies that allow computation to be performed at the network edge, on both downstream data on behalf of cloud services and upstream data on behalf of IoT services. We define an edge device as any computing or networking resource residing between data sources and cloud-based datacenters.

Edge Computing wird als einer der nächsten großen Hypes im IT-Bereich gesehen. Nach dem Forschungs- und Beratungsunternehmen Gartner wird bis 2022 ein Großteil der Daten aus der Cloud ausgelagert:

> **Zitat:** Gartner zum Thema Edge Computing[a]
>
> Around 10% of enterprise-generated data is created and processed outside a traditional centralized data center or cloud. By 2022, Gartner predicts this figure will reach 75%.

[a] `https://www.gartner.com/smarterwithgartner/what-edge-computing-means-for-infrastructure-and-operations-leaders/`, zuletzt aufgerufen am 05.01.2022

Zeitkritische Anwendungen, wie die Echtzeitverarbeitung von Sensordaten aus autonomen Fahrzeugen, werden laut Gartner vor Ort bzw. entlang der Edge verarbeitet und gepuffert. Die Analyse großer Datenmengen wird weiterhin in der Cloud realisiert – jedoch werden nicht mehr alle Daten zentral gespeichert. Neben der reduzierten Kommunikationslast entstehen eine Reihe weiterer Vorteile – nicht zuletzt ein erhöhtes Datenschutzniveau.

Nach [SD16] besteht jedoch in der Programmierbarkeit von Edge-Anwendungen und -Diensten ein offenes Problem im Raum. Programmierer müssen ihre entwickelten Funktionen so abändern, dass einzelne Teile zwischen Fog und Cloud über die Edge verteilt werden. Gegenwärtig wir nach [SD16] die Verteilung händisch vorgenommen. Dies ist einerseits fehleranfällig, andererseits ist dieser Ansatz weder skalierbar noch flexibel, da nicht auf eine dynamische Konfiguration eines Edge-Netzwerkes reagiert werden kann.

Zur automatisierten Zerlegung von Analysen wird in Kapitel 6 ein neu entwickeltes Verfahren vorgestellt. Komplexere Analysen, die in SQL formuliert wurden, werden analysiert und auf Basis der unterstützten Anfrageoperatoren der einzelnen Rechenknoten im Edge-Netzwerk verteilt. Neben der Verteilung wird der Fokus auf die datensparsame Verarbeitung der erfassten Daten gelegt. Ausgehend von der vorliegenden Infrastruktur wird bestimmt, wie weit die Daten vorgefiltert bzw. aggregiert werden können, ohne dass Informationen verloren gehen.

Die Idee, Daten entlang eines Netzwerkes zu berechnen, ist nicht neu. Das Fallbeispiel in Anhang A.5 zeigt anhand von TinyDB ein System, welches Daten schrittweise innerhalb eines Netzwerkes aggregiert. Ansätze zur Realisierung von Speicherstrukturen und zur Umsetzung der Kommunikation zwischen den beteiligten Edge-Knoten werden in [GCN+19] diskutiert.

2.3.4 Dew Computing

Eine Alternative bzw. Ergänzung zum Cloud und Edge Computing bildet das *Dew Computing*. Dieses Konzept lässt sich von den bisherigen Ansätzen wie folgt abgrenzen:

> **Zitat:** Dew Computing nach Yingwei Wang [Wan16]
>
> Dew computing is an on-premises computer software-hardware organization paradigm in the cloud computing environment where the on-premises computer provides functionality that is independent of cloud

> services and is also collaborative with cloud services. The goal of dew computing is to fully realize the potentials of on-premises computers and cloud services.

Die Unabhängigkeit (engl.: independence) spiegelt sich darin wieder, dass durch lokale Ressourcen in Form eines *On-Premise*-Rechners die Funktionalitäten von Cloud Services nachgebildet werden und ohne ständige Internetverbindung lauffähig sind. Durch den Austausch von Informationen zwischen Cloud- und Dew-Rechnern, beispielsweise durch Synchronisation, wird der Aspekt der Kollaboration umgesetzt.

Ähnlich zum Cloud Computing unterteilt Wang [Wan16] Dew Computing nach dessen Anwendungszweck und den dabei verwendeten Ressourcen in sieben Kategorien:

1. **Web in Dew:** Die On-Premise-Rechner enthalten (modifizierte) Teile des World Wide Webs.
2. **Storage in Dew:** Der On-Premise-Rechner synchronisiert seinen Speicher mit einer Cloud.
3. **Database in Dew:** Datenbanken werden als Backup eines Cloud-Services auf dem On-Premise-Rechner gehalten.
4. **Software in Dew:** Software wird gleichzeitig als Cloud Service als auch auch auf dem On-Premise-Rechner angeboten.
5. **Platform in Dew:** Auf dem On-Premise-Rechner wird eine Software-Suite angeboten, deren Einstellungen und Anwendungsdaten mit den verknüpften Cloud-Services synchronisiert werden.
6. **Infrastructure as Dew:** Der On-Premise-Rechner enthält a) entweder eine virtuelle Maschine, welche auch in der Cloud existiert und synchronisiert wird, oder b) zumindest alle Systemeinstellungen und Anwendungsdaten synchronisiert.
7. **Data in Dew:** Dies umfasst alle Anwendungen, die nicht in die vorherigen Kategorien eingeordnet werden können.

Dew Computing besitzt dementsprechend ein breites Anwendungsspektrum. Anwendung findet Dew Computing beispielsweise in der Auslagerung maschineller Lernverfahren für die Kontrolle von industriellen Prozessen [STE21] und bei der Umsetzung intelligenter Parkleitsysteme [Yu21].

2.3.5 Fazit

Bedauerlicherweise werden nicht in jedem Fall die Privatheitsansprüche der Nutzer von cloud-basierten Lösungen vollkommen bewahrt. Dies betrifft insbesondere die Angebote von international agierenden Firmen, welche nicht immer an das europäische oder deutsche Datenschutzrecht gebunden sind. Dadurch können auf den gesammelten personenbezogenen Daten auch ohne Wissen des Nutzers ungewollte Analysen durchgeführt werden oder es erfolgt eine unberechtigte Weitergabe der Daten an Dritte. Eine Klassifizierung dieser möglichen Bedrohungen und Angriffswege ist in [SMS21] aufgeführt. Nach [AAA+21] stellt bei der Realisierung von Lösungsansätzen die beschränkte Kapazität eine der wichtigsten offenen Herausforderungen im Edge Computing und Internet der Dinge dar.

Als Alternative zu den Serverfarmen in großen Rechenzentren kann das Internet der Dinge selbst zur (Vor-) Verarbeitung der eigenen erzeugten Datenmengen beitragen. Sensoren, Sender und weitere an der Kommunikation beteiligte Geräte können Teile von komplexen Anfragen und Auswertungen übernehmen und somit Daten vorverdichten und filtern. Dies führt zu einem ähnlichen Ansatz wie es in [BHX17] für die Optimierung der Latenz in Edge Computing-Netzwerken vorgeschlagen wird. Der in Kapitel 6 entwickelte Ansatz geht dabei noch einen Schritt weiter: An Stelle von zuvor ausgewählten, statischen Programmteilen und Teilanfragen wird zur Laufzeit von Analysen *dynamisch* entschieden, welche Teile der Anfrage auf den einzelnen Knoten ausgeführt werden können. Durch diese optimierte Auslastung der Geräte auf Basis der beschränkten Anfragekapazitäten wird zeitgleich das Ziel der Datensparsamkeit umgesetzt. Jeder Knoten besitzt nur noch diejenigen Daten, die zur weiteren Berechnung der ursprünglich gestellten Anfrage notwendig sind. Weitere, ungewollte Analysen sind dadurch nur noch schwer umsetzbar.

2.4 Datenschutz

Datenschutz spielt bei der Ausgestaltung von Informationssystemen eine zunehmend größere Rolle. In diesem Abschnitt werden zunächst allgemeine Begriffsbestimmungen eingeführt, um darauf aufbauend die wichtigsten Aspekte des Datenschutzes für diese Arbeit zu bestimmen. Die weitere Betrachtung der gesetzlichen Grundlagen und inwieweit Datenschutz durch Technikgestaltung realisiert werden kann, zeigt auf, wie die in dieser Arbeit entwickelten Konzepte zur vertrauenswürdigen Anfrageverarbeitung (siehe Kapitel 3) und zur verteilten Berechnung von Anfragen (siehe Kapitel 6) dazu beitragen, Datenschutz konsequent und ohne Einbußen hinsichtlich der Genauigkeit der Analysen zu realisieren. Eine weiterführende Übersicht zur Geschichte des Datenschutzes bzw. dessen gesetzliche und technische Grundlagen ist in [Poh18] zu finden. In dieser Arbeit gehen wir von der folgenden, allgemeinen Definition von Datenschutz aus:

Definition: Datenschutz nach Alan Westin [WR67]

Datenschutz ist das Recht des Betroffenen, einer Gruppe bzw. einer Institution, selbst darüber zu bestimmen, in welchem Ausmaß seine bzw. ihre personenbezogenen Informationen gespeichert und verarbeitet werden.[a]

[a] Originaltext: „Privacy is the claim of individuals, groups, or institutions to determine for themselves when, how and to what extent information about them is communicated to others." [WR67]

Bei dem Betroffenen kann es sich um eine natürliche Person, eine rechtliche Person, eine Firma oder eine Organisation handeln, über die personenbezogene Daten gespeichert werden. Der Begriff *personenbezogene Daten* umfasst dabei mehr als Namen und Anschriften von Personen. Die Identifikation einer Person kann vielmehr durch die Kombination mehrerer Einzelmerkmale erfolgen:

Definition: Personenbezogene Daten in Anlehnung an [Eur16]

Unter personenbezogene Daten werden diejenigen Informationen verstanden, die einer identifizierten oder identifizierbaren Person zugeordnet werden können oder zu deren Identifikation beitragen können. Zu den personenbezogenen Daten gehören Identifikationsnummern sowie Daten zur physischen und psychischen Verfassung und zum geistigen, ökonomischen, religiösen, kulturellen und sozialen Hintergrund.

Unter personenbezogenen Daten werden neben Stammdaten, wie Name, Vorname und die Wohnanschrift [BDH+13] auch Schul- und Studienleistungen [Bun83] und viele weitere Daten wie *die rassische oder ethnische Herkunft, politische Meinungen, religiöse oder philosophische Überzeugungen, Gewerkschaftszugehörigkeit, Gesundheit, Sexualleben, strafbare Handlungen oder Ordnungswidrigkeiten* [Deu15, § 35] angesehen.

Zu beachten ist, dass viele Daten erst dann einen Personenbezug aufweisen, wenn sie mit anderen personenbezogenen Daten verknüpft werden. Beispielsweise stellt die Religionszugehörigkeit eine sensible Information da, wird sie jedoch als einzelnes Datum, beispielsweise für eine statistische Auswertung, betrachtet, ist sie keine personenbezogene Information. Erst in Kombination mit dem Vor- und dem Nachnamen einer Person fällt die Information unter das Bundesdatenschutzgesetz.

Die Unterscheidung in Daten mit und ohne Personenbezug fällt nicht immer einfach. Häufig werden Verbindungen zwischen verschiedenen Datensätzen erst dann offensichtlich, wenn mehrere Informationsquellen zu einem gemeinsamen Datensatz integriert werden. Dies ist insbesondere der Fall, wenn verschiedene Web-Services, wie Geodatendienste, soziale Netzwerke und weitere Daten aus den Repositorien von Content-Management-Systemen miteinander verknüpft werden. Dabei sind nicht nur die Daten aus den einzelnen Quellen selbst zum Verknüpfen geeignet, sondern auch die dazugehörigen Metadaten, wie die IP-Adresse des Nutzers bzw. dessen Daten im Cache des Browsers.

Die Verarbeitung und Analyse von Daten beginnt mit deren Erfassung, beispielsweise mittels Sensoriksystemen oder beim Aufrufen von Webseiten durch das Anlegen von Cookies. Für die Methoden zur Speicherung und Auswertung der personenbezogenen Daten existieren vielfältige Begriffe:

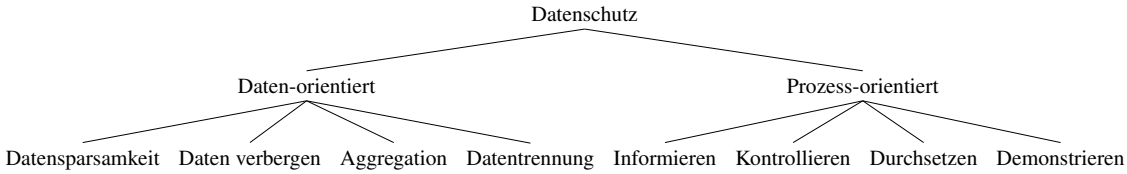


Abbildung 2.3: Unterteilung der Datenschutzaspekte in Daten- und Prozess-orientierte Forderungen nach [Hoe12].

Definition: Datenspeicherung und -verarbeitung in Anlehnung an [Deu18]

Zur Datenverarbeitung zählen alle (automatisierten) Handlungen, welche die Daten sammeln, speichern, organisieren, aufnehmen, adaptieren, verändern, übertragen, veröffentlichen, blockieren oder löschen. Dies betrifft nicht nur die Speicherung und Verarbeitung innerhalb eines Informationssystems, sondern auch die Papierform und die manuelle Verarbeitung.

Aus der Datenspeicherung und -verarbeitung personenbezogener Daten wurden aus Sicht von Datenschützern [Sch10] zwei allgemeine technische Forderungen erarbeitet: Datenschutz durch Technikgestaltung (*Privacy by Design*) und durch datenschutzfreundliche Voreinstellungen (*Privacy by Default*). Während *Privacy by Default* auf einen besseren Basisschutz, beispielsweise durch Zugriffskontrolle und Verschlüsselung, setzt, bezieht sich *Privacy by Default* auf die Integration von Datenschutzmechanismen in das Design und die Implementation von Informationssystemen.

Im folgenden Unterabschnitt werden detailliertere Aspekte des Datenschutzes vorgestellt. Diese Faktoren sollen Entwicklern als Referenz für eine datenschutzkonforme Ausgestaltung von Informationssystemen dienen. Anschließend werden ausgehend von den rechtlichen Aspekten in Unterabschnitt 2.4.2 konkretere technische und organisatorische Maßnahmen abgeleitet. Im Anhang A.6 können zusätzlich die gesetzlichen Grundlagen zur rechtlichen Umsetzung der oben eingeführten Datenschutzaspekte nach [Hoe12] nachgelesen werden.

2.4.1 Aspekte des Datenschutzes

Die Forderungen des Datenschutzes lassen sich – aus Sicht der Entwickler von Informationssystemen – in Daten- und Prozess-orientierte Aspekte unterteilen (vgl. Abbildung 2.3) [Hoe12]. Zu den Daten-orientierten Aspekten zählen Lösungsstrategien, die direkt mit den von dem Informationssystem gesammelten bzw. zu sammelnden Daten zusammenhängen. Die Prozess-orientierten Aspekte stellen nicht-funktionale Forderungen an das Informationssystem dar.

Datenorientierter Datenschutz

Unter datenorientiertem Datenschutz werden alle datenschutzfördernden Techniken zusammengefasst, deren Anwendung auf dem Datenbestand und dem Sammeln der Daten erfolgt. Diese umfassen die *Minimierung* der Sammlung, Speicherung und Verarbeitung von personenbezogenen Daten und das *Verstecken* von personenbezogenen Daten vor öffentlicher Einsicht. Zusätzlich wird die *getrennte Verarbeitung* personenbezogener Daten, sofern möglich, gefordert. Außerdem soll die *Aggregation* der Daten soweit erfolgen, dass die Daten nur noch die kleinstmögliche Menge an Details innehaben.

Minimierung der gespeicherten Daten: Nach [Hoe12] stellt Datenminimierung die grundlegendste Form des Datenschutzes dar – wo keine Daten erhoben werden, treten keine Datenschutzkonflikte auf. Um dies zu realisieren, sollen bereits bei der Erhebung nicht benötigte Daten verworfen werden. Es erfolgt dabei einerseits eine enge Kopplung an die Zweckbindung der Datenverarbeitung; zudem wird nach [VvdB18] vorgeschlagen, dass Datenminimierung auch durch entsprechende Voreinstellungen, Anonymisierung bzw. Pseudonymisierung und eine schnellstmögliche Löschung nicht mehr benötigter Daten realisiert werden kann.

Verstecken von Informationen: Personenbezogene Daten sollen nach Ansicht von [Hoe12] nicht frei zugänglich sein, insbesondere für Dritte, wie Werbeunternehmen. Dies trifft speziell auf Informationen zu, die eine direkte Identifikation einzelner Personen ermöglichen. Als Verfahren zum Verstecken von Informationen werden nach [Hoe12] die Verschlüsselung der gespeicherten Daten und des Netzwerkverkehrs sowie der Einsatz von Onion-Routing-Systemen, wie TOR[15], vorgeschlagen.

Aggregation von Daten: In den Basisrelationen von Datenbanken, bzw. in den Datenströmen von Sensordaten werden feingranulare Daten gespeichert. Diese Daten besitzen meist einen hohen Informationsgehalt. Dies können beispielsweise Detailinformationen über die Gewohnheiten von Einzelpersonen oder detaillierte, bis auf den Millimeter genaue Positionsangaben sein. Solche Merkmale sind häufig dazu geeignet, Personen eindeutig zu identifizieren.

Durch die Aggregation von Daten lassen sich Detailinformationen ausblenden. Eine Aggregationsfunktion erhält als Eingabe eine Menge von Werten, die i. d. R. auf eine kleinere Ergebnismenge, häufig einen einzelnen Wert, abgebildet wird. Ein Beispiel für eine Aggregationsfunktion ist die Abbildung von verschiedenen Gehältern der Mitarbeiter einer Firma auf das durchschnittliche Gehalt, gruppiert nach der dazugehörigen Abteilung. Dabei ist nichtsdestominder darauf zu achten, dass die einzelnen Personengruppen ausreichend groß sind, um keine Rückschlüsse auf Einzelpersonen ziehen zu können. Nach [Hoe12] können die Daten beispielsweise sowohl zeitlich als auch räumlich in unterschiedlichen Granularitätsstufen zusammengefasst werden.

Verteilte Speicherung und Verarbeitung von Informationen: Die verteilte, lokale Speicherung und Verarbeitung von Informationen soll nach [Hoe12] gegenüber der zentralen Auswertung bevorzugt werden. Sind die Daten über einzelne Personen lokal abgelegt, so besteht ein geringeres Risiko, dass ein vollständiges Personenprofil erstellt werden kann. Neben der verteilten Speicherung durch den Einsatz von lokalen Datenbanken empfehlen die Autoren zudem, dass auch die Informationen in den Datenbankrelationen weiter aufgespalten werden sollen. Dabei soll, sofern dies nicht dem Auswertungszweck widerstrebt, auch die inverse Abbildung, beispielsweise durch einen natürlichen Verbund, nur schwer realisierbar sein.

Prozessorientierter Datenschutz

Die Aspekte des prozessorientierten Datenschutzes umfassen nicht-funktionale Anforderungen, die eher rechtlicher als technischer Natur sind. Darunter fallen das angemessene *Informieren* des Nutzers bei der Erhebung und Verarbeitung von Daten, die ihn betreffen, die *Kontrolle*, wie personenbezogene Daten in einem Informationssystem verarbeitet werden, das *Durchsetzen* von Datenschutzrichtlinien, die vom Gesetzgeber oder anderen Anforderungen vorgegeben sind und das *Demonstrieren*, dass das Informationssystem mit Datenschutzrichtlinien und gesetzlichen Vorgaben übereinstimmt.

Die im Folgenden aufgeführten Aspekte lassen sich beispielsweise durch den Einsatz von Provenance-Techniken [AH18a] umsetzen. Die Datenerhebung bzw. -speicherung wird durch den Einsatz von (Data) Provenance nachvollziehbarer bzw. rekonstruierbarer, da Nutzer und Entwickler einen besseren Überblick darüber bekommen, wie ihre Daten verarbeitet wurden (*why*- und *how*-Provenance), bzw. welche Originaldaten für die Verarbeitung verwendet wurden (*where*-Provenance).

Informieren: Nutzer müssen in angemessener Weise informiert werden, wenn Informationen über sie verarbeitet werden. Dies führt zu einer Steigerung der Transparenz: Nur wenn der Nutzer darüber unterrichtet ist, welche Daten über ihn zu bestimmten Verwendungszwecken verarbeitet werden, kann er informierte Entscheidungen treffen. Die Unterrichtung betrifft insbesondere die Verarbeitung durch Dritte bzw. wenn neben dem eigentlichem Verarbeitungszweck weitere Dienste, wie personenbezogene Werbung, verwendet werden. Neben der Schaffung der Transparenz schlagen die Autoren in [Hoe12] die Platform for Privacy Preferences (P3P) vom W3C [W3C07] als Auskunftsplattform vor. Zusätzlich sollen Dienstbetreiber über Verstöße benachrichtigen, beispielsweise wenn personenbezogene Daten zweckentfremdet wurden.

[15] `https://www.torproject.org/`, zuletzt aufgerufen am 05.01.2022

Kontrollieren: Weiterhin muss durch die Nutzer kontrolliert werden können, wie die Verarbeitung personenbezogener Daten im Informationssystem erfolgt. Im Gegensatz zum Informieren greift der Nutzer aktiv ein: Es soll ihm gestattet sein, seine eigenen Daten zu betrachten und Änderungen vorzunehmen – oder sogar seine Daten vollständig zu löschen. Nach [Hoe12] stärkt dieser Aspekt die in den Datenschutzgesetzen [Eur16, Deu18] festgelegten Rechte der Betroffenen. Zudem kann der Nutzer so bessere Entscheidungen darüber treffen, ob er ein bestimmtes System nutzen wird. Beispiele für Kontrolle sind beispielsweise die Freigabeeinstellungen in sozialen Netzwerken, der Do-Not-Track-Modus in Webbrowsern oder die Abwahl von Cookies auf Webseiten.

Durchsetzen: Die Datenschutzrichtlinien sollen bzw. müssen gemäß des Aspektes *Durchsetzen* so umgesetzt werden, dass sie mit den relevanten gesetzlichen Aspekten (LDSG, BDSG, Strafgesetzbuch, ggf. Spezialgesetze) kompatibel sind. In [Hoe12] wird empfohlen, die Datenschutzrichtlinien regelmäßig zu überprüfen und ggf. an die geltende Rechtsprechung anzupassen. Eine Art digitales Rechtemanagement (engl.: Digital Rights Management, DRM), beispielsweise in Form von Zugangskontrolllisten, für Privatheitsansprüche soll dabei den Zugang zu den Daten regeln.

Demonstrieren: Unter dem Aspekt *Demonstrieren* werden nach [Hoe12] alle Techniken aufgeführt, die zeigen, dass die Umsetzung mit den ausgehandelten Datenschutzrichtlinien und gesetzlichen Anforderungen konform ist. Im Gegensatz zum *Durchsetzen* wird durch diesen Aspekt explizit bewiesen, dass das System den Ansprüchen konform agiert. Bei Zuwiderhandlungen soll zudem das Ausmaß an Datenschutzverstößen nachvollziehbarer gestaltet werden. Als mögliche Umsetzungen werden in [Hoe12] Logging- und Auditing-Maßnahmen vorgeschlagen.

Alternative Ansätze

Ähnliche Aspekte hinsichtlich des Datenschutzes lassen sich u. a. auch in [Org13], [Biz07] und [Han11] finden, wenngleich dies dort aus einer tendenziell rechtlichen Blickrichtung erfolgt. In den beiden letztgenannten Artikeln werden sieben goldene Grundregeln genannt, welche die Grundlagen des Datenschutzes auf das Wesentliche reduzieren. Dazu gehören:

1. **Rechtmäßigkeit:** Die Verarbeitung personenbezogener Daten erfolgt nur auf Basis einer gesetzlichen Grundlage.
2. **Einwilligung:** Sie erfolgt freiwillig, informiert, widerrufbar und i. d. R. schriftlich.
3. **Zweckbindung:** Erhobene Daten dürfen nur zum vereinbarten Zweck verwendet werden.
4. **Erforderlichkeit:** Es gilt, den Aspekt der Datensparsamkeit einzuhalten und die Daten sollen nach der Erfüllung des Verwendungszweckes gelöscht werden.
5. **Transparenz:** Dies beinhaltet die Unterrichtung des Betroffenen darüber, welche Daten zu welchem Zweck über ihn erhoben werden.
6. **Datensicherheit:** Die Sicherheit der Daten bei der Erhebung, der Verarbeitung und Nutzung ist durch technische und organisatorische Maßnahmen, beispielsweise aus dem Grundschutzkatalog des BSI [Mün16], zu gewährleisten.
7. **Kontrolle:** Betriebliche Datenschutzbeauftragte und staatliche Aufsichtsbehörden sollen über die Einhaltung der Datenschutzregelungen wachen.

Ausgehend von diesen Grundsätzen werden in [Han11] häufige Fehler beim Design von Informationssystemen angeführt. Dazu zählen, aus technischer Sicht, die Speicherung der Daten für evtl. weitere, noch unbekannte Verwendungszwecke (*Storage as Default*, *Function Creep as Feature*) und der Gebrauch von künstlichen Schlüssel zur Identifikation (*Linkability by Default*), aber auch unscharfe Formulierungen in den Datenschutzbestimmungen der Unternehmen (*Fuzzy or Incomplete Information as Default*), um erstere Aspekte zu rechtfertigen.

Handlungsempfehlungen des Bundesamtes für Sicherheit in der Informationstechnik: Das Bundesamt für Sicherheit in der Informationstechnik (BSI) schlägt in seinem IT-Grundschutz-Katalog [Mün16] verschiedene technische und organisatorische Maßnahmen zur Umsetzung des Datenschutzes in Behörden und Unternehmen vor. Diese werden als übergreifender Baustein für weitere Sicherheitsmaßnahmen zusammengefasst.

Das BSI listet insgesamt 13 organisatorische Mängel auf, die bei der Umsetzung von Informationssystemen auftreten können. Dazu gehören u. a. die fehlende Zulässigkeit und Nichteinhaltung der Zweckbindung, eine unzureichende Datenvermeidung bzw. Datensparsamkeit bei der Datenerhebung sowie fehlende Transparenz und Kontrolle bei der Verarbeitung personenbezogener Daten.

Bei der Planung und Konzeption eines Informationssystems schlägt das BSI mehrere Handlungsempfehlungen vor. Dies beinhaltet den Aufbau eines Systems zum Datenschutzmanagement und die Festlegung von technisch-organisatorischen Maßnahmen basierend auf dem aktuellen Stand der Technik bei der Verarbeitung personenbezogener Daten. Bei der konkreten Umsetzung sind folgende Aspekte zu beachten:

- Verpflichtung/Unterrichtung der Mitarbeiter bei der Verarbeitung personenbezogener Daten,
- Organisatorische Verfahren zur Sicherstellung der Rechte der Betroffenen bei der Verarbeitung personenbezogener Daten,
- Führung von Verfahrensverzeichnissen und Erfüllung der Meldepflichten bei der Verarbeitung personenbezogener Daten,
- Datenschutzrechtliche Freigabe,
- Meldung und Regelung von Abrufverfahren bei der Verarbeitung personenbezogener Daten,
- Regelung der Auftragsdatenverarbeitung bei der Verarbeitung personenbezogener Daten sowie
- Regelung der Verknüpfung und Verwendung von Daten bei der Verarbeitung personenbezogener Daten.

Für den anschließenden Betrieb empfiehlt das BSI zudem die Protokollierung und Dokumentation des Systems sowie eine datenschutzgerechte Löschung bzw. Vernichtung der Daten nach Erfüllung des Verwendungszweckes.

Datenschutzansprüche in smarten Umgebungen: Bünnig und Cap untersuchen in [BC07], inwieweit durch bestimmte Aspekte die Vorbehalte in smarten Umgebungen bzgl. des Datenschutzes verringert werden können. Zu den Kernforderungen zählen:

- **Bewusstsein**: Der Nutzer muss darüber informiert sein, welche Daten über ihn (automatisiert) gesammelt werden.
- **Kontrolle**: Ferner muss der Nutzer in der Lage sein, zu bestimmen, wie die Daten genutzt werden dürfen.
- **Anonymisierung bzw. Pseudonymisierung** auf Anwendungs- als auch Kommunikationsebene.
- **Zugangskontrolle**: Geteilte Informationen des Nutzers mit der smarten Umgebung müssen mittels Zugangskontrolle und Verschlüsselung vor unbefugten Zugriff abgesichert werden.
- **Auditing**: Die smarte Umgebung muss über Mechanismen zur nachweislichen Sicherstellung der gesetzten Datenschutzrichtlinien verfügen.

2.4.2 Datenschutz durch Technikgestaltung und Voreinstellungen

Datenschutz durch Technikgestaltung (engl.: Privacy by Design) und Datenschutz durch Voreinstellungen (engl.: Privacy by Default) bezeichnen zwei Prinzipien für den Entwurf und den Betrieb von Informationssystemen. Der Technische Datenschutz soll bereits in der Designphase von komplexen Softwareprodukten integriert werden und wird somit als ein Schlüsselfaktor für die Qualität eines Produktes interpretiert bzw. verstanden. Die DSGVO formuliert, neben den gesetzlichen Grundlagen, auch Anforderungen an die technische Gestaltung von Informationssystemen:

Zitat: Artikel 23 DSGVO [Eur16]: Datenschutz durch Technik und datenschutzfreundliche Voreinstellungen

1. Der für die Verarbeitung Verantwortliche führt unter Berücksichtigung des Stands der Technik und der Implementierungskosten sowohl zum Zeitpunkt der Festlegung der Verarbeitungsmittel als auch zum Zeitpunkt der Verarbeitung technische und organisatorische Maßnahmen und Verfahren durch, durch die sichergestellt wird, dass die Verarbeitung den Anforderungen dieser Verordnung genügt und die Rechte der betroffenen Person gewahrt werden.

2. Der für die Verarbeitung Verantwortliche setzt Verfahren ein, die sicherstellen, dass grundsätzlich nur solche personenbezogenen Daten verarbeitet werden, die für die spezifischen Zwecke der Verarbeitung benötigt werden, und dass vor allem nicht mehr personenbezogene Daten zusammengetragen oder vorgehalten werden als für diese Zwecke unbedingt nötig ist und diese Daten auch nicht länger als für diese Zwecke unbedingt erforderlich gespeichert werden. Die Verfahren müssen insbesondere sicherstellen, dass personenbezogene Daten grundsätzlich nicht einer unbestimmten Zahl von natürlichen Personen zugänglich gemacht werden.

Im Erwägungsgrund 78 der DSGVO [Eur16] wird näher auf die Schaffung von technischen und organisatorischen Maßnahmen zur Umsetzung des Datenschutzes eingegangen. Bei der Entwicklung von Informationssystemen sollen die Grundsätze *Datenschutz durch Technikgestaltung* (Privacy by Design) und *Datenschutz durch datenschutzfreundliche Voreinstellungen* (Privacy by Default), basierend auf dem aktuellen Stand der Technik, berücksichtigt werden. Auf konkrete Techniken wird in dem Erwägungsgrund hingegen nicht eingegangen; es wird vielmehr auf die allgemeineren Ziele, wie Datensparsamkeit, Pseudonymisierung und Transparenz, verwiesen.

Technische und organisatorische Maßnahmen

In §21 des Landesdatenschutzgesetzes Mecklenburg-Vorpommerns [Lan02] werden sechs Forderungen an die technischen, als auch organisatorischen Maßnahmen zur Sicherstellung des Datenschutzes gestellt:

1. **Vertraulichkeit:** Daten dürfen nur Befugte zur Kenntnis nehmen. Dies schließt unter anderem unbefugtes Abhören und Einsichtnahme ein. Als Gegenmaßnahmen können Verschlüsselung sowie Zugangs- und Benutzerkontrollen angewandt werden.

2. **Integrität:** Daten müssen unversehrt, vollständig und aktuell sein. Es darf keine gezielte Verfälschung und Manipulation der Daten bzw. Verarbeitungsvorschriften erfolgen. Integritätsbedingungen und -kontrollen können dem entgegenwirken.

3. **Verfügbarkeit:** Daten müssen zeitgerecht zur Verfügung stehen und ordnungsgemäß verarbeitet werden. Dies wird meist als nichtfunktionale Anforderung an Softwaresysteme gestellt. Die technische Umsetzung beruht in vielen Fällen auf dem Einsatz von Clustern hochperformanter, moderner Rechner.

4. **Authentizität:** Daten müssen ihrem Ursprung zugeordnet werden können. Der Erzeuger der Daten darf nicht verschleiert werden bzw. darf die Urheberschaft nicht leugnen. Qualifizierte elektronische Signaturen und Signaturkarten, die den höchsten erreichbaren Sicherheitsstandards genügen, können zur Authentisierung eingesetzt werden.

5. **Revisionsfähigkeit:** Es muss feststellbar sein, wer wann welche Daten wie verarbeitet hat. Dies kann durch Provenance-Informationen (where- und how-Provenance) erreicht werden.

6. **Transparenz:** Die Verarbeitung der Daten muss vollständig und in akzeptabler Zeit nachvollzogen werden können. Dazu sollten, zumindest in einem groben Rahmen, die Verarbeitungsmechanismen der verarbeitenden Stellen offengelegt sein. Dies kann beispielsweise, wie bei der Revisionsfähigkeit, durch Provenance-Informationen erreicht werden.

In dieser Arbeit werden insbesondere die Punkte Vertraulichkeit und Integrität betrachtet. Weitere Datenschutzaspekte, wie das Recht auf Vergessen, werden in dieser Arbeit nicht behandelt. Speziell die Datenvermeidung

stellt eine hohe Gestaltungsanforderung an Informationssysteme dar. Datenschutzaspekte lassen sich in vielen Fällen nicht ohne Weiteres in bestehende Systeme integrieren, da häufig eine umfangreiche technische Veränderung am System notwendig wäre. Es wird vielmehr angestrebt, Datenvermeidung bei der Entwicklung von neuer Software durch datenschutzfreundliche Ausgestaltungen des Systems bzw. einzelner Algorithmen zu integrieren. Da in vielen Anwendungsfällen nicht auf personenbezogene Daten verzichtet werden kann, wird in [BDH+13] ein Stufenkonzept für die Datensparsamkeit vorgestellt:

> **Zitat:** Stufen der Datensparsamkeit nach [BDH+13]
>
> 1. Der Verzicht auf personenbezogene Daten,
> 2. die Verarbeitung möglichst weniger personenbezogener Daten,
> 3. die frühestmögliche Anonymisierung,
> 4. die frühestmögliche Pseudonymisierung und
> 5. die frühestmögliche Löschung der personenbezogenen Daten, auf deren Verarbeitung nicht verzichtet werden kann.

Für jeden Einsatzzweck muss überprüft werden, ob nach der oben angegebenen Reihenfolge auf personenbezogene Daten ganz oder teilweise verzichtet werden kann. Speziell in Datenbanksystemen können während der Anfrageverarbeitung weiterführende Konzepte, wie Sichten und Integritätsbedingungen, eingesetzt werden, um den Aspekt der Datensparsamkeit technisch – und insbesondere auch automatisiert – umzusetzen. Durch Sichten werden zweckgebunden nur die Daten präsentiert, die zuvor für den jeweiligen Einsatzzweck bzw. Nutzer als notwendig spezifiziert wurden. Integritätsbedingungen können, beispielsweise als Realisierungsvariante für Datenschutzprofile, genutzt werden, um diese Sichten bzw. die dahinter liegenden Basisrelationen weiter einzuschränken.

Neben der Datenvermeidung, Datensparsamkeit und Anonymisierung können auch kryptographische Verfahren als Schlüsseltechnologie für Datenschutz durch Technikgestaltung angesehen werden [BDH+13]. Durch Verschlüsselung und digitale Signaturen kann eine zweckgebundene Auswertung erfolgen und die Vertraulichkeit des Systems durch gegenseitige Authentisierung der an der Kommunikation bzw. am Datenaustausch Beteiligten erfolgen.

Datenschutz durch Voreinstellungen

Durch für den Nutzer datenschutzfreundliche Voreinstellungen soll sichergestellt werden, dass, zumindest in den meisten Fällen, die Datenverarbeitung derart erfolgt, dass die Interessen der Nutzer berücksichtigt werden, ohne dass diese gesondert Einstellungen treffen müssen. Die Voreinstellungen müssen dabei so gewählt werden, dass sie möglichst mit den Einstellungen übereinstimmen, die ein Nutzer bei einer Berechtigungsprüfung vergeben würde.

Als mögliche Umsetzungen für datenschutzfreundliche Voreinstellungen stehen verschiedene Mittel zur Verfügung: Betreiber von Informationssystemen können in Richtlinien festlegen – bzw. vom Nutzer festlegen lassen –, zu welchen Zwecken Daten verarbeitet werden dürfen. Die Anwendung liest anschließend die getroffenen Einstellungen aus und ändert die Verarbeitung der personenbezogenen Daten entsprechend ab. Nähere Details zu Datenschutzrichtlinien werden im nächsten Kapitel im Abschnitt 3.2 behandelt.

Als weitere Voreinstellung kann die standardmäßige Anonymisierung der Daten angesehen werden, sofern der Nutzer nicht durch ein Opt-In-Verfahren der Verarbeitung seiner personenbezogenen Daten zustimmt. Nach Voigt [VvdB18] wird empfohlen, die Datensätze, sofern möglich, zu anonymisieren, da auf anonymisierten Daten die DSGVO keine Anwendung findet. Voigt gibt allerdings zu bedenken, dass durch Kombination verschiedener, umfangreicher Datensätze eine Re-Identifikation vereinfacht wird. Er empfiehlt daher, die verwendeten Anonymisierungsverfahren nach dem aktuellen Stand der Technik auszurichten. Zudem ist zu beachten, dass eine Pseudonymisierung der Daten in vielen Fällen nicht ausreichend ist. Die DSGVO verweist in ihren Erwägungsgründen explizit darauf hin:

> **Zitat:** Erwägungsgrund 26 DSGVO [Eur16]: Keine Anwendung auf anonymisierte Daten
> Einer Pseudonymisierung unterzogene personenbezogene Daten, die durch Heranziehung zusätzlicher Informationen einer natürlichen Person zugeordnet werden könnten, sollten als Informationen über eine identifizierbare natürliche Person betrachtet werden.

Datensparsamkeit spielt, egal ob nach dem alten oder neuen Bundesdatenschutzgesetz, der DSGVO oder einer etwaigen Einführung der Datensouveränität (siehe Anhang A.7), eine wichtige Rolle bei der Umsetzung von datenschutzkonformen Informationssystemen. Der folgende Abschnitt stellt in Kurzform den aktuellen Stand der Technik in Bezug auf datenschutzgerechte Technologien vor. Im Fokus stehen dabei grundlegende Konzepte zur Umsetzung der Datensparsamkeit.

2.5 Verfahren und Systeme zur Wahrung des Datenschutzes

In diesem Abschnitt werden Maße und Verfahren zur Sicherstellung des Datenschutzes näher vorgestellt. Diese Technologien zur Verbesserung der Privatsphäre (engl.: Privacy Enhancing Technologies; PETs) sind recht breit gefächert: Sie reichen von einfachen organisatorischen Maßnahmen bis hin zum Eingriff in die verwendeten Data-Mining-Methoden. Um einen besseren Überblick über die Einordnung bestehender PETs zu erhalten, wird im nächsten Abschnitt eine mögliche Kategorisierung einer bestehenden Erhebung [DDK+15] des Stands der Techniken vorgestellt. Diese Einteilung dient zur Vorbereitung für ein Einordnung der in den Kapiteln 4 und 6 entwickelten neuen Technologien zur Verbesserung der Privatsphäre.

2.5.1 Einordnung von Datenschutztechnologien in die Informationsverarbeitung

Basierend auf den möglichen Designstrategien wurde in [DDK+15] von den Autoren weiterhin untersucht, wie sich diese in die Verarbeitungskette von Big Data einordnen und, im Hinblick auf die Einhaltung der DSGVO, umsetzen lassen. Für alle Phasen in dieser Verarbeitungskette wird für die *Durchsetzung* und das *Demonstrieren* der Datenschutzaspekte empfohlen, dass Möglichkeiten zur automatisierten Definition von Datenschutzrichtlinien geschaffen werden. Es sollen zudem Werkzeuge zum Einsatz kommen, die die Durchsetzung, die Verantwortlichkeit und die Einhaltung der Bestimmungen überwachen. Eine alternative Einordnung der Aspekte nach [Hoe12] im Kontext von Big Data ist in Abbildung 2.4 dargestellt.

Datenakquise

Bei der Datenerhebung bzw. -erfassung lässt sich die *Datenminimierung* zunächst dadurch realisieren, dass bereits im Vorfeld geklärt wird, welche Daten zu erheben sind. Es sollen nur diejenigen Daten erfasst werden, die für die Verarbeitung zwingend notwendig sind. Die *Aggregation* umfasst die lokale Anonymisierung und Vorverarbeitung der Daten auf der Ebene der Datenquelle. Zum *Verstecken* sollen datenschutzfördernde Werkzeuge eingesetzt werden, die ein mögliches Tracking verhindern, die Verschlüsselung der Daten ermöglichen und die Identität des Nutzers verschleiern können. Die Aspekte des *Informierens* und *Kontrollierens* sollen durch die Schaffung von mehr Transparenz und Eingriffsmöglichkeiten, wie z. B. durch Datenschutzprofile (siehe Abschnitt 3.2), ermöglicht werden.

Datenanalyse und -bereinigung

Während der Datenanalyse soll homomorphe Verschlüsselung angewendet werden, um auch auf verschlüsselten Daten bestimmte Auswertungen, wie Wertevergleiche, ausführen zu können. Die Berechnungen selbst sollen datenschutzkonform erfolgen, indem z. B. keine Zwischenergebnisse nach außen gegeben werden. Beide Techniken dienen dem Aspekt des *Versteckens*. Zwecks *Aggregation* wird vorgeschlagen, dass die Daten durch unterschiedliche Anonymisierungsverfahren, die auf Generalisierung [SS98] oder Differential Privacy [DR14] aufbauen, vor der Veröffentlichung anonymisiert werden.

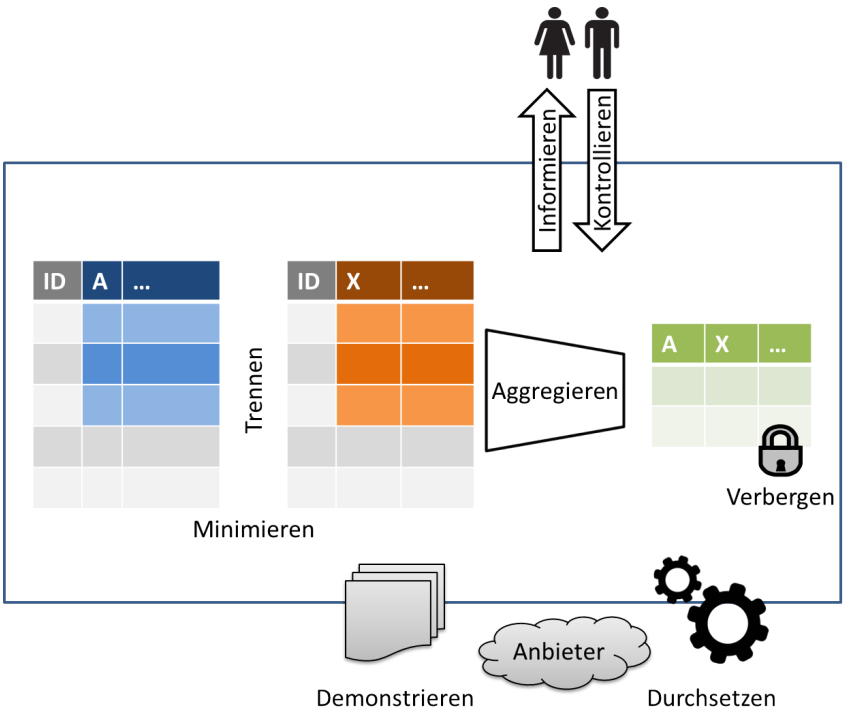


Abbildung 2.4: Einordnung der Datenschutzstrategien nach [Hoe12] in die Verarbeitungskette von Big-Data-Anwendungen (Abbildung nach [KHL13]).

Datenspeicherung

Für den Aspekt des *Separierens* sollen die Daten einerseits verteilt und dezentral gespeichert werden. Zudem wird vorgeschlagen, dass auch die Analyse verteilt erfolgen kann. Zum *Verstecken* der Daten werden Verschlüsselungs-, Authentifizierungs- und Zugriffskontrollverfahren (siehe Anhang A.2) empfohlen.

Datennutzung

Um die Daten datenschutzgerecht unter Einbehaltung des Aspekts des *Aggregierens* zu nutzen, muss eine Anonymisierung (siehe Unterabschnitt 3.4.1) der Daten erfolgen. Dabei muss jedoch einerseits darauf geachtet werden, dass die Qualität der Daten auf einem hohen Niveau erhalten bleibt. Andererseits muss trotz Aggregation und Datenminimierung auch die Data Provenance [AH18b] nachvollziehbar bleiben, um den Ursprung und die Authentizität der Daten nachzuvollziehen. Dies betrifft nicht nur die Daten selbst, sondern auch die damit verbundenen Metadaten.

Empfehlungen der Studie

Nach ausführlicher Analyse und Vorstellung verschiedener Datenschutztechniken sieht die Studie weiterhin Handlungsbedarf für die Umsetzung des gesetzlich festgeschriebenen Anspruchs auf Privatsphäre. Zum einen sollen Datenschutzbehörden sowie die datenerhebenden und -verarbeitenden Stellen zusammenarbeiten, um die Designstrategien in Prozessen und Werkzeugen umzusetzen:

> **Zitat:** Empfehlungen zu Privacy Design nach [DDK+15]
> Data Protection Authorities, data controllers and the big data analytics industry need to actively interact in order to define how privacy by design can be practically implemented (and demonstrated) in the area of big data analytics, including relevant support processes and tools.

Parallel sollen Forschung und Industrie dezentrale und datenschutzkonforme Analysemodelle entwickeln. Regierungsstellen sollen diese durch entsprechende Entschlüsse unterstützen und fördern:

> **Zitat:** Empfehlungen zu dezentraler Datenanalyse nach [DDK+15]
> The research community and the big data analytics industry need to continue and combine their efforts towards decentralised privacy preserving analytics models. The policy makers need to encourage and promote such efforts, both at research and at implementation levels.

Dabei ist stets eine Abwägung zwischen dem Nutzen des Systems und der Privatsphäre der betroffenen Personen zu treffen: Der Einsatz vieler datenschutzfördernder Technologien vermag die Privatsphäre der Nutzer zu schützen – wenn das System jedoch nicht oder nur sehr eingeschränkt seinen vorgesehenen Einsatzzweck erfüllen kann, so kann auch gleich auf dessen Einsatz verzichtet werden. Obwohl Datensparsamkeit als eine der Kerntechniken in Gesetzen, wie der Datenschutz-Grundverordnung [Eur16] und dem Bundesdatenschutzgesetz [Deu18] gefordert wird, wurde dieser Aspekt kaum umgesetzt. Petrlic und Sorge führen diese Entwicklung darauf zurück, „dass die Regelung (notwendigerweise) sehr unkonkret ist, [wodurch] sie in der Praxis aber kaum eine Rolle [spielt]“ [PS17].

Notwendigkeit und Datensparsamkeit

Das in dieser Arbeit entwickelte Konzept schließt genau diesen Zwischenraum: Anstatt Daten im Vorfeld zu anonymisieren oder zu löschen, wird gemäß der Zweckbindung ermittelt, welche Daten für die vom System bzw. Nutzer gewünschten Analysen zwingend notwendig sind, um weiterhin eine präzise Antwort zu generieren. Zu diesem Zweck wird die – in SQL – formulierte Anfrage analysiert und die Daten anschließend schrittweise vorverarbeitet.

Im Folgenden wird auszugsweise den Stand der Technik dargestellt. Dazu werden in Unterabschnitt 2.5.2 sowie in den Anhängen A.8 und A.9 einige konkrete Techniken vorgestellt, die bestehende Big-Data-Auswertungen

und -Analysemethoden um verschiedene Datenschutzverfahren erweitern. Im Anhang A.10 wird zusätzlich auf Frameworks zur Wahrung des Datenschutzes eingegangen.

2.5.2 Anonymisierungsverfahren

Datenschutz, im Speziellen die Aspekte der Datenvermeidung und Datensparsamkeit, wurden in der Vergangenheit mehr als rechtliches statt als technisches Problem angesehen [Lan01]. Der Einsatz von Datenschutztechniken fokussierte sich in der Vergangenheit auf verschiedene Anonymisierungsalgorithmen [Sam01, DR14, LLZM12], welche den Personenbezug aus den Datenbeständen entfernten. Dieser Unterabschnitt gibt einen kurzen Einblick über die verschiedenen Klassen von Anonymisierungsverfahren. Diese dienen als Motivation für den in Kapitel 4 vorgestellten Ansatz zur Erkennung von Quasi-Identifikatoren.

Generalisierung

Zu den bekanntesten Anonymisierungsverfahren zählt die Klasse der Generalisierungsverfahren, zu denen u. a. k-Anonymität [Sam01], l-Diversität [MKGV07] und t-closeness [LLV07] gehören. Diese Verfahren basieren auf zwei Prinzipien: Die namensgebende *Generalisierung* ermöglicht die Reduzierung der Präzision einzelner Werte. Durch das Weglassen von Nachkommastellen, das Abbilden numerischer Werte auf ein Werteintervall und die Zuordnung von Begriffen zu einem Oberbegriff wird die Zuordnung zwischen den Daten und der Identität einer Person erschwert. Dies kann jedoch auch zu einer ungenauen Auswertung führen, da verfälschte Werte in die Analyse einfließen. Diese Überführung wird mittels sogenannter Domänengeneralisierungshierarchien vorgenommen. Durch *Unterdrückung* werden einzelne Tupel aus der Ergebnisrelation entfernt. Dabei werden die fehlenden Daten häufig durch NULL-Werte oder die Zeichenkette $\star\star\star$ repräsentiert. Dies wird insbesondere in Fällen verwendet, in denen eine zu hohe Generalisierung der anderen Tupel notwendig wäre, um ein gewisses Maß an Datenschutz zu gewährleisten.

Domänengeneralisierungshierarchien: Unter Generalisierung werden Methoden zur Aggregation bzw. zur Verallgemeinerung großer Datenmengen auf eine höhere Ebene der Abstraktion verstanden. Die Generalisierung kann beispielsweise durch Clusteringverfahren realisiert werden, welches ähnliche Objekte auf neu gebildete Kategorien (hier: die generalisierten Werte) abbildet. Dabei werden ähnliche Objekte zu dem gleichen Wert generalisiert, während unähnliche Objekte zu unterschiedlichen Werten generalisiert werden. Die einzelnen Objekte sind in Domänengeneralisierungshierarchien (engl.: *domain generalization hierarchies*; DGHs) als Knoten eines gerichteten Baumes dargestellt. Die Stufe im Baum beschreibt den Verdichtungsgrad der Generalisierung: je näher der Knoten am Wurzelelement angeordnet ist, desto stärker generalisiert ist der Wert. Die Blätter des Baumes repräsentieren die ursprünglichen Werte der Objekte; die Wurzel hingegen den allgemeinsten Wert. Abbildung 2.5 zeigt einen Generalisierungsbaum am Beispiel des Attributes `Genre`.

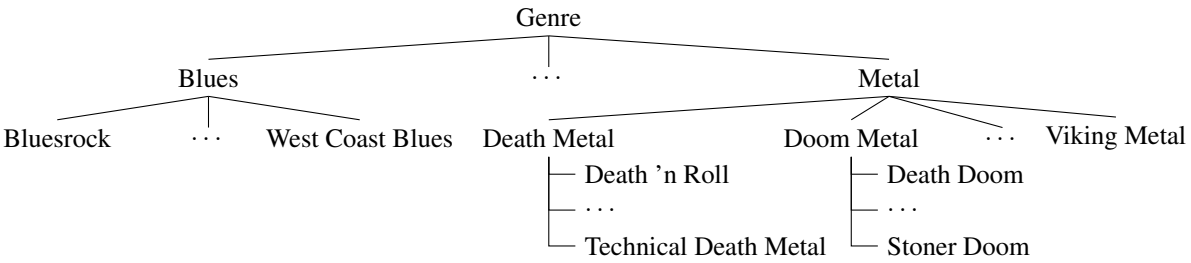


Abbildung 2.5: Generalisierungsbaum für die Domäne “Genre”.

Numerische Werte brauchen nicht explizit als Generalisierungsbaum dargestellt zu werden, weil die generalisierten Werte durch mathematische Funktionen, wie z. B. die ganzzahlige Division ohne Rest, berechnet werden

können. Für nicht-numerische Werte, wie z. B. Zeichenketten für Berufe, Religionen oder den Familienstand, sind Generalisierungsbäume hingegen unerlässlich, weil sinnvolle Zuordnungen von ähnlichen Attributwerten zu einem neu eingefügten, verallgemeinerten Wert häufig Expertenwissen voraussetzen.

Kategorische Attribute lassen sich zudem nur durch die Verwendung von Ähnlichkeitsmatrizen miteinander vergleichen. In Tabelle 2.1 ist ein Beispiel für eine Ähnlichkeitsmatrix für drei Attributwerte des Attributs `Genre` aus der `Tracks`-Relation gegeben. Existieren in einer Relation beispielsweise nur wenige Tupel mit dem Attributwerten `Hard Rock` und `Heavy Metal`, so könnten diese zu dem übergeordneten Genre `Metal` generalisiert werden.

	Pop	Hard Rock	Heavy Metal
Pop	1,0	0,2	0,1
Hard Rock	0,2	1,0	0,9
Heavy Metal	0,1	0,9	1,0

Tabelle 2.1: Beispiel für eine Ähnlichkeitsmatrix für kategorische Attributwerte.

Welche Attribute zur Generalisierung ausgewählt werden bzw. wie stark die Attributwerte generalisiert werden, hängt von dem verwendeten Anonymisierungsverfahren, dem geforderten Datenschutzniveau und dem konkreten Datenbestand ab. In den folgenden Abschnitten wird eine Auswahl dieser Verfahren kurz vorgestellt.

k-Anonymität: K-Anonymität [Sam01] beschreibt ein grundlegendes Maß zur Bestimmung der Anonymität von Personen innerhalb einer Sammlung von Daten. Eine Relation ist k-anonym, wenn, auf Basis der (quasi-) identifizierenden Attributmengen der Relation[16], ein Tupel von k-1 anderen Tupeln nicht unterscheidbar ist. Dadurch wird die Anonymität einzelner Personen gewährleistet, da die sensitiven Attribute nicht mehr eindeutig zu einer bestimmten Person zugeordnet werden können. Formal wird diese Forderung wie folgt definieren:

Definition: k-Anonymität nach [Sam01]
Sei $r(R)$ eine Relation über dem Schema $R = (A_1, \ldots, A_n)$ und einem Quasi-Identifikator $QI \subseteq R$. $r(R)$ gilt bzgl. einem gegebenen $k \in \mathbb{N}$ und QI als k-anonym, falls für jede Belegung $qi \in r(QI)$ die Bedingung $|\sigma_{QI=qi}(r(R))| \geq k$ gilt.

Erfüllt eine gegebene Relation R die Forderung nach k-Anonymität nicht, so muss diese durch Generalisierung und Unterdrückung anonymisiert werden. Während bei der Unterdrückung gezielt einzelne Tupel ausgeblendet werden, so werden bei der Generalisierung gezielt einzelne Attributwerte auf einen allgemeineren Wert, beispielsweise einen Oberbegriff, einen gerundeten Wert oder einen Wertebereich, abgebildet. Dies geschieht auf Basis der DGHs für die Attribute des bzw. der in R vorkommenden Quasi-Identifikatoren.

Wird nur Generalisierung verwendet, so wird für die anonymisierte Relation der Begriff *generalisierten Relation* verwendet. Nach [Sam01] ist diese wie folgt definiert:

Definition: Generalisierung nach [Sam01]
Seien $r(R)$ und $s(R)$ zwei Relationen über dem gleichem Relationenschema $R = (A_1, \ldots, A_n)$ und DGH_{A_i} die Domänengeneralisierungshierarchie für das Attribut A_i. $s(R)$ ist die generalisierte Relation von $r(R)$, wenn folgende Eigenschaften erfüllt sind:

1. $|r(R)| = |s(R)|$,
2. $\forall A_i \in R : dom(A_i, r(R)) \leq_D dom(A_i, s(R))$ und
3. $\forall t \in r(R) \exists t' \in s(R) \forall A_i \in R : t[A_i] \leq_{DGH_{A_i}} t'[A_i]$.

[16] Die formale Definition eines Quasi-Identifikators wird in Abschnitt 3.1 eingeführt.

Forderung (1) stellt sicher, dass keine Tupel unterdrückt werden. Durch (2) wird ausgedrückt, dass die Domänen dom der einzelnen Attribute von $r(R)$ identisch zu denen aus $s(R)$ sind bzw. in $s(R)$ generalisierte Werte enthalten können. Abschließend stellt Forderung (3) sicher, dass zwischen allen Attributwerten jeden Tupels aus $r(R)$ bzw. $s(R)$ eine bijektive Abbildung besteht, welche zu den Attributwerten $t[A_i]$ den gleichen oder einen generalisierten Wert in $t'[A_i]$ zuordnet. Für generalisierte Relationen mit Unterdrückung lässt sich die obige Definition in den Punkten (1) und (3) derart abändern, dass zu den Tupel aus $r(R)$ nicht zwangsweise korrespondierende Tupel in $s(R)$ existieren müssen.

Technische Umsetzung der k-Anonymität: Aus Datenbanksicht lässt sich die Unterdrückung von einzelnen Tupeln, welche die Anforderung an die k-Anonymität nicht erfüllen, in SQL durch die Kombination einer Aggregatfunktionen mit einer geschachtelten Anfrage realisieren. Dazu werden zunächst über einer Relation $r(R)$ für einen Quasi-Identifikator Q die Belegungen ermittelt, die weniger als *k* Mal auftauchen. Dies wird durch einen Filter in der `HAVING`-Klausel auf den gruppierten Werten realisiert. Anschließend werden von $r(R)$ die ermittelten Tupel abgezogen.

Beispiel: Ermittlung nicht-k-anonymer Äquivalenzklassen in SQL

Die folgende Anfrage verdeutlicht dies für die Relation `users` und den Quasi-Identifikator `(firstname, lastname)` (Vor- und Nachname) für $k = 5$.

```
1 SELECT *
2 FROM users
3 WHERE (firstname, lastname) NOT IN(
4   SELECT firstname, lastname
5   FROM users
6   GROUP BY firstname, lastname
7   HAVING COUNT(*) < 5
8 )
```

Alternativ können zunächst die Belegungen des Quasi-Identifikators bestimmt werden, welche die k-Anonymität erfüllen. Von $r(R)$ werden dann nur die ermittelten Tupel übernommen. Für mehrere Quasi-Identifikatoren müssen jeweils einzelne Selektionen auf die QIs erfolgen.

Durch die Negation der Bedingung erfolgt die Bestimmung der nicht-k-anonymen Tupel. Diese können anschließend generalisiert werden, um so den Informationsverlust im Vergleich zur Unterdrückung zu verhindern. Die in PArADISE (siehe Abschnitt 2.5.3) verwendeten Techniken zur Umsetzung der Generalisierung werden im Anhang B.2.1 näher ausgeführt.

Permutation

In [LLZM12] stellen Li et al. ein neues Anonymisierungskonzept vor, welches auf der Permutation von gruppierten Daten beruht. Im Gegensatz zu den bisher gängigen Anonymisierungsverfahren wie k-Anonymität [Swe02] verzichtet dieses Verfahren auf Generalisierungstechniken und behält die Originaldaten bei. Zudem werden mehr Korrelationen zwischen den sensitiven Attributen bewahrt, sofern diese keine (quasi-)identifizierenden Merkmale bilden. Das Slicing-Verfahren verspricht einen höheren Datennutzen, indem es die Verbindung von stark korrelierenden Attributen bewahrt. Auf dieser Grundlage ist es trotz Anonymisierung weiterhin möglich, Data-Mining- und Analyse-Techniken, wie beispielsweise Regressionsanalysen, in hochdimensionalen Datenbeständen anzuwenden.

Das Slicing-Konzept permutiert nicht den kompletten Datenbestand, sondern partitioniert die Daten sowohl horizontal als auch vertikal. Bei der horizontalen Partitionierung wird der Datenbestand nach festgelegten Filterkriterien (Selektion nach einem bestimmten Attribut, Anzahl an Partitionen, ...) in mehrere Mengen von Tupeln zerlegt. Die Attributpartitionierung der Attributmenge A in n nicht-leere Teilmengen wird so umgesetzt, dass jedes Attribut genau einer nicht-leeren Teilmenge A_i angehört.

Die vertikale Partitionierung unterteilt den Datenbestand spaltenweise, indem aus der vorhandenen Attributmenge mehrere kleine Attributmengen gebildet werden. Eine mögliche vertikale Zerlegung besteht im Aufteilen der Attribute eines Quasi-Identifikators [Dal86] auf mehrere Partitionen. Die Partitionierung der Tupel in b Buckets wird derart realisiert, dass jedes Tupel genau einem Bucket B_i zugehörig ist und jeder Bucket aus mindestens einem Tupel besteht.

Aus einem Datenbestand, der durch n horizontale und m vertikale Partitionsvorgaben zerteilt wird, entsteht ein Raster aus $n * m$ kleineren Relationen. Jede dieser Teilrelationen wird anschließend permutiert, indem die Reihenfolge der Tupel zufällig vertauscht oder sortiert wird. Die permutierten Relationen werden anschließend zu einer einzelnen Relation verknüpft.

Verbesserungen bzgl. des Informationsgehaltes können u. a. durch das Bilden von überlappenden Clustern und durch Überprüfen der l-Diversität-Eigenschaft der einzelnen Tupelpartitionen erreicht werden [DK14]. In [MSD06] wird zudem aufgezeigt, dass das Permutieren von Daten einen höheren Datennutzen und eine geringere Reidentifikationschance bietet als gezieltes Vertauschen von einzelnen Datensätzen.

Differential Privacy

Dwork beschreibt *Differential Privacy* als Versprechen der datenhaltenden Stelle gegenüber dem datengebenden Nutzer mit folgenden Worten:

> **Zitat:** The Algorithmic Foundations of Differential Privacy [DR14]
>
> You will not be affected, adversely or otherwise, by allowing your data to be used in any study or analysis, no matter what other studies, data sets, or information sources, are available.

Das Grundprinzip von Differential Privacy ist, durch absichtliches Verrauschen von Daten zu verhindern, dass bestimme Aussagen über eine Person getroffen werden können [DMNS06]. Differential Privacy nutzt dabei die Unschärfeeigenschaft aus. Informationen gelten als unscharf, wenn sie verschiedene Möglichkeiten für ihre Interpretation besitzen. So ist es möglich, dass eine Aussage, die auf diesen Informationen getroffen wird, wahr ist, aber auch die negierte Aussage zu einem gewissen Grad wahr ist. Unschärfe tritt in Informationssystemen immer dann auf, wenn unzureichend Daten vorliegen oder aufgrund von unpräzisen Sensoren verrauschte Daten erzeugt werden.

Im Falle von Differential Privacy wird Unschärfe explizit konstruiert. Dies kann durch das Entfernen von Tupeln aus einer Relation oder das Hinzufügen von Rauschen geschehen. Ähnlich wie bei der Generalisierung und Unterdrückung werden beim Rauschen einzelne Attributwerte verfälscht. Zugunsten einer erhöhten Komplexität können jedoch statistische Verfahren, wie Durchschnittsbildungen, mit nur geringem Genauigkeitsverlust ausgeführt werden. Im Gegensatz zu nichtspezifischen Aussagen (z. B. $23 \leq X \leq 42$), werden unscharfe Aussagen durch eine Unschärfefunktion charakterisiert, die angibt, wie wahrscheinlich das Auftreten eines Wertes ist.

Formale Definition von Differential Privacy: Formal wird Differential Privacy wie folgt definiert:

> **Definition:** Differential Privacy nach [DR14]
>
> Eine Zufallsfunktion f ermöglicht ϵ-Differential-Privacy für zwei Relationen $r(R)$ und $r'(R)$ über dem gleichen Relationenschema R, wenn
>
> - $r(R)$ und $r'(R)$ sich in höchstens einem Element unterscheiden und
> - für alle Teilmengen S des Wertebereichs dom von $f(r(R))$ bzw. $f(r'(R))$ (d. h., $S \subseteq dom(f(r(R)))$ bzw. $S \subseteq dom(f(r'(R)))$) die Gleichung $P[f(r(R)) \in S] \leq e^{\epsilon} \times P[f(r'(R)) \in S]$ gilt.
>
> Die Wahrscheinlichkeit P, dass die Funktion f einen konkreten Wert annimmt, darf sich zwischen den beiden Relationen $r(R)$ und $r'(R)$ nur minimal unterscheiden.

Der Grenzwert für den minimalen Unterschied wird durch das ϵ bestimmt. Je höher dieser Wert gewählt wird, desto geringer ist das zu erwartende Datenschutzniveau. Die Wahl eines geeigneten ϵ ist dabei nach Auffassung von Petrlic und Sorge [PS17, S. 39] anwendungsabhängig.

Ausgehend von der formalen Definition von Differential Privacy werden in [DR14] verschiedene Klassen von Algorithmen zur konkreten Umsetzung vorgeschlagen. Diese lassen sich zudem miteinander kombinieren, um aus einfacheren Algorithmen komplexere Anonymisierungsfunktionen zu schaffen.

Die vorgestellten Anonymisierungstechniken werden in Abbildung 2.6 kurz gegenüber gestellt. In dem Beispiel gehen wir von einer vereinfachten Relation aus der Musikdatenbank aus.

2.5.3 Das PArADISE-Framework

Das Akronym *PArADISE* [GH16a] steht für Privacy AwaRe Assistive Distributed Information System Environment und bezeichnet ein sogenanntes Langzeitrahmenprojekt des Lehrstuhles für Datenbank- und Informationssysteme der Universität Rostock. Das Framework unterstützt Forscher, Anwendungsentwickler und Nutzer bei der Umsetzung datenschutzfreundlicher, verteilter Informationssysteme als Teil des Internets der Dinge. Die Architektur des Frameworks ist in Abbildung 2.7 zu sehen.

PArADISE untergliedert sich in drei Phasen, die Unterstützung bei der Entwicklung von smarten Systemen leisten. In der *Entwicklungsphase* werden Modelle zur Erkennung und Vorhersage von Nutzeraktivitäten auf Basis von erfassten und annotierten Sensordaten abgeleitet. Machine-Learning-Algorithmen werden typischerweise in R oder Python (ML-Code) verfasst. In PArADISE werden diese Algorithmen in SQL-Anfragen überführt (ML2PSQL), die auf parallelen Datenbankmanagementsystemen (PDBMS), wie Postgres-XL[17] oder Actian Vector[18], ausführbar sind. Dadurch lassen sich mehrere Terabyte an (Sensor-)Daten effizient durch Intra-Operator-Parallelisierung verarbeiten. Zu den von PArADISE unterstützten Algorithmen gehören beispielsweise Hidden-Markov-Modelle, die auf Matrix-Matrix-Multiplikationen basieren. Deren parallele Umsetzung wird in [MH17] vorgestellt. Die nicht in SQL transformierbaren Codeanteile aus R bzw. Python, wie beispielsweise grafische Ausgaben, werden gesondert behandelt und nach Ausführung der Anfrage umgesetzt.

Während der Entwicklungsphase werden sämtliche zur Verfügung stehenden Daten genutzt, um die verschiedenen Modelle zur Situations-, Handlungs- und Intentionserkennung (siehe Unterabschnitt 2.1.1) abzuleiten. Dies ändert sich in der folgenden Phase zur *Daten- und Dimensionsreduktion*. In dieser Phase wird untersucht, wie die notwendige Datenmenge zur Ableitung der zuvor ermittelten Modelle aussieht. Dies kann beispielsweise dadurch erreicht werden, dass eine Verringerung der Anzahl an Sensoren bzw. deren Abtastfrequenz erfolgt. Zudem erfolgt der Einsatz von Data-Provenance-Techniken [AH18b] und Methoden zur Dimensionsreduktion, wie der Hauptkomponentenanalyse (engl.: Principal Component Analysis; für Details siehe [Sal16]).

In der abschließenden Nutzungsphase erfolgt die Bereitstellung des entwickelten Systems in der Systemumgebung. Im Gegensatz zu heute typischen Informationssystemen wird für die Implementierung von PArADISE keine einfache Client-Server-Architektur verwendet, sondern das Paradigma des Edge Computings (siehe Unterabschnitt 2.3.3) erweitert.

Einordnung

Die vorliegende Arbeit stellt das Konzept und die Umsetzung der Dekomposition komplexer Anfragen in einer verteilten Systemumgebung für die Nutzungsphase vor. Dabei ist mit *Verteilung* nicht (nur) die klassische Parallelisierung einer Anfrage auf mehreren gleichartigen Rechenknoten zu verstehen (horizontale Verteilung), sondern vielmehr die Zerlegung der Anfrage entlang der beteiligten Rechenknoten entlang der Achse von Sensorknoten bis hin zur Cloud. Diese Art der Verteilung wird in der vorliegenden Arbeit als *vertikale Verteilung* bezeichnet.

[17] `https://www.postgres-xl.org/`, zuletzt aufgerufen am 05.01.2022

[18] `https://www.actian.com/analytic-database/vector-analytic-database`, zuletzt aufgerufen am 05.01.2022

Vorname	Nachname	Geburtsjahr	Titel	Preis
Peter	Walch	1985	Cosmo Canyon	0.69
Peter	Walch	1987	Phantom Forest	1.11
Peter	Kramer	1988	Phantom Forest	1.11
Torben	Kramer	1987	Dancing Mad	0.92
Torben	Kramer	1987	Dancing Mad	0.73
Mathilde	Kramer	1987	Phantom Forest	1.13
Mathilde	Kramer	1998	Phantom Forest	1.13
Mathilde	Kramer	1985	Phantom Forest	1.12
Mathilde	Kramer	1985	Fishermans Horizon	1.15
Mathilde	Kramer	1999	Phantom Forest	1.10

Vorname	Nachname	Geburtsjahr	Titel	Preis
Peter	NULL	[198*]	Cosmo Canyon	0.67
Peter	NULL	[198*]	Phantom Forest	1.10
Peter	NULL	[198*]	Phantom Forest	1.11
Torben	Kramer	1987	Dancing Mad	0.93
Torben	Kramer	1987	Dancing Mad	0.73
Mathilde	Kramer	1987	Phantom Forest	1.12
Mathilde	Kramer	1987	Phantom Forest	1.11
Mathilde	Kramer	[19**]	Phantom Forest	1.13
Mathilde	Kramer	[19**]	Fishermans Horizon	1.16
Mathilde	Kramer	[19**]	Phantom Forest	1.10

Differential Privacy

Generalisierung

Vorname	Nachname	Geburtsjahr	Titel	Preis
Peter	Walch	1985	Cosmo Canyon	0.67
Peter	Kramer	1987	Phantom Forest	1.10
Peter	Kramer	1985	Phantom Forest	1.11
Torben	Kramer	1987	Dancing Mad	0.93
Torben	Kramer	1987	Dancing Mad	0.73
Mathilde	Kramer	1987	Phantom Forest	1.12
Mathilde	Kramer	1987	Phantom Forest	1.11
Mathilde	Kramer	1985	Phantom Forest	1.13
Mathilde	Kramer	1999	Fishermans Horizon	1.16
Mathilde	Kramer	1999	Phantom Forest	1.10

Permutation

Vorname	Nachname	Titel	Geburtsjahr	Preis
Peter	Walch	Cosmo Canyon	1985	1.11
Peter	Kramer	Phantom Forest	1987	1.10
Peter	Kramer	Phantom Forest	1985	0.67
Torben	Kramer	Dancing Mad	1987	0.93
Torben	Kramer	Dancing Mad	1987	0.73
Mathilde	Kramer	Fishermans Horizon	1999	1.16
Mathilde	Kramer	Phantom Forest	1985	1.13
Mathilde	Kramer	Phantom Forest	1987	1.12
Mathilde	Kramer	Phantom Forest	1987	1.11
Mathilde	Kramer	Phantom Forest	1999	1.10

Abbildung 2.6: Übersicht über die verschiedenen Klassen von Anonymisierungsverfahren anhand des laufenden Beispiels. In der Mitte der Abbildung befindet sich die Originalrelation vor Anwendung der Anonymisierung. Links oben ist die anonymisierte Relation nach Verwendung von Differential Privacy auf den Attributen *Nachname*, *Geburtsjahr* und *Preis* zu sehen. Die roten Zellen verdeutlichen die Attributwerte, die durch das Verfahren verändert wurden. Rechts daneben ist die Relation unter Verwendung der Generalisierungstechnik k-Anonymität mit $k = 2$ zu sehen. Als Einteilung für die Äquivalenzklassen wurden die Attribute *Vorname*, *Nachname* und *Geburtsjahr* gewählt. Die untere Relation stellt eine mögliche Permutation der Ausgangsrelation unter Beibehaltung der Attributgruppen *(Vorname, Nachname, Titel)* sowie *(Geburtsjahr und Preis)* dar. Zur Einteilung des horizontalen Splits wurde das Attribut *Vorname* gewählt.

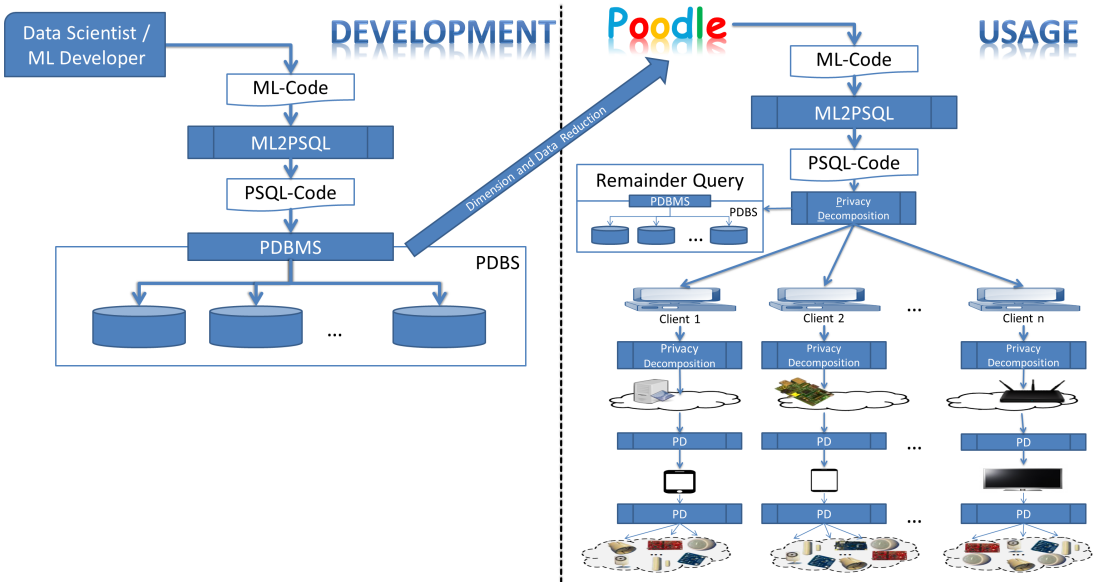


Abbildung 2.7: Die Architektur des PArADISE-Frameworks unterteilt sich in zwei Aspekte: Parallelisierung (links) und Datenschutz (rechts). Diese Abbildung erschien initial in [MH17].

Abgrenzung zum Stand der Technik

Im Gegensatz zum Stand der Technik im Bereich des Edge Computings erfolgt die Zerlegung der Anfrage nicht manuell, sondern automatisiert auf Basis der Anfragekapazitäten der beteiligten Rechenknoten. Unter der Anfragekapazität werden die unterstützten Operatoren der zu Grunde liegenden Anfragesprache aufgefasst. Die Zerlegung verfolgt dabei nicht das Ziel, die Anfrage performanter zu gestalten, sondern vielmehr den Aspekt der Datensparsamkeit (Details siehe Anhang A.6) auf technischer Ebene zu verwirklichen. Wie wir aber in den folgenden Kapiteln erkennen werden, können zur Umsetzung der Datensparsamkeit Techniken der Anfrageoptimierung adaptiert werden. Dadurch sind zielgerichtete, effiziente Datenauswertungen und Datenschutz keine unvereinbaren Aspekte für moderne Informationssysteme.

Zielsetzungen und Schwerpunkte

Die Nutzungsphase des PArADISE-Framework greift dabei alle in Abschnitt 1.1 formulierten Zielsetzungen und Schwerpunkte dieser Arbeit auf, die in den nachfolgenden Kapiteln gelöst und bearbeitet werden:

Allgemeines Lösungskonzept: Kapitel 3 stellt zunächst die grundlegende Konzeption des Anfrageprozessors mit seinen einzelnen Teilphasen zur Vor- und Nachverarbeitung relationaler Anfragen vor. Dabei wird aufgezeigt, an welchen Schritten der Anfrageverarbeitung in Datenbanksystemen eingegriffen werden kann, um überhaupt den Datenschutz fördernde Technologien einzusetzen.

Erkennung schützenswerter Daten: Anschließend werden in Kapitel 4 der Begriff des Quasi-Identifikators und dessen Rolle für die Parametrisierung von Anonymisierungsalgorithmen vorgestellt. Es wird eine neue Suchstrategie präsentiert, mit der die Quasi-Identifikatoren in hochdimensionalen Daten mit geringem zeitlichen Aufwand erkannt werden.

Vertikale Anfrageverteilung: Die beiden genannten Kapitel dienen zeitgleich zur Motivation des in Kapitel 6 vorgestellten Ansatzes zur vertikalen Verteilung von Anfragen. Durch den unumgänglichen Informationsverlust durch die Anwendung „klassischer“ Anonymisierungsverfahren werden häufig viele Anfrageergebnisse unbrauchbar – dies wird durch den vorgestellten Ansatz vermieden. Der Ansatz setzt auf eine automatisierte Entscheidung, welche Teile einer Anfrage für die Berechnung lokaler Zwischenergebnisse auf den leistungsschwächeren Komponenten eines (Assistenz-)Systems genutzt werden können. Dadurch werden statt Rohdaten lediglich vorgefilterte und voraggregierte Daten übertragen. Eine missbräuchliche Nutzung zu anderen Zwecken als den vorgesehenen Anwendungszwecken wird somit deutlich erschwert.

Kapitel 3

Konzept

Die Umsetzung der Anforderungen an den technischen Datenschutz bildet ein vielschichtiges und komplexes Problem, welches nicht aus nur einem Blickwinkel betrachtet werden muss. Einerseits müssen die rechtlichen Anforderungen an die Anonymität und die Forderung nach Datensparsamkeit bzw. -minimierung erfüllt werden, andererseits besteht seitens des Informationssystems Anspruch auf eine ausreichende Qualität der erhobenen Daten. Zudem liegt es in der Entscheidung des Nutzers, wie seine Daten genutzt werden; beim Datenschutz geht es schließlich nicht um den Schutz der Daten[1], sondern um den Schutz des Nutzers vor der missbräuchlichen Nutzung seiner Daten.

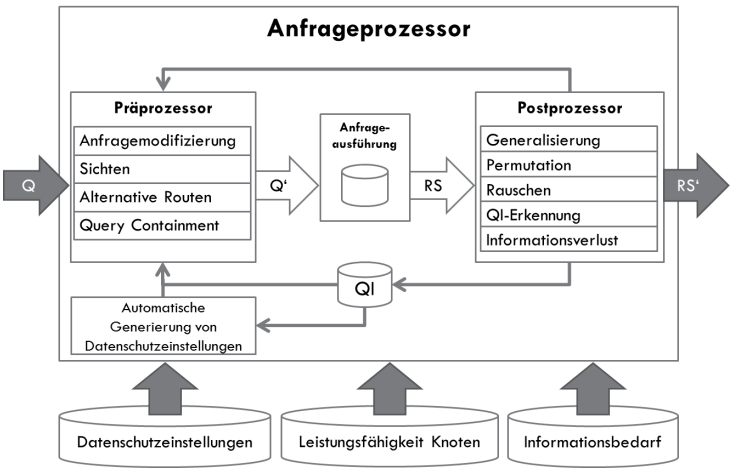


Abbildung 3.1: Übersicht über die einzelnen Bestandteile des Anfrageprozessors. Eine Anfrage Q wird im Präprozessor zunächst modifiziert, bevor sie ausgeführt wird. Der Postprozessor nimmt eine abschließende Anonymisierung der Ergebnisrelation RS vor. Weitere Parameter bilden die hinterlegten Datenschutzeinstellungen, die Leistungsfähigkeit respektive die Anfragekapazität der einzelnen Rechenknoten und der Informationsbedarf des Informationssystems.

An der Universität Rostock wurde im Rahmen des PArADISE-Projektes ein mehrstufiges Konzept (siehe Abbildung 3.1) zur Umsetzung des technischen Datenschutzes in Form eines Prozessors für relationale Anfragen entwickelt. Das Gesamtkonzept wurde erstmals in [GH16a] vorgestellt.

[1] Beim Schutz der Daten handelt es sich vielmehr um Datensicherheit. Dies kann als Grundlage für den Datenschutz aufgefasst werden (vgl. [BDH+13] und [Kos19]).

Die Verarbeitung einer Anfrage wird in drei Stufen umgesetzt: Der ***Anfrageprozessor*** interpretiert die Anfrage, führt sie auf der Datenbank aus und liefert eine Ergebnisrelation zurück. Dieses Vorgehen unterscheidet sich nicht von der normalenÄnfrageverarbeitung. Es zeigt uns aber auf, an welchen Stellen in die Anfrageverarbeitung eingegriffen werden kann: Der erste Zeitpunktpunkt ist zwischen dem Stellen der Anfrage seitens des Nutzers und vor der Ausführung der Anfrage auf der Datenbank. Die Modifikation des Anfrageergebnisses nach Ausführung der Anfrage und vor der Rückgabe an den Nutzer ist der zweite mögliche Zeitpunkt.

Vor der Ausführung auf der Datenbank schaltet sich der ***Präprozessor*** ein und bietet damit die Möglichkeit zur vorherigen Modifizierung der Anfrage. Die Datenbank erhält und führt somit nicht die originale Anfrage aus, sondern die im Präprozessor durch ***Query-Containment***-Techniken ***modifizierte Anfrage***. Bestehende Systeme (wie z. B. [MFHH02]) und Algorithmen (siehe z. B. [SCDFSM14]) zeigen, wie Daten ressourcenschonend auf Kleinstsystemen aggregiert werden können. Im Präprozessor wird entschieden, inwiefern solche Ansätze für eine datenschutzfreundliche, vertikal verteilte Anfrageverarbeitung geeignet sind.

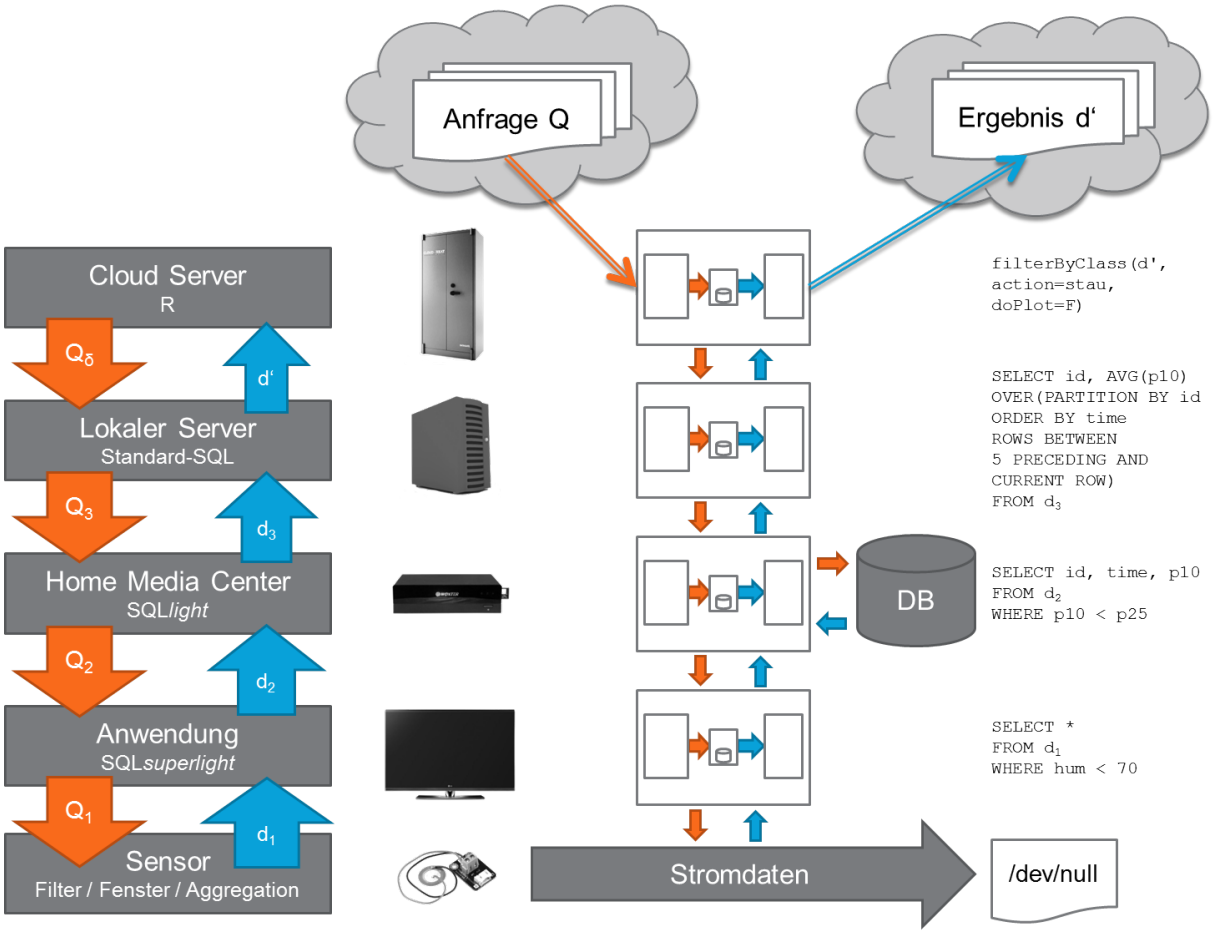


Abbildung 3.2: Der Anfrageprozessor wird entlang eines Edge-Netzwerkes auf die einzelnen Rechenknoten verteilt. Jeder Knoten übernimmt Teile Q_i der Anfrage Q, welche automatisiert bestimmt werden. Auf diese Weise können einfache Attribut-Konstanten- bzw. Attribut-Attribut-Selektionen auf die Sensoren bzw. sensornahen Ebenen verlagert werden, während komplexere Operationen, wie Fensterfunktionen und Visualisierungen, auf den höheren Ebenen realisiert werden.

Weitere wichtige Aspekte stellen die Komplexitäten hinsichtlich Zeit- und Ressourcenbedarf dar. Assistenzsysteme bestehen meist aus mehreren Rechnern, die eine unterschiedliche ***Leistungsfähigkeit*** besitzen. Daten werden in der Regel durch Sensoren erzeugt, die meist nicht in der Lage sind, die Verdichtung der Informationen direkt umzusetzen und als ***Sicht*** bereitzustellen. Im Kontrast dazu besitzen cloudbasierte Cluster, auf denen die Analysewerkzeuge ausgeführt werden, über eine extrem hohe Rechenleistung. Im Rahmen des PArADISE-Frameworks und als Kern dieser Arbeit wurde ein Verfahren entwickelt, welches – in Abgrenzung zu den in Abschnitt 2.3.3 vorgestellten Ansätzen – *automatisiert* bestimmt, welche Teile einer Anfrage von der Cloud auf die Knoten eines Edge-Netzwerkes bis hin zu den Sensoren ausgelagert werden können. Die Details dazu werden im Kapitel 6 genauer ausgeführt.

Das Rerouting von Anfragen über ***alternative Routen*** wird in dieser Arbeit nicht behandelt. Durch diese Technik werden Anfragen bei Ausfall eines Knotens innerhalb eines Edge-Netzwerkes umgeleitet, wobei zusätzlich die Verteilung der Anfrage neu vorgenommen wird.

Bevor das Ergebnis zurückgegeben wird, untersucht der ***Postprozessor*** die Ergebnisrelation auf Anonymitätseigenschaften und modifiziert ggf. die Relation unter Berücksichtigung von Anonymitätsmaßen und dem erforderlichen ***Informationsbedarf***. Dabei wird bestimmt, welche Anonymisierungsklasse – ***Generalisierung***, ***Permutation*** oder ***Rauschen*** – am besten geeignet ist. Ein kurzer Überblick zu diesem Verfahren ist in Abschnitt 3.4 gegeben.

In diesem Kapitel werden eingangs Ansätze zur ***Ermittlung von Quasi-Identifikatoren*** (Abschnitt 3.1) und zur ***(automatischen) Formulierung von Privatheitsansprüchen*** durch ***Datenschutzrichtlinien*** (Abschnitt 3.2) vorgestellt. Anschließend wird im Abschnitt 3.3 eine Übersicht über die Präprozessor-Phase hergestellt. In Abschnitt 3.4 werden abschließend die Konzepte des Postprozessors, insbesondere die Verwendung der *Kullback-Leibler-Divergenz* zur Bestimmung des Informationsverlustes (siehe Abschnitt 3.4.3), erläutert.

3.1 Quasi-Identifikatoren

Zum Schutz personenbezogener Daten existieren Konzepte wie k-Anonymität [Swe02], l-Diversität [MKGV07] und t-closeness [LLV07]. Diese Konzepte unterteilen die Attribute einer Relation in Schlüssel, Quasi-Identifikatoren (engl.: *quasi identifiers*; abgekürzt mit QI), sensitive Daten und sonstige, uninteressante Daten. Ziel ist es, dass die sensitiven Daten sich nicht eindeutig einer bestimmten Person zuordnen lassen. Innerhalb von Assistenzsystemen (siehe Abschnitt 2.1), bei denen die Zuordnung von Sensordaten zu einer bestimmten Person vorab bekannt ist, kann alternativ eine Identifikation von Handlungen, Intentionen und Bewegungsprofilen zu einer Person verhindert werden. Da durch Schlüsselattribute Tupel eindeutig bestimmt werden können, dürfen diese unter keinen Umständen zusammen mit den sensitiven Attributen veröffentlicht werden.

Während Schlüssel im Laufe des Datenbankentwurfes festgelegt werden, lassen sich Quasi-Identifikatoren erst beim Vorliegen der Daten feststellen, da sie von den konkreten Attributwerten der Relation abhängen. Im Gegensatz zu Schlüsselattributen ist zwar durch QIs keine eindeutige Identifizierung einer bestimmten Person oder Handlung möglich, jedoch kann durch die Verknüpfung von Informationen aus verschiedenen Relationen bzw. Datenquellen oder durch Hintergrundwissen der Personenbezug in vielen Fällen wiederhergestellt werden.

Unter *sensitiven Daten* werden diejenigen Informationen zusammengefasst, die es zu schützen gilt. Im Kontrast dazu sind die *insensitiven Daten*[2] solche, die nicht unbedingt geschützt werden müssen. Die Zuordnung in sensitive und insensitive Daten ist abhängig von der Anwendungsdomäne und der betroffenen Person, auf welche sich die erhobenen Daten beziehen. Dies bezieht sich primär auf die Zweckbindung der Datenerhebung und -verarbeitung und die vom Nutzer getroffenen Entscheidungen und Zustimmungen.

Die Grundlage für viele Anonymisierungsverfahren und einige im Rahmen dieser Arbeit entwickelte Algorithmen bilden die Quasi-Identifikatoren. Der Begriff Quasi-Identifikator wurde von Dalenius [Dal86] geprägt und bezeichnet „a subset of attributes that can uniquely identify most tuples in a table“. QIs einer Relation $r(R)$ sind somit Attributkombinationen, welche die Tupel von $r(R)$ fast eindeutig bestimmen. Für *most tuples* wird häufig ein Grenzwert `t` festgelegt, der bestimmt, ob eine Attributkombination ein Quasi-Identifikator ist oder nicht. Dieser Grenzwert sollte jedoch nur als Richtwert aufgefasst werden, da dadurch nur *mögliche* Attributkombinationen

[2] In der Fachliteratur werden *insensitive Daten* häufig auch als *weitere* bzw. *sonstige Daten* bezeichnet.

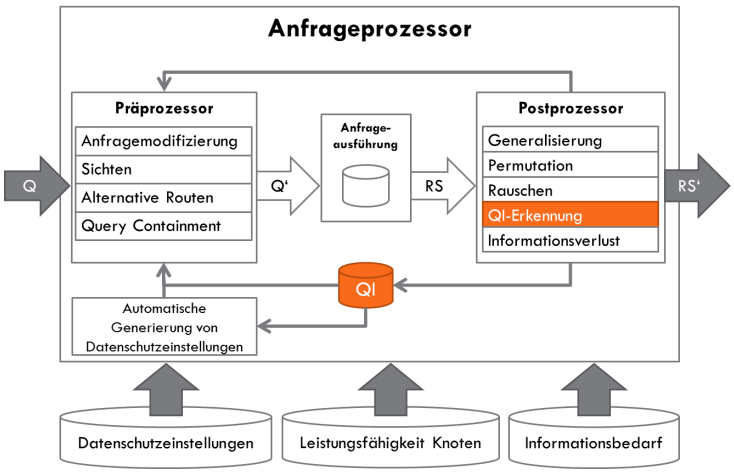


Abbildung 3.3: Einordnung der Quasi-Identifikatoren in den Anfrageprozessor. Innerhalb des Postprozessors erfolgt die Erkennung der Quasi-Identifikatoren (i) zum Start des Prozessors auf den Basisrelationen und (ii) nach Ausführung einer Anfrage auf den Ergebnisrelationen. Die ermittelten Quasi-Identifikatoren werden in einem zentralen Metadaten-Archiv gespeichert.

---

zur Identifikation einzelner Personen bzw. Tupel hervorgehoben werden. Eine konkrete Interpretation der gefundenen Quasi-Identifikatoren ist vom jeweiligen Anwendungsszenario abhängig. Die ermittelten Quasi-Identifikatoren sollten von einem Domänenexperten überprüft werden.

Definition: Quasi-Identifikator

Sei $r(R)$ eine Relation über dem Relationenschema R und $\{A_1, \ldots, A_n\} \subseteq R$ eine Teilmenge der Attribute von R. Zusätzlich sei t der Grenzwert für die Quasi-Identifikator-Eigenschaft. Eine Attributmenge $\{A_1, \ldots, A_n\}$ heißt Quasi-Identifikator, wenn $t \leq \frac{|\pi_{A_1,\ldots,A_n}(r(R))|}{|r(R)|}$ gilt.

Die obige Definition basiert auf der sogenannten *Distinct Ratio* nach [MX07], welche die Anzahl der unterschiedlichen Tupel in das Verhältnis zur Anzahl der Tupel in $r(R)$ setzt. Alternativ dazu kann die *Separation Ratio* verwendet werden, welche die Anzahl der eindeutigen Tupel nutzt.

Die Unterschiede zwischen den beiden Varianten werden in Kapitel 4 näher ausgeführt. In dem Kapitel wird weiterhin ein neues Verfahren entwickelt, das alle vorkommenden Quasi-Identifikatoren innerhalb einer Relation effizient ermittelt. Wie am Ende von Kapitel 4 in Abschnitt 4.6 sehen werden, existieren innerhalb einer Relation sowohl in realen als auch synthetischen Datensätzen eine zu große Anzahl an Quasi-Identifikatoren. Dies verursacht einen zu hohen Informationsverlust, wenn gleichzeitig die Privatheit durch die Anonymisierung gewährleistet werden soll. Um dies zu verhindern, werden zusätzlich die im Abschnitt 3.3 und Kapitel 6 eingeführten Konzepte zur Verteilung von Anfragen eingesetzt. Diese verhindern, dass zu viele Roh- bzw. Detaildaten an zentraler Stelle verarbeitet werden, wodurch die Anzahl an Quasi-Identifikatoren und die damit einhergehende Anonymisierung reduziert wird.

Beispiel: Quasi-Identifikatoren im laufendem Beispiel

In diesem Beispiel bestimmen wir die Quasi-Identifikatoren der nachfolgenden Relation mit zehn Tupeln und einem Grenzwert $t = 70\%$:

ID	Titel	Album	Genre	Titelnummer
1	You Don't Love Me	A Hard Road	Blues	2
2	You Don't Love Me	Live Oak	Rock	1
3	You Don't Love Me	Live Oak	Rock	2
4	Here Comes Sunshine	Dick's Picks	Rock	1
5	Here Comes Sunshine	Dick's Picks	Rock	1
6	Dreams	Live Oak	Rock	1
7	Dreams	Live Oak	Rock	1
8	Dreams	Skydog	Rock	2
9	Dreams	Wanee Festival	Rock	3
10	Dreams	Live Oak	Rock	3

Wie unschwer zu erkennen ist, stellt das Attribut ID den Schlüssel der Relation dar. Schlüsselattribute übersteigen per Definition den Grenzwert p und sind somit Quasi-Identifikator. Weitere Quasi-Identifikatoren sind $\{Album,\ Titelnummer\}$ sowie $\{Titel,\ Genre,\ Titelnummer\}$. Zusätzlich sind alle Attributkombinationen, welche diese drei QIs als Teilmenge enthalten, ebenfalls Quasi-Identifikatoren. Kein Quasi-Identifikator ist beispielsweise $\{Titel,\ Album\}$, da für diese Attributkombination nur sechs unterschiedliche Attributwerte in der Relation auftauchen.

Die Erkennung von Quasi-Identifikatoren erfolgt innerhalb des Anfrageprozessors zu zwei Zeitpunkten: Eine initiale Bestimmung erfolgt nach dem Start des Anfrageprozessors, sofern der jeweilige Rechenknoten direkten Zugriff auf einen Teil der im System hinterlegten Basisrelationen besitzt. Die auf diese Art erhobenen Quasi-Identifikatoren werden als Metadaten im System gespeichert. Diese Metadaten werden einerseits für die automatische Generierung von Datenschutzprofilen (siehe Abschnitt 3.2) genutzt, andererseits für die Anfragetransformation im Präprozessor (siehe Abschnitt 3.3). Zudem erfolgt eine Ermittlung der Quasi-Identifikatoren im Postprozessor, nachdem eine Anfrage ausgeführt wurde. Somit kann zur Ausführungszeit der Anfrage entschieden werden, ob eine Ergebnisrelation anonymisiert werden muss, und wenn ja, mit welchen Algorithmen und Parametern. Details dazu sind in Abschnitt 3.4 zu finden.

Literatur: In [GH14] wurde erstmals das im Anfrageprozessor von mir entwickelte Verfahren beschrieben, mit dem sich Quasi-Identifikatoren effizient in hochdimensionalen Basisrelationen ermitteln lassen. Kapitel 4 stellt den entwickelten Algorithmus in detaillierter Form vor. Aufbauend auf diesem Verfahren wurde im Rahmen einer von mir betreuten Bachelorarbeit [Ros19] ein Algorithmus zur Erkennung von Quasi-Identifikatoren auf Verbundrelationen entwickelt.

3.2 Datenschutzprofile

Das zentrale Anliegen des Datenschutzes ist es, dass der einzelne Mensch selbst entscheiden kann, ob eine Erhebung seiner personenbezogenen Daten erfolgen darf und wie diese gespeichert und verarbeitet werden können. Datenschutzeinstellungen müssen dabei einen guten Kompromiss zwischen feingranularen Einstellungen und Verständlichkeit bieten. Studien haben gezeigt, dass bereits die von Facebook angebotenen Sichtbarkeitsbereiche (von *nur ich* über *jeder meiner Freunde* bis *jeder*) bei einem Großteil der Nutzer zu Missverständnissen führen – über 60% der in [LGKM11] befragten Nutzer konnten nicht die bevorzugte Einstellung tätigen.

Um diesen Anforderungen (siehe Anhang B.1.1) gerecht zu werden, wurden verschiedene Standards entwickelt, welche die Formulierung von Privatheitsansprüchen ermöglichen. Daneben existieren verschiedene Arten zu Access-Control-Lists, welche einen feingranularen Dateizugriff auf Betriebssystem- bzw. Datenbankebene regeln (vgl. Kapitel Grundlagen, Anhang A.2). Im Rahmen des PArADISE-Projektes wurde zudem auf Basis der *Platform for Privacy Preferences* (siehe [W3C07] und Anhang B.1.2) eine modifizierte Auszeichnungssprache zur Formulierung von Privatheitsansprüchen in smarten Umgebungen geschaffen. Diese wird im Anhang B.1.3 näher vorgestellt und ermöglicht eine feingranulare und zweckgebundene Vergabe von Zugriffsrechten.

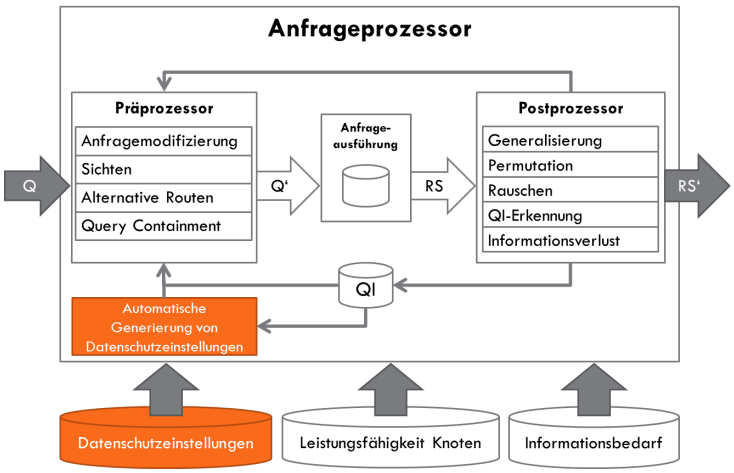


Abbildung 3.4: Einordnung der Einstellungen für Datenschutzprofile in den Anfrageprozessor

---

Literatur: Dieses Konzept basiert auf dem von mir verfassten Artikel *Generating Privacy Constraints for Assistive Environments* [GH15a], welcher PP4SE in seiner ersten Version vorstellt. Der dazugehörige technische Bericht [GH15b] stellt weitere Details, insbesondere zum assoziierten XML Schema, vor. Eine von mir betreute Bachelorarbeit [Mei15] untersucht die Eignung und Umsetzung von Data-Mining-Verfahren zur Adaption von Datenschutzrichtlinien am Beispiel von PP4SE. Im Rahmen einer studentischen Projektarbeit [RLRG16] wurde das Schema um Parameter für verschiedene Anonymisierungsmaße und -verfahren erweitert.

3.3 Präprozessor

Bei jeder Ausführung einer Anfrage startet als Erstes der Präprozessor. Im Präprozessor werden die gestellten Anfragen analysiert und ggf. abgeändert. Dabei wird zum einen untersucht, ob die Anfrage Quasi-Identifikatoren (siehe Kapitel 4) enthält, zum anderen, ob die Anfrage die spezifizierte Datenschutzrichtlinie (Details siehe Abschnitt 3.2) einhält. Im Ergebnis der Präprozessor-Phase entsteht eine modifizierte Anfrage: Eine zerlegte Anfrage, die normalerweise nicht direkt auf dem Ausgangssystem ausgeführt wird, jedoch an andere Teile des Informationssystems weitergeleitet wird, um das Ergebnis vertikal verteilt zu berechnen.

In Kapitel 6 wird das Konzept für diese vertikale Verteilung vorgestellt – der Hauptbeitrag dieser Arbeit. Der Ansatz bedient sich dabei der etablierten Anfrageverarbeitung in Datenbanken (siehe beispielsweise [Fre87] und [Gra93]): Zunächst werden in der übergebenen Anfrage die Sichten in Basisrelationen aufgelöst und die Anfrage anschließend in einen Ausdruck der Relationenalgebra überführt. Im Anschluss wird der Anfrageplan entschachtelt und einzelne Ausdrücke werden, sofern erforderlich, vereinfacht. Vor der abschließenden Codeerzeugung und Ausführung auf einem konkreten Datenbanksystem erfolgt die Optimierung des Anfrageplans. Als weitere Möglichkeit eignet sich neben der algebraischen Optimierung ein CHASE-basierter Ansatz. Dieser wird in Abschnitt 7.2.1 kurz skizziert.

In der Anfrageoptimierung unterscheidet sich das in Abschnitt 6.1 näher vorgestellte Konzept: Während die klassische logische Anfrageoptimierung darauf abzielt, durch die Anwendung von Heuristiken Teilziele des Anfrageplans so zu vertauschen bzw. umzuformen, dass die Anfrage möglichst effizient abgearbeitet wird, stehen aus Sicht des Datenschutzes andere Optimierungsziele im Vordergrund. Durch die vertikale Verteilung einer Anfrage soll der Anfrageplan so umgeformt werden, dass möglichst große Teile der Anfrage bereits auf Geräten mit beschränkter Anfragekapazität ausgeführt werden können. Diese Kapazitäten werden durch eine Menge an unterstützten Operatoren der relationalen Algebra für jedes der an der Verarbeitung der Anfrage beteiligten Geräte angegeben. Beide Ziele stehen nicht zwangsweise im Widerspruch, sondern ergänzen sich teilweise. Die zum Lösen

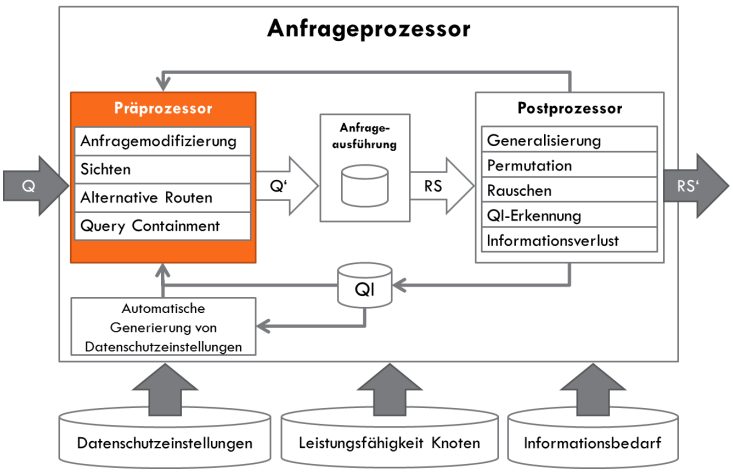


Abbildung 3.5: Anordnung des Präprozessors im Anfrageprozessor

des Query-Containment-Problems entwickelten Techniken (siehe Kapitel 5) tragen, mit leichten Veränderungen, zur Minimierung personenbezogener Daten bei.

Das entwickelte Konzept berechnet auf Basis der Anfragekapazitäten eine optimale Verteilung der Anfrage, sodass unter den Aspekten der Datensparsamkeit und -minimierung möglichst viele Daten vorverarbeitet, aggregiert und gefiltert werden können. Dies führt zu einer Reduzierung bzw. Vermeidung von personenbezogenen Daten auf den höheren Ebenen innerhalb des Informationssystems. Werden personenbezogene Daten bereits auf den Endgeräten des Nutzers so verdichtet, dass sie keinen bzw. nur einen sehr geringen Personenbezug aufweisen, muss der Betreiber des Informationssystems weniger eigene Anstrengungen hinsichtlich der Umsetzung bzw. Einhaltung des Datenschutzes unternehmen – ein Gewinn für alle Beteiligten.

Das Konzept kann nicht garantieren, dass sämtlicher Personenbezug aus den Daten entfernt wird. Im Postprozessor kann nach Ausführung der transformierten Anfrage auf eine Vielzahl von Anonymisierungsverfahren zurückgegriffen werden. Details zu deren Einbettung, Implementierung und Anwendung folgen im nächsten Abschnitt.

Literatur: Das allgemeine Konzept zur Umsetzung der Datensparsamkeit in Edge-Netzwerken und dessen Einordnung in den Anfrageprozessor wurden initial in [GH16a] von mir publiziert. Mein Beitrag [GH16b] stellt die formalen Grundlagen der Anfragezerlegung vor; in [GKB+16] wird dessen Anwendung in IoT-Umgebungen gezeigt. Die Beiträge [GH17] und [GH18] übertragen mein Konzept der Anfragezerlegung auf komplexe Aggregatfunktionen. In den von mir betreuten studentischen Abschlussarbeiten von Kluth [Klu17] und Langmacher [Lan17] werden die Einsatzmöglichkeiten verschiedener Query-Containment-Techniken für die Anfragezerlegung evaluiert.

3.4 Postprozessor

Der Postprozessor beschäftigt sich mit verschiedenen Ansätzen zur Anonymisierung des Anfrageergebnisses (siehe Unterabschnitt 3.4.1). Der Postprozessor untersucht zudem, ob die durch Folgen von Anfragen (siehe Unterabschnitt 3.4.2) entstehenden Teilrelationen weiterhin den Anonymitätskriterien entsprechen. Das Problem des durch die Anonymisierung entstehenden Informationsverlustes wird ebenfalls betrachtet (siehe Unterabschnitt 3.4.3). Abschließend wird im Postprozessor der Frage nachgegangen, ob anonymisierte Daten durch gezielte Angriffe wieder deanonymisiert werden können.

Die Anonymisierung der Daten erfolgt nach einem vierstufigen Konzept: Im ersten Schritt wird ermittelt, welche

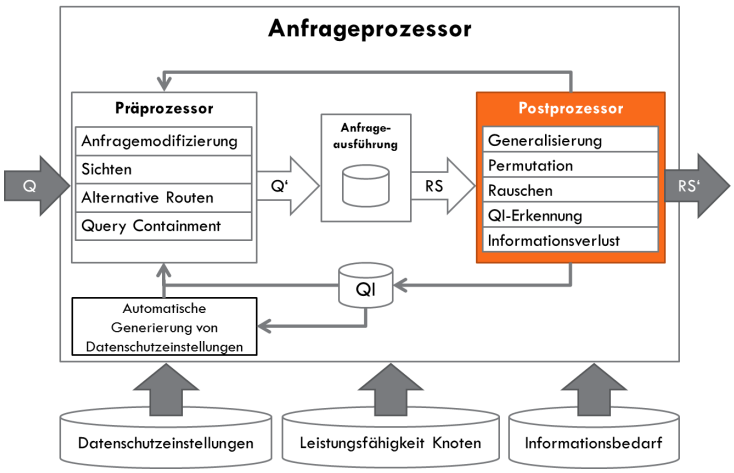


Abbildung 3.6: Anordnung des Postprozessors im Anfrageprozessor

Daten geschützt werden müssen (sensible Daten laut Datenschutzrichtlinie nach Abschnitt 3.2) und welche Daten als Quasi-Identifikator geeignet sind (siehe dazu Kapitel 4). Anschließend wird überprüft, welche Bedrohungen von einer Anfrage ausgehen. Dabei wird untersucht, welche QIs in der Projektionsliste der Anfrage enthalten sind. Ausgehend von der durch die QI abgedeckten Attributmenge werden in der dritten Stufe mögliche Werte für die Parameter der Anonymisierungsalgorithmen und -maße festgelegt. Zuletzt erfolgt die Auswahl eines geeigneten Anonymisierungsalgorithmus, der die optimale Anonymisierung bei gleichzeitig hohem Informationsgehalt liefert. Dabei können, je nach Art der Anfrage, verschiedene Ansätze, wie Permutation, Generalisierung und Rauschen, genutzt werden. Die Umsetzung dieser Verfahren wird im folgenden Unterabschnitt 3.4.1 kurz vorgestellt.

3.4.1 Anonymisierung

Durch Anonymisierung wird der Personenbezug aus den erfassten, gespeicherten bzw. angefragten Daten entfernt. Dadurch gelten diese Informationen nicht mehr als personenbezogen und fallen nach Auffassung von Voigt und von dem Bussche [VvdB18] somit nicht mehr unter die diversen Datenschutzregelungen und -gesetze. In PArADISE werden die in Abschnitt 2.5 eingeführten Klassen von Anonymisierungsverfahren – Generalisierung, Permutation und Differential Privacy – unterstützt.

Sofern genügend Rechenleistung auf dem für die Anonymisierung zuständigen Rechenknoten zur Verfügung steht und die Anwendung nicht zeitkritisch ist, können mehrere unterschiedliche Anonymisierungen für ein Anfrageergebnis erzeugt und diese einer Anonymitäts-Informationsgehalt-Bewertung unterzogen werden. Die qualitativ *beste* Anonymisierung wird daraufhin für diese und ähnlich strukturierte Anfragen verwendet. Nähere Details zur Umsetzung der einzelnen Verfahren werden in Anhang B.2.2 beschrieben.

Offene Risiken: Unklar bleibt, ob durch automatisierte Verfahren unter Zuhilfenahme weiterer Datenquellen, wie öffentliche Register, der Personenbezug aus den anonymisierten Daten wiederhergestellt werden kann. [AMS+15] stellt eine gute Klassifikation zu Deanonymisierungsverfahren vor. Die einzelnen Verfahren werden in die vier Gruppen Hintergrundwissen, Pläne & Strategien (z. B. Matching- und klassifikationsbasierte Verfahren), Ausführung (z. B. Angriffe via Inferenz) und Ergebnisanalyse (z. B. Identitätsenthüllung und Verlinkung von Datensätzen) unterteilt. In [RMPADF13] werden die verschiedenen Angriffsmöglichkeiten zur Deanonymisierung von Datensätzen systematisch untersucht und kategorisiert. Ausgehend von verschiedenen Eigenschaften, wie dem Hintergrundwissen des Angreifers und seiner zur Verfügung stehenden Angriffsstrategien, wird ein konkretes Anonymisierungsmaß vorgeschlagen. Dieses verhindert den beschriebenen Angriff des Angreifers so weit wie möglich.

Die Anonymisierungsverfahren müssen entsprechend dem Stand der Technik angepasst werden bzw. deren Parametrisierung zu Gunsten einer stärkeren Anonymisierung gewählt werden. Zusätzlich wird nach [VvdB18] eine Kombination der verschiedenen Techniken empfohlen. In PArADISE wurden speziell für auf Generalisierung basierte Verfahren mögliche Angriffs- bzw. Deanonymisierungsszenarien analysiert und Handlungsempfehlungen zur Stärkung der gewählten Anonymisierung entwickelt. Details dazu sind in [Gol17] bzw. [GGH17] zu finden.

3.4.2 Folgen von Anfragen

Die Anonymität einzelner personenbezogener Daten muss auch über mehrere, beliebige Anfragen hinweg gewahrt werden. Ein weiterer Aspekt, der im Postprozessor realisiert wurde, ist die Überprüfung von anonymisierten Relationen auf mögliche Angriffsmöglichkeiten durch Ausnutzung von Statistiken und Hintergrundwissen.

Zwar unterliegen Einzeleinträge von personenbezogenen Daten dem Datenschutzgesetz, jedoch dürfen statistische Informationen in Form von aggregierten Werten ggf. veröffentlicht werden, sofern dies, beispielsweise, vertraglich vereinbart wurde. Angriffe sind zum Beispiel durch die Ausführung mehrerer statistischer Anfragen möglich, wenn der paarweise Durchschnitt mehr als einen im Vorfeld festgelegten Grenzwert von m Tupeln enthält. Das folgende Beispiel zeigt, wie durch eine Folge von zwei einfachen Anfragen und dem Einsatz von Hintergrundwissen weiteres Wissen abgeleitet werden kann.

Beispiel: Indirekte Berechnung der Einkäufe eines Nutzers (in Anlehnung an [SSH18, Seite 506])

Die beiden nachfolgenden Anfragen berechnen die Gesamtsumme aller Einkäufe aus dem Preis aller „Dragonsong"-Einkäufe und der Nutzer `Holger` bzw. `Andreas` (als Attribut `uid`). Unter den Annahmen, dass die beiden Nutzer nicht den „Dragonsong"-Titel gekauft haben und dass die Summe der Einkäufe eines der beiden Nutzer bekannt ist, lässt sich die Summe der Einkäufe des jeweils anderen Nutzers (als sensitive Information) bestimmen.

```
1 SELECT sum(price) AS holgerSumPrice
2 FROM buys NATURAL JOIN users NATURAL JOIN tracks
3 WHERE uid = 'Holger'
4 OR title = 'Dragonsong'
```

```
1 SELECT sum(price) AS andreasSumPrice
2 FROM buys NATURAL JOIN users NATURAL JOIN tracks
3 WHERE uid = 'Andreas'
4 OR title = 'Dragonsong'
```

Sind o. B. d. A. die Summe der Einkäufe von Andreas (`andreasPrice`) bekannt, kann durch die Berechnung

$$holgerPrice = holgerSumPrice - (andreasSumPrice - andreasPrice) \tag{3.1}$$

die Summe der Einkäufe von Holger (`holgerPrice`) indirekt bestimmt werden.

Um den Datenschutz zu wahren, sollten Anfragen, die den Grenzwert m unterschreiten, abgelehnt werden oder ein anonymisiertes Ergebnis zurückliefern. Es kann abgeschätzt werden, wie viele Anfragen mindestens benötigt werden, um ein einzelnes Tupel bzw. dessen Äquivalenzklasse zu ermitteln. Eine Übersicht zu verschiedenen Anfrageklassen und wie eine Folge von Anfragen für einen Angriff konstruiert werden kann, ist in [DDS79] zu finden. Im Rahmen von PArADISE wurde für die Verhinderung von Folgen von Anfragen ein einfacher, mengentheoretischer Ansatz entwickelt. Dieser und verwandte Ansätze werden in den folgenden Abschnitten kurz vorgestellt.

3.4.3 Informationsverlust

Durch die Verwendung von Anonymisierungstechniken entsteht zwangsweise ein Verlust an Informationen – sei es pro Attribut, pro Tupel oder über die gesamte Relation hinweg. Nach Shannon [Sha48] wird einer Information immer ein gewisses Maß an Bedeutung zugewiesen:

> **Zitat:** Information nach Shannon [Sha48]
> Frequently the messages have meaning; that is they refer to or are correlated according to some system with certain physical or conceptual entities. These semantic aspects of communication are irrelevant to the engineering problem. The significant aspect is that the actual message is one selected from a set of possible messages.

Der Wert einer Information kann auf vielfältige Weise bemessen werden: Monetär, beispielsweise als Kaufpreis oder als Wiederherstellungskosten, oder zum Bemessen des Aufwandes, der zum Schutz oder zum Erlangen der Information notwendig ist. Im Folgenden betrachten wir den Wert einer Information aus zwei Sichtweisen:

1. **Aus Sicht der Auswertung**: Welchen Wert hat ein einzelnes Attribut bzw. Tupel für die gesamte Analyse, d. h., wie viel trägt es zur Auswertung bei?

2. **Aus Sicht des Datenschutzes**: Wie schützenswert ist eine Information? Wie sehr möchte ein Nutzer eine einzelne Information schützen? Wie sehr trägt ein Attribut zur Identifizierung eines Datensatz bei?

Unter *Informationsverlust* wird der Unterschied zwischen den Werten zweier Datensätze bzw. Einzelformationen I_1 und I_2 verstanden, wobei I_1 die Information vor und I_2 die Information nach dem Verlust von einzelnen Datensätzen bzw. der Reduzierung der Genauigkeit einer Einzelinformation ist. Für die Berechnung des Informationsgehaltes einer Information bzw. des Informationsverlustes existieren viele verschiedene Möglichkeiten, wie beispielsweise die Berechnung der Entropie. Die Entropie E als Maß des Informationsgehaltes einer Information I ist nach Shannon [Sha48] wie folgt definiert:

$$E(I) = -\sum_{i=1}^{n} p_i * log_2(p_i) \tag{3.2}$$

Der Logarithmus zur Basis 2 wurde gewählt, da die Entropie aus dem Bereich der Signalverarbeitung stammt, bei der die Codierung von Informationen durch Bits realisiert wird. Um den Informationsverlust durch Anonymisierungstechniken besser einschätzen zu können, d. h., den verbleibenden Wert der Information für die Auswertung bzw. die Güte der gesamten Analyse zu bestimmen, wird eine angepasste Version der Kullback-Leibler-Divergenz [KL51] (KLD; auch: Kullback-Leibler-Distanz) eingesetzt, um alle in PArADISE verwendeten Anonymisierungstechniken zu unterstützen. Details zu den Anpassungen können Anhang B.3 entnommen werden.

Die Kullback-Leibler-Divergenz wird im PArADISE-Framework an verschiedenen Stellen eingesetzt. Anonymisierungsverfahren, insbesondere diejenigen, die auf Generalisierung beruhen, nutzen die Kullback-Leibler-Divergenz, um den Informationsverlust vor der Herausgabe abzuschätzen. Als Anwendungsbeispiel wird die Berechnung in der (α, k)-Anonymity-Implementierung [WLFW09, WLFW06] verwendet, die in PArADISE als Modul implementiert wurde [Joh16]. Zudem wird die Kullback-Leibler-Divergenz bei der Abschätzung des Informationsverlustes bei der vertikalen Verteilung von Anfragen (siehe Kapitel 6) verwendet. Im letzteren Anwendungsfall wird darauf geachtet, dass möglichst viele Informationen verloren gehen, da das in Kapitel 6 vorgestellte Verfahren zwar die Daten schrittweise extrem verdichtet, das Ergebnis jedoch semantisch äquivalent bleibt.

3.5 Demonstrator

Die Implementierung des Demonstrators für den JDBC-Treiber entstammt ursprünglich aus der Arbeit einer von mir betreuten studentischen Projektgruppe [RLRG16]. Im Rahmen eigener bzw. betreuter Arbeiten wurden

zusätzliche Funktionalitäten, wie Anonymisierungsalgorithmen [Joh16], Empfehlungssysteme für Datenschutzrichtlinien [Mei15] sowie Algorithmen für Query Containment-Überprüfungen (siehe [Klu17] und Kapitel 6), implementiert.

Details zu den Implementierungen der eigenen entwickelten Techniken sind in den Implementierungsabschnitten der Kapitel 4 und 6 aufgeführt. In Anhang F wird ein Überblick über die Rahmenarchitektur und die grundlegenden Funktionalitäten zum Betrieb in einer Mehrschichtenarchitektur gegeben. Der vollständige Programmcode befindet sich im Begleitmaterial bzw. im Git-Repository des PArADISE-Projektes[3].

3.6 Zusammenfassung

In diesem Kapitel wurde das für PArADISE entwickelte Konzept zur technischen Umsetzung von Datenschutzaspekten vorgestellt. Dabei wurden die folgenden neuen wissenschaftlichen Beiträge herausgearbeitet:

- Entwicklung einer Auszeichnungssprache zur Formulierung von Datenschutzaspekten,
- Direkte Integration einer Permutation-basierten Anonymisierungstechnik in die Anfrageverarbeitung,
- Ausarbeitung eines Algorithmus zur Sicherstellung der k-Anonymität über eine begrenzte Folge von Anfragen und
- Modifikation der Kullback-Leibler-Divergenz zur Anwendung in einem iterativen Generalisierungsalgorithmus.

Als weitere Kernkonzepte wurden Verfahren zur effizienten Erkennung von Quasi-Identifikatoren und zur Umsetzung des Aspektes der Datensparsamkeit in IoT-basierten Umgebungen entwickelt. Diese Verfahren werden in den folgenden Kapiteln 4 und 6 detailliert beschrieben.

[3] `https://git.informatik.uni-rostock.de/dbis/PArADISE/PArADISE`, zuletzt aufgerufen am 05.01.2022.

Kapitel 4

Erkennung von Quasi-Identifikatoren

Im vorherigen Kapitel 3 wurde mehrfach auf den Einsatz von Quasi-Identifikatoren (QIs) in verschiedenen Techniken, wie der Parametrisierung von Anonymisierungsalgorithmen, eingegangen. QIs nehmen somit eine zentrale Rolle im Datenschutz ein, da ohne sie in vielen Fällen nicht entschieden werden kann, anhand welcher Merkmale die Identifikation einer Person ermöglicht wird.

In diesem Kapitel erfolgt in den ersten Abschnitten 4.1 und 4.2 eine Einführung in die formalen Grundlagen von Quasi-Identifikatoren sowie zu bestehenden Konzepten zu deren Ermittlung. Im darauffolgenden Abschnitt 4.3 wird eine Verbesserung bestehender Verfahren zum Erkennen von Quasi-Identifikatoren in hochdimensionalen Daten vorgestellt. Die darauffolgenden Abschnitte 4.4 und 4.5 gehen auf die Implementierung bzw. Evaluierung des Verfahrens ein. Im vorletzten Abschnitt 4.6 wird näher auf die Auswirkungen der gefunden Quasi-Identifikatoren auf die Anonymisierung von personenbezogenen Daten eingegangen. Abschnitt 4.7 fasst die erzielten Ergebnisse dieses Kapitels zusammen.

Literatur: Dieses Kapitel basiert auf dem Artikel [GH14], in dem das neu entwickelte Verfahren zuerst vorgestellt wurde. In Abgrenzung zu dem Artikel werden in diesem Kapitel eine detailliertere Übersicht zum Stand der Technik gegeben und die formalen Grundlagen des Verfahrens, insbesondere für die eingeführte Wichtungsfunktion, beschrieben. Zudem werden Modifikationen des Konzepts für Änderungsoperationen auf bereits untersuchten Relationen vorgestellt.

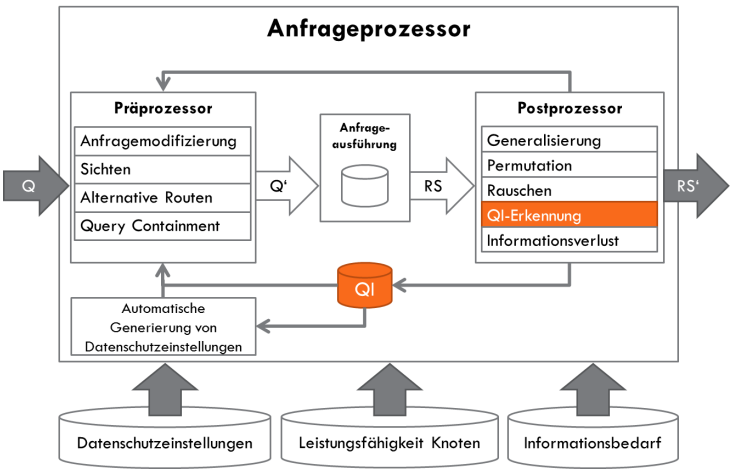


Abbildung 4.1: Anordnung der Quasi-Identifikatoren und deren Erkennung im Anfrageprozessor

4.1 Begriffsbestimmungen

Wie bereits in Kapitel 3 beschrieben, bezeichnet ein Quasi-Identifikator „a subset of attributes that can uniquely identify most tuples in a table“ [Dal86]. Aus Sicht relationaler Datenbanken lässt sich dies wie folgt formalisieren:

Definition: Quasi-Identifikator

Sei $r(R)$ eine Relation über dem Relationenschema R und $\{A_1, \ldots, A_n\} \subseteq R$ eine Teilmenge der Attribute von R. Zusätzlich sei t der Grenzwert für die Quasi-Identifikator-Eigenschaft. Eine Attributmenge $\{A_1, \ldots, A_n\}$ heißt Quasi-Identifikator, wenn

$$t \leq p = \frac{|\pi_{A_1,\ldots,A_n}(r(R))|}{|r(R)|} \tag{4.1}$$

gilt.

Wie beim Datenbankentwurf reicht es auch für die Angabe von Quasi-Identifikatoren aus, wenn die minimale Menge von Attributen angegeben wird, welche die Eigenschaft eines QI hat. Eine solche Menge wird als minimaler Quasi-Identifikator bezeichnet.

Definition: Minimaler Quasi-Identifikator

X heißt minimaler Quasi-Identifikator (mQI) über der Relation $r(R)$, wenn X ein Quasi-Identifikator über $r(R)$ ist und jede nicht-leere Teilmenge Y von X kein Quasi-Identifikator ist.

X ist mQI: X ist QI $\wedge$ ($\nexists Y \subset X : (Y \neq \oslash) \wedge (Y$ ist QI$)$)

Insbesondere ist X kein minimaler Quasi-Identifikator, wenn eine Teilmenge X-{A} von X mit A $\in$ X existiert, die ein Quasi-Identifikator ist. Das Finden aller Quasi-Identifikatoren stellt ein NP-vollständiges Problem dar, weil die Menge der zu untersuchenden Teilmengen exponentiell zur Anzahl der Attribute einer Relation steigt. Besteht eine Relation aus n Attributen, so existieren insgesamt 2^n-1 Attributkombinationen, für die ermittelt werden muss, ob sie ein QI sind.

Der Wert p lässt sich auf der Basis einer der beiden folgenden Metriken definieren [MX07]:

Definition: Distinct Ratio

Die *Distinct Ratio* zählt die Anzahl der unterschiedlichen Belegungen für $A_1, \ldots, A_n$ in der Relation $r(R)$ mit $|r(R)| > 0$. Mittels Relationenalgebra lässt sich der prozentuale Anteil p folgendermaßen formulieren:

$$p = \frac{|\pi_{A_1,\ldots,A_n}(r(R))|}{|r(R)|} \tag{4.2}$$

Diese Metrik wurde für die obige Definition eines Quasi-Identifikators verwendet. Eine Verschärfung stellt die *Separation Ratio* dar, welche mehrfach auftretende Attributwerte nicht einfach eliminiert, sondern im Zähler unberücksichtigt lässt.

Definition: Separation Ratio

Die *Separation Ratio* gibt an, wie viele Kombinationen der Attributwerte für die gegebene Attributmenge $A_1, \ldots, A_n$ nur einmal in der Relation $r(R)$ mit $|r(R)| > 0$ vorkommen. In der Relationenalgebra lässt sich der prozentuale Anteil p wie folgt darstellen:

$$p = \frac{|\sigma_{cnt=1}(\gamma_{;cnt \leftarrow cntd_{A_1,\ldots,A_n}}(r(R)))|}{|r(R)|} \tag{4.3}$$

Wird für den Grenzwert t der Wert 1 gewählt, so sind die gefundenen Quasi-Identifikatoren mit diesem Grenzwert auch Schlüssel der Relation. Um eine Vergleichbarkeit unseres Algorithmus mit dem von Motwani und Xu zu gewährleisten, verwenden wir ebenfalls die *Distinct Ratio* nach [MX07]. Auf die Implementierung für die Berechnungen beider Metriken wird in Abschnitt 4.4 eingegangen.

4.2 Vorarbeiten

In diesem Abschnitt wird näher auf den Begriff des Quasi-Identifikators und dessen Bestimmungsmöglichkeiten eingegangen. Dabei werden zunächst verwandte Verfahren zur Ermittlung von Schlüsseln betrachtet. Abschließend wird kurz auf die Einsatzmöglichkeiten von Quasi-Identifikatoren eingegangen.

4.2.1 Erkennung von Schlüsseln

Die Erkennung von Schlüsselattributen ist im allgemeinen Fall NP-vollständig [GKM$^+$03], wobei der zu durchsuchende Lösungsraum exponentiell ist [HQA$^+$13]. Zwar kann die Komplexität des Problems nicht verändert werden, allerdings ist es möglich, durch geschicktes Vorgehen die benötigte Rechenzeit und den Platzbedarf für die Berechnungen möglichst gering zu halten. Im Folgenden werden kurz einige bekannte und aktuelle Ansätze zur Einschränkung des Suchraumes skizziert.

Effiziente Erfragung von identifizierenden Attributmengen

In [Kle98] wird ein Verfahren zur Suche von identifizierenden Attributmengen in relationalen Strukturen vorgestellt. Wird für eine Menge von Attributen X einer Relation $r(R)$ erkannt, dass diese eine identifizierende Attributmenge für $r(R)$ ist, so sind auch alle Obermengen von X identifizierende Attributmengen für $r(R)$. Besteht X beispielsweise aus den drei Attributen A, B, und C, wobei bekannt ist, dass B eine identifizierende Attributmenge für $r(R)$ ist, dann sind auch die Mengen $\{A, B\}$, $\{B, C\}$ und $\{A, B, C\}$ identifizierende Attributmengen für $r(R)$.

Ist eine Attributmenge X hingegen keine identifizierende Attributmenge für $r(R)$, so sind auch alle Teilmengen $Y \subset X$ keine identifizierenden Attributmengen für $r(R)$. Ist beispielsweise $\{A, C\}$ keine identifizierende Attributmenge für $r(R)$, so sind weder $\{A\}$ noch $\{C\}$ identifizierende Attributmengen für $r(R)$. Diese Mengen werden in [Kle98] als *negierte Schlüssel* bezeichnet:

> **Zitat:** Negierter Schlüssel [Kle98]
> Sei R ein Relationen-Schema, U die Attributmenge von R und $X \subseteq U$. Ein negierter Schlüssel (NKEY) X gilt in R, wenn es eine Relation r von R gibt, für die gilt: $\exists t, t' \in r : t \neq t'$ mit $t[X] = t'[X]$.

Für die Erkennung negierter Schlüssel wird in [Kle98] ein Algorithmus vorgestellt, der alle maximalen negierten Schlüssel bestimmt, bei denen für alle Attribute gleiche Attributwerte für mindestens ein Paar von Tupeln auftreten. Dieser Algorithmus weist eine Komplexität von $O(n^2 * m)$ auf, wobei n die Anzahl der Tupel in $r(R)$ und m die Anzahl der Attribute von R ist. Zudem wird in der Arbeit vorgeschlagen, Heuristiken, Datenbankstatistiken sowie das Wissen über vorhandene funktionale Abhängigkeiten zu nutzen, damit insbesondere kleinere Attributkombinationen mit einer geringen Zahl von verschiedenen Attributwerten nicht getestet werden müssen.

> **Beispiel:** Negierte Schlüssel anhand von Datenbankstatistiken überprüfen
> Ist für eine Attributmenge $\{A, B\} \subset R$ bekannt, dass das Produkt der Mächtigkeiten von den aktiven Domänen[a] $adom(A)$ und $adom(B)$ kleiner ist als die Tupelanzahl der Relation $r(R)$, so kann $\{A, B\}$ keine identifizierende Attributmenge für $r(R)$ sein. Die Forderung
>
> $$|adom(A)| * |adom(B)| \geq |r(R)| \tag{4.4}$$
>
> stellt somit eine notwendige – jedoch nicht hinreichende – Bedingung dar.
>
> [a] Als *aktive Domäne* eines Attributes A aus der Relation R bezeichnen wir die Menge der vorkommenden Werte von A in $r(R)$. Die aktiven Domänen der einzelnen Attribute sind bekannt, sobald diese auf ihre Schlüsseleigenschaft überprüft wurden.

Der auf Basis der oben genannten Aspekte in [Kle98] entwickelte Algorithmus erfragt durch einen Dialog mit dem Nutzer die Schlüsseleigenschaften einer Relation ab. Dabei interagiert der Nutzer mit einem Ausschnitt der Gesamtrelation und entscheidet, ob eine Attributkombination als Schlüssel festgelegt wird.

Finding Minimal Keys in a Relation Instance

In [GW99] wird ein einfaches Verfahren zur Schlüsselerkennung basierend auf bidirektionaler Breitensuche vorgeschlagen. Dafür wird die A-priori-Eigenschaft zur Reduzierung des Suchraumes angewendet. Der Suchraum wird sowohl *bottom-up*, beginnend bei einelementigen Attributmengen, als auch *top-down*, startend bei der Gesamtrelation, durchsucht. Da beide Strategien nicht sehr performant sind, wenn Schlüssel aus vielen (bottom-up) bzw. wenigen Attributen (top-down) zusammengesetzt sind, wird von Giannella und Wyss vorgeschlagen, das Wissen über bereits entdeckte (negierte) Schlüssel auszutauschen, wodurch der Suchraum in beiden Phasen weiter eingeschränkt wird. Dieses Prinzip wird in Abschnitt 4.3 bei der Vorstellung des eigenen Ansatzes zur Erkennung von Quasi-Identifikatoren näher erläutert.

A Hybrid Approach for Efficient Unique Column Combination Discovery

In [PN17] wird ein hybrides Verfahren zur Erkennung von minimalen Schlüsselkombinationen (Hybrid Unique Column Combinations – HyUCC) in großen Datensätzen vorgestellt. Durch dieses, am Hasso-Plattner-Institut in Potsdam entwickelte Verfahren, werden Techniken der Approximation mit Validierungstechniken verknüpft. Dazu wird einerseits aus der gesamten Relation eine Testmenge gebildet, für welche attributbasiert geprüft wird, welche Kombinationen nicht eindeutig sein können. Das Sampling wird auch in anderen Ansätzen, wie z. B. [SBHR06], genutzt, wird aber in Verbindung mit der Validierung in Bezug auf die Laufzeit deutlich verbessert.

In der Validierungsphase wird aus den verbleibenden Kandidaten auf Tupelebene bestimmt, ob diese wirklich eindeutig sind. Zwischen den beiden Phasen wird auf Basis eines Kostenmodells gewechselt. Liegt die Dauer der Berechnung innerhalb einer Phase über einem zuvor festgelegten Grenzwert, so wird in die jeweils andere Phase gewechselt. Mit jedem Phasenwechsel wird der Grenzwert etwas erhöht, da sonst die Berechnungszeit stets darüber liegen würde.

Scalable Discovery of Unique Column Combinations

In [HQA+13] wird ein weiteres Verfahren des HPI zur Erkennung von eindeutigen Attributkombinationen (Discovery of Unique Column Combinations, DUCC) vorgestellt. Je nach Beschaffenheit des Datensatzes ist dieser etwas ältere Algorithmus teilweise schneller als HyUCC. DUCC nutzt u. a. Greedy-Strategien, um nach Möglichkeit jeweils die Kombination zu testen, die eine minimale Schlüsselkombination sein könnte. Dafür wird der Eindeutigkeitswert jeder getesteten Menge in einer geordneten Liste abgespeichert. Der Eintrag mit dem höchsten Wert wird anschließend um ein weiteres Attribut erweitert und getestet.

Als weitere Strategie wird ein *Random Walk* vorgeschlagen. Ausgehend von einer zufälligen Attributkombination werden an diese Attribute konkateniert. Dies wird solange fortgesetzt, bis die Kombination eindeutig ist. Anschließend werden einzelne Attribute entfernt, bis die Kombination wieder nicht mehr eindeutig ist. Dieses Verfahren wird solange wiederholt, bis keine neuen Kombinationen mehr geprüft werden können. Dabei entsteht ein sogenannter Rand, an welchem sich alle minimalen Schlüsselkombinationen befinden. Im Gegensatz zur Greedy-Strategie wird durch dieses Verfahren die Wahrscheinlichkeit in der parallelen Berechnung gesenkt, dass gleiche Attributkombinationen erneut getestet werden.

4.2.2 Erkennung von Quasi-Identifikatoren

Im Vergleich zu den Verfahren zum Erkennen von Schlüsseln wurden bisher nur wenige dieser Ansätze auf Quasi-Identifikatoren übertragen. Neben dem grundlegenden Algorithmus von Motwani und Xu [MX07] existiert nach dem eigenen Erkenntnisstand nur die Arbeit von Kruse [Kru18, KN18] zur Erkennung von Quasi-Identifikatoren, wenngleich diese in den beiden letztgenannten Arbeiten als *Approximate* bzw. *Partial Unique Column Combinations* bezeichnet werden.

Efficient Algorithms for Masking and Finding Quasi-Identifiers

Motwani und Xu stellen in [MX07] einen ersten Algorithmus zur Bestimmung von minimalen Quasi-Identifikatoren vor. Der Algorithmus basiert auf einer Breitensuche im aufgespannten Suchraum über alle Attributkombinationen, wobei mit den einelementigen Attributmengen gestartet wird. Durch dieses Vorgehen wird die Minimalität der Quasi-Identifikatoren bei der Berechnung sichergestellt, da keine Obermengen von bereits gefunden Quasi-Identifikatoren betrachtet werden.

Der Algorithmus baut auf der von Mannila et. al [MTV97] vorgeschlagenen, ebenenweisen Erzeugung von Attributmengen auf. Dabei wird die Minimalitätseigenschaft von Quasi-Identifikatoren sofort erkannt und der Suchraum beim Durchlauf auf der nächsten Ebene eingeschränkt. Auf Grund dieser Eigenschaft ist der Algorithmus effizienter als das Testen der 2^n-1 möglichen Teilmengen, allerdings stellt die von Big-Data-Anwendungen erzeugte Datenmenge eine neue Herausforderung dar. Insbesondere durch die hohe Dimensionalität der Daten wird der Suchraum massiv vergrößert. Aus diesem Grund wird im folgenden Abschnitt 4.3 ein neuer Algorithmus vorgestellt, der auf dem Ansatz von Motwani und Xu aufbaut.

Efficient Discovery of Approximate Dependencies

Basierend auf Vorarbeiten am HPI wurde von Kruse [Kru18, KN18] ein Verfahren zur Bestimmung von partiellen funktionalen Abhängigkeiten und partiellen Schlüsseln entwickelt. Letztere entsprechen – aus dem Blickwinkel der Datenschutz-Welt – in ihrer Definition den Quasi-Identifikatoren.

Im Gegensatz zu den bestehenden Algorithmen zur Schlüsselsuche können Stichprobenverfahren nicht mehr in der bisherigen Form eingesetzt werden. Dies liegt daran, dass das Auftauchen *eines* Duplikates nicht zwangsweise auf die Verletzung der Quasi-Identifikator-Eigenschaft hinweist. Kruse führt dazu Fehlerabschätzungen ein, die den Anteil der Tupel, welche die Schlüsseleigenschaft verletzten, berechnen.

Der vorgeschlagene Algorithmus, Pyro, durchläuft den Suchraum zunächst *bottom-up* solange aufsteigend, bis eine Attributkombination gefunden wurde, welche einen geringen Fehler aufweist. Dabei wird, im Gegensatz zu den bisher vorgestellten Ansätzen, der Suchraum mittels Tiefensuche untersucht. Anschließend wird von dieser Position *top-down* nach unten gegangen, bis das erlaubte Fehlerausmaß wieder überschritten wird. Dadurch wird die Minimalität der gefunden Quasi-Identifikatoren sichergestellt. Die gefundenen (negierten) QIs werden, wie in den anderen vorgestellten Algorithmen, zur Einschränkung des Suchraumes verwendet. Anschließend werden die obigen Schritte erneut durchgeführt, wobei bei der Tiefensuche neue Pfade entlang des eingeschränkten Suchraumes untersucht werden.

4.2.3 Einsatz von Schlüsseln und Quasi-Identifikatoren

Quasi-Identifikatoren können, nachdem sie berechnet wurden, vielfältig eingesetzt werden. Hauptanwendung ist die Parametrisierung von Algorithmen zur Generalisierung, wie z. B. der k-Anonymität [SS98]. Des Weiteren existieren Techniken, die auf Basis von QIs bestimmen, welche Daten verschlüsselt oder fragmentiert abgespeichert werden sollen [CdVF$^+$07]. Dadurch wird einerseits die Grundlage für Secure Multi Party Computation geschaffen, andererseits können auch weiterführende Anonymisierungstechniken, wie das Slicing [LLZM12], dadurch vorbereitet werden. Eine konkrete Anwendung und Evaluation lässt sich hierzu in [JMS14] finden. In dem Artikel wird gezeigt, dass durch eine geschickte Auswahl der zu anonymisierenden Attribute, die zu den Quasi-Identifikatoren gehören, eine höhere Korrektheit der Ergebnisse von Klassifikationsalgorithmen erzielen lässt. Ausgehend von den Fragmenten lassen sich zudem Sichten generieren, die durch attributbasierte Kontrollmechanismen vor unbefugten Zugriff abgesichert werden können [TT14]. Durch den in [CdVF$^+$07] vorgestellten Algorithmus zur minimalen Fragmentierung ist es zudem zu gleichen Teilen möglich, Privatsphäre und Datennutzen zu bewahren.

Der in Kapitel 6 entwickelte Ansatz zur Realisierung von Datensparsamkeit nutzt intern ebenfalls Quasi-Identifikatoren, um beispielsweise zu entscheiden, welche Projektionen oder Selektionen möglichst früh ausgeführt werden sollten. Zudem werden die im PArADISE-Framework hinterlegten Datenschutzeinstellungen (siehe Abschnitt 3.2) mit den berechneten Quasi-Identifikatoren verglichen.

Quasi-Identifikatoren dürfen, wenn Daten nicht anonymisiert werden, nicht als Ganzes in der Projektionsliste einer Anfrage auftauchen. Es muss daher sichergestellt werden, dass die Attributmenge jedes minimalen Quasi-Identifikators um mindestens ein Attribut reduziert wird.

Im trivialen Fall existiert für eine Relation R nur ein Quasi-Identifikator mit der Attributmenge $Q \subseteq R$. Gilt für die Projektionsliste X, dass $Q \not\subseteq X$, dann ist Q nicht vollständig in X enthalten und X muss somit nicht eingeschränkt werden. Gilt hingegen $Q \subseteq X$, reicht es aus, genau ein Attribut $A \in Q$ aus X zu entfernen, wodurch Q nicht mehr vollständig in X vorhanden ist.

Die Auswahl von A ist dabei von der Semantik der Anfrage abhängig. Beispielsweise kann jedem Attribut $A_i \in R$ ein Gewicht $w_i; 0 \leq w_i \leq 1$ zugeordnet werden, welches den Wert des Attributes für die weitere Anfrage darstellt. Entsprechend kann das Attribut mit dem geringsten Wert wegprojiziert werden. Dieses Vorgehen eignet sich in der Regel nur für die abschließende Projektion vor der Ausgabe des Anfrageergebnisses, da in der Regel viele Attribute für weitere Selektionen und Aggregate benötigt werden. Alternativ dazu lässt sich eine Projektion mit anderen Algebraoperatoren vertauschen bzw. in mehrere Projektionen aufteilen. Details dazu werden in Abschnitt 6.3 des übernächsten Kapitels gegeben.

Im nächsten Abschnitt wird das im Rahmen von PArADISE entwickelte Verfahren zur Bestimmung von Quasi-Identifikatoren vorgestellt. Die Anwendung des Verfahrens auf verschiedene Anwendungsfälle zeigt auf, dass bereits bei recht überschaubaren Attributmengen Hunderte von verschiedenen, teils überlappenden, minimalen Quasi-Identifikatoren existieren. Um sicherzustellen, dass kein Quasi-Identifikator in der Projektionsliste vollständig vorhanden ist, muss entsprechend der obigen Forderung mindestens ein Attribut aus jedem QI entfernt bzw. dessen Attributwerte generalisiert, verrauscht oder permutiert werden – was bei vielen überlappenden QIs zu einem hohen Informationsverlust führt. Aus diesem Grund eignen sich Anonymisierungsverfahren nur bedingt für hochdimensionale Daten. Das in Kapitel 6 vorgestellte Verfahren verhindert durch die lokale Vorberechnung der Daten eine unnötige Anonymisierung, da durch Vorfiltern, -projizieren und -aggregieren Quasi-Identifikatoren aus dem Datenbestand wegfallen, jedoch kein Informationsverlust auftritt.

In den folgenden Abschnitten wird das für PArADISE entwickelte Verfahren zur Erkennung von Quasi-Identifikatoren beschrieben. Basierend auf dem Stand der Technik wurde zunächst untersucht, inwiefern die bisherigen Ansätze für den Einsatz in Assistenzsystemen geeignet sind. Ausgehend von den Ergebnissen wurde ein eigener Ansatz entwickelt, der insbesondere für die Auswertung von Sensordaten geeignet ist.

4.3 Eigener Ansatz

Viele Verfahren für unterschiedliche Problemklassen skalieren sehr schlecht, wenn die Dimensionalität des zu lösenden Problems steigt. Um diesen *Fluch der Dimensionalität* [Bel15] entgegenzuwirken, werden häufig Verfahren zur Reduzierung der Anzahl an Dimensionen eingesetzt. Im Bereich der Schlüsselsuche sind die Dimensionen die einzelnen Attribute in der Relation. Wie dem Stand der Forschung zu entnehmen ist, werden die Attribute in einer hierarchischen Struktur angeordnet und der Reihe nach durch uni- bzw. bidirektionale Breitensuche oder mittels Tiefensuche durchsucht. Dabei werden Teilbäume, die keinen Schlüssel zurückliefern können, eliminiert, wodurch die Anzahl der zu durchsuchenden Kombinationen unter Umständen stark reduziert werden kann. Irrelevante Attributkombinationen können durch einfache Tests früh erkannt und von den verbleibenden Prüfungen ausgeschlossen werden.

Neben der Anzahl von Attributen hängt die Laufzeit der Schlüsselsuche auch stark von der betroffenen Datenmenge – sprich der Anzahl der zu ladenden Tupel – ab. Wenn die Datenmenge extrem groß wird, ist ein Laden des gesamten Datensatzes in den Hauptspeicher meist nicht gewünscht, da das Auslesen der Daten vom Festplattenspeicher einen Flaschenhals[1] darstellt. Um dem entgegenzuwirken, verwenden viele Algorithmen Verfahren zum Sampling des Datensatzes. Wird in einer Teilmenge der Daten bereits festgestellt, dass Duplikate auftreten, dann kann eine bestimmte Attributmenge auch nicht Schlüssel für den gesamten Datenbestand sein.

Für die Suche nach Quasi-Identifikatoren wird im vorgestellten Ansatz ebenfalls ein Verfahren für die Begrenzung des Suchraumes eingesetzt. Auf eine Einschränkung in der Tupelmenge durch Sampling wird allerdings

[1] Die typische Zugriffszeit einer Magnetplatte liegt mit 10 ms in etwa um den Faktor 1000 über der Zugriffszeit des Arbeitsspeichers mit 10 ns. Moderne SSD-Festplatten verringern diese Lücke, schließen sie aber nicht ganz [Tan09].

verzichtet. Zwar kann das Sampling-Verfahren auch so ergänzt werden, dass es eine Attributkombination erst ausschließt, wenn eine bestimmte Anzahl an Duplikaten erreicht wurde. Dies ist jedoch in vielen Fällen nicht zielführend, da, wie in Abschnitt 4.5 experimentell nachgewiesen wird, in realen Anwendungsfällen der Unterschied zwischen Tupeln meist zu gering ist, um in einer Teilmenge ausreichend Duplikate zu erkennen.

Das entwickelte Konzept beruht auf drei Prinzipien: Bidirektionale Breitensuche, Wichtung basierend auf der Größe der Attributkombinationen sowie der dynamischen Berechnung der Distinct Ratio bei Änderungen am Datenbestand. Diese Prinzipien werden im Folgenden detailliert vorgestellt.

4.3.1 Berechnung der Distinct Ratio

Grundlage für die Entscheidung, ob eine Attributmenge ein Quasi-Identifikator ist, stellt die Überprüfung mittels der Distinct bzw. Separation Ratio dar. Für den eigenen Ansatz wird im Folgenden die Distinct Ratio verwendet, da diese einfacher zu berechnen ist. Während die Separation Ratio (siehe Gleichung 4.2) aus einer arithmetischen Division, einer Projektion, einer Selektion, zwei Aggregationen sowie einer Gruppierung besteht, entfallen bei der Distinct Ratio (siehe Gleichung 4.2) die Selektion und die Gruppierung. Dies stellt eine signifikante Reduzierung der Laufzeit dar.

Der Unterschied zwischen beiden Metriken liegt im Umgang mit den Duplikaten. Während bei der Distinct Ratio mehrere Duplikate mit dem Wert `1` gezählt werden, beträgt dieser bei der Separation Ratio `0`. Dadurch liefert die Separation Ratio stets den gleichen[2] bzw. einen niedrigeren Wert zurück als die Distinct Ratio. Im PArADISE-Framework sind beide Metriken verfügbar; da sie ähnliche Ergebnisse zurückliefern, können sie je nach konkretem Anwendungsfall bzw. den eigenen Bedürfnissen ausgetauscht werden. Auf die Implementierung der Berechnung wird näher in Abschnitt 4.4 eingegangen.

4.3.2 Gewichtete bidirektionale Breitensuche

Den Hauptfaktor zur effizienten Bestimmung der Quasi-Identifikatoren stellt die Verwendung der bidirektionalen Breitensuche dar. Die dabei verwendeten Suchstrategien, Bottom-Up und Top-Down, werden im Folgenden mit ihren Vor- und Nachteilen kurz eingeführt. Die durch Kombination beider Techniken resultierende Suchstrategie wird abschließend durch eine Wichtungsfunktion zur weiteren Reduzierung der Laufzeit ergänzt.

Bottom-Up

Der in [MX07] von Motwani und Xu entwickelte Algorithmus zur Erkennung aller minimalen Quasi-Identifikatoren basiert auf einem Bottom-Up-Verfahren. Für den eigenen Ansatz wird dieses Verfahren ebenfalls verwendet. Zusätzlich zu der ebenenweisen Überprüfung einer Relation mit n Attributen wird zusätzlich im ersten Durchlauf mit allen einelementigen Attributmengen vermerkt, ob für ein Attribut nur ein einziger Attributwert vorhanden ist. Ist dies der Fall, so kann das Attribut für die weiteren Berechnungen ignoriert werden, da es keinen Einfluss auf die Distinct bzw. Separation Ratio besitzt.

Anschließend wird die Relation ebenenweise von den n einelementigen Attributkombinationen zu der n-elementigen Attributkombination auf Quasi-Identifikatoren durchsucht. Wird dabei für eine k-elementige ($1 \leq k < n$) Attributkombination *X* festgestellt, dass diese ein Quasi-Identifikator für den gegebenen Grenzwert t ist, so werden alle Attributkombinationen Y, für die $X \supset Y$ gilt, nicht weiter berücksichtigt. Somit werden nur die minimalen Quasi-Identifikatoren bestimmt. Dies ist gewollt, da insbesondere für Parametrisierung von Anonymisierungsfunktionen minimale QIs benötigt werden, da es ansonsten zu einem ungewollten erhöhten Informationsverlust durch übermäßige Anonymisierung kommen kann.

Die Bottom-Up-Methodik ist in Algorithmus 1 als Pseudocode skizziert. Im ersten Schritt erfolgt eine Initialisierung der verwendeten Variablen. Dabei wird zudem die zu untersuchende Relation r einmalig gescannt, um festzustellen, wie viele Tupel vorhanden sind. Anschließend werden innerhalb der ersten Schleife alle einelementigen Attributmengen gebildet. Für jede Attributmenge $testSet$ wird im ersten Durchlauf der zweiten Schleife überprüft, wie viele unterschiedliche bzw. einzigartige Tupel in der Relation vorhanden sind (Methode $getPercentage$).

[2] Beide Metriken liefern den gleichen Wert, $p = 1$, wenn keine Duplikate im Datensatz enthalten sind.

Algorithmus 1: BottomUp-Phase

Data: Relation *r = r(R)* mit Menge von Attributen *attributes*, Grenzwert *threshold*
Result: eine Menge minimaler QI *qiLowerSet*
$int\ num = getNumTuples(r)$;
$Set < Set >\ set = new\ Set < Set > ()$;
foreach $attr \in attributes$ **do**
  $set := set \cup \{attr\}$;
end
while $set \neq \emptyset$ **do**
  foreach $testSet \in set$ **do**
    $double\ p := getPercentage(testSet, num)$;
    if $p \geq threshold$ **then**
      $qiLowerSet := qiLowerSet \cup \{testSet\}$;
    end
  end
  $set := buildNewLowerSet(set, attributes)$;
end
return $qiLowerSet$;

Algorithmus 2: Erstellung der neuen unteren Testmenge

Data: Aktuelle untere Testmenge *lSet*, Liste der Attribute in R *attributes*
Result: Neue untere Testmenge *lSetNew*
$Set\ lSetNew := new\ Set()$;
foreach $set \in lSet$ **do**
  foreach $A \in attributes$ **do**
    if $(\nexists q \in qiLowerSet : q \subseteq set \cup \{A\}) \wedge (A \notin set)$ **then**
      $lSetNew := lSetNew \cup \{set \cup \{A\}\}$;
    end
  end
end
return $lSetNew$;

Daraus wird das Verhältnis p zur Gesamtzahl aller Tupel in $r(R)$ gebildet. Liegt der Anteil über dem festgelegten Grenzwert (engl.: *threshold*), so ist diese Attributmenge ein minimaler Quasi-Identifikator und wird in die Menge aller QIs, *qiLowerSet*, aufgenommen.

Wurden alle Attributkombinationen des aktuellen Schleifendurchlaufs überprüft, ermittelt der abschließend aufgerufene Algorithmus $buildNewLowerSet$ (siehe Algorithmus 2) die k+1-elementigen Attributkombinationen unter Berücksichtigung der bisher ermittelten minimalen QIs. Die zweite Schleife wird solange ausgeführt, bis alle potentiellen Attributkombinationen überprüft worden sind. Dies ist der Fall, wenn (a) die n-te Ebene erreicht und überprüft wurde oder (b) die Bestimmung der k+1-elementigen Attributkombinationen eine leere Menge zurückliefert. Bedingung (a) tritt dabei nur ein, wenn keine Quasi-Identifikatoren in $r(R)$ enthalten sind. Abbildung 4.2 zeigt den Ablauf des Algorithmus anhand eines Beispiels mit fünf Attributen.

Vor- und Nachteile des Bottom-Up-Verfahrens: Der Algorithmus arbeitet insbesondere dann sehr effizient, wenn die Relation $r(R)$ viele, aus wenigen Attributen zusammengesetzte QIs enthält. Dadurch wird der Suchraum relativ früh eingeschränkt, wodurch die Anzahl der benötigten Tabellensuchläufe reduziert wird. Zudem lassen sich kleinere Attributkombinationen, die aus wenigen Attributen zusammengesetzt sind, leichter überprüfen als Attri-

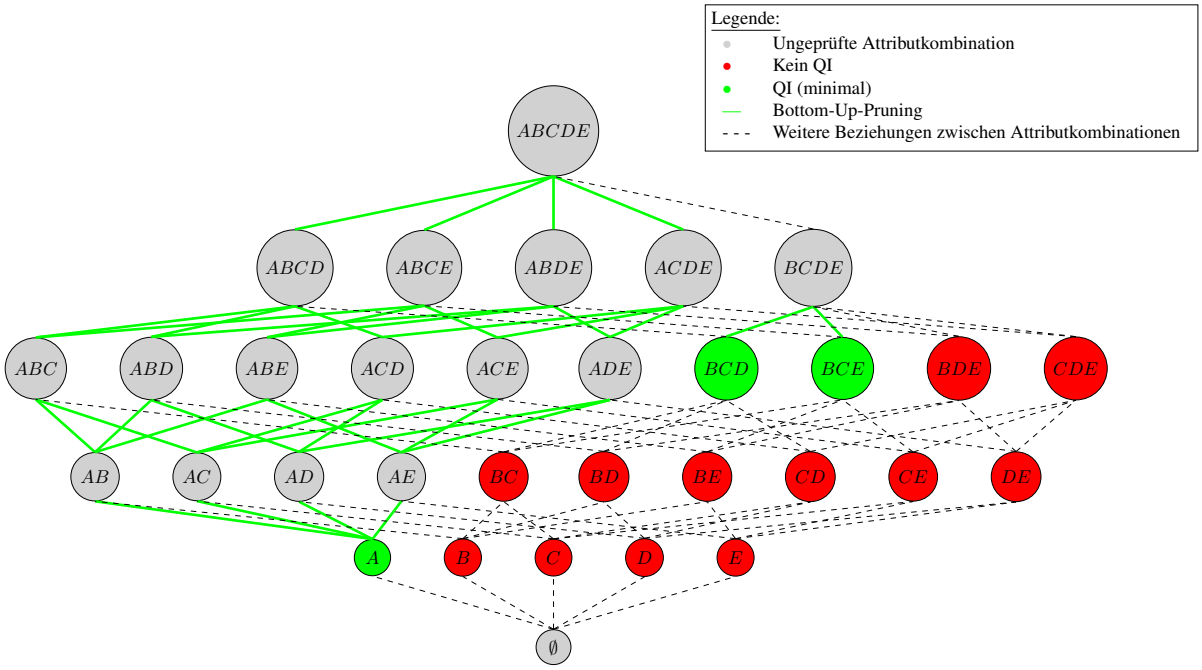


Abbildung 4.2: Die Abbildung zeigt den Ablauf des Bottom-Up-Algorithmus anhand eines einfachen Beispiels. Im ersten Bottom-Up-Durchlauf werden die atomaren Attribute $\{A\}$, $\{B\}$, $\{C\}$, $\{D\}$ und $\{E\}$ überprüft. Im nächsten Schritt werden alle zwei-elementigen Attributkombinationen überprüft, welche $\{A\}$ nicht als Teilmenge enthalten, da diese als Quasi-Identifikator erkannt wurde. Es verbleiben somit die Mengen $\{B, C\}$, $\{B, D\}$, $\{B, E\}$,$\{C, D\}$, $\{C, E\}$ und $\{D, E\}$. Da keine dieser Mengen ein Quasi-Identifikator ist, wird der Suchraum nicht weiter eingeschränkt. Bei den drei-elementigen Attributmengen werden anschließend $\{B, C, D\}$ und $\{B, D, E\}$ als QIs erkannt. In der Folge können bei den vier-elementigen Attributkombinationen durch den QI $\{A\}$ vier Attributkombinationen ausgeschlossen werden, durch $\{B, C, D\}$ bzw. $\{B, D, E\}$ entfällt auch die fünfte. Da auf dieser Ebene keine Attributkombinationen mehr überprüft werden können, terminiert der Algorithmus.

---

butkombinationen mit vielen Attributen. Auf diesen Aspekt wird im Abschnitt 4.3.2 (Wichtung der bidirektionalen Breitensuche) eingegangen.

Der Nachteil des Algorithmus zeigt sich, wenn die Relation $r(R)$ viele QIs umfasst, welche aus vielen Attributen zusammengesetzt sind. Dadurch wird der Suchraum erst zum Ende des Bottom-Up-Algorithmus eingeschränkt; die Anzahl der zu betrachtenden Attributmengen wird nicht wahrnehmbar reduziert. Sofern $r(R)$ keine QIs enthält, so wird dies durch das Bottom-Up-Verfahren erst nach Überprüfung aller 2^n-1 möglichen Attributkombinationen festgestellt. Das Verhalten entspricht in diesem Extremfall einem Brute-Force-Verfahren.

Top-Down

Für die Erklärung des Algorithmus 5 zur gewichteten bidirektionalen Breitensuche wird zusätzlich zum Bottom-Up-Vorgehen das entgegengesetzte Top-Down-Vorgehen benötigt. Dieses Verfahren setzt auf die in [Kle98] vorgeschlagenen negierten Schlüssel auf. Analog zu den negierten Schlüsseln gilt, dass eine Teilmenge T kein Quasi-Identifikator ist, wenn mindestens eine Attributkombination existiert, die kein QI ist und T als Teilmenge enthält.

Eine Beschreibung der Top-Down-Methodik ist in Algorithmus 3 angegeben. Die Überprüfung der Attributkombinationen erfolgt wie beim Bottom-Up-Verfahren ebenenweise, jedoch in umgekehrter Reihenfolge. Für eine Relation mit n Attributen wird zunächst die n-elementige Teilmenge, set, gebildet und mittels der Methode $getPercentage$ überprüft. Liegt der berechnete Anteil p unterhalb des festgelegten Grenzwertes t (engl.: *thres-*

hold), so ist diese Attributmenge kein Quasi-Identifikator und wird in die Menge der bekannten negierten QIs, *optOutSet*, aufgenommen, andernfalls in die Menge der bekannten Quasi-Identifikatoren, $qiUpperSet$.

Algorithmus 3: Top-Down-Phase

```
Data: Relation r(R) mit Menge von Attributen attrs, Grenzwert threshold
Result: Eine Menge minimaler Quasi-Identifikatoren qiSet
Setset := {attrs};
Set optOutSet := new Set();
Set qiUpperSet := new Set();
while set ≠ emptyset do
    foreach testSet ∈ set do
        double p := getPercentage(testSet, r);
        if p < threshold then
            optOutSet := optOutSet ∪ {subset};
        else
            qiUpperSet := qiUpperSet ∪ {testSet};
            foreach o ∈ qiSet do
                if testSet ⊂ o then
                    qiUpperSet := qiUpperSet − {o};
                end
            end
        end
    end
    set := buildNewUpper(set);
end
return qiUpperSet;
```

Sofern nicht alle Attributkombinationen aus set ein negierter Quasi-Identifikator sind, werden anschließend alle (n-1)-elementigen Teilmengen aus den bestehenden Mengen in set gebildet. Die Bildung der Teilmengen ist in Algorithmus 4 skizziert. Dieses Vorgehen wird solange fortgesetzt, bis alle möglichen Attributmengen überprüft wurden. Die gefundenen Quasi-Identifikatoren werden in *qiUpperSet* gespeichert.

Durch das Top-Down-Vorgehen wird, im Gegensatz zum Bottom-Up-Algorithmus, die Minimalität der Quasi-Identifikatoren nicht gewährleistet. Die Minimalität lässt sich durch einen zusätzlichen Überprüfungsschritt erreichen: Wenn auf der Ebene k ein Quasi-Identifikator gefunden wird, werden die dazugehörigen QIs in der Ebene mit k+1 Attributen aus $qiUpperSet$ gestrichen. Im Algorithmus 3 wird dies in der innersten Schleife umgesetzt. Abbildung 4.3 zeigt den Ablauf des Algorithmus anhand des bereits für den Bottom-Up-Algorithmus benutzten Beispiels.

Vor- und Nachteile des Top-Down-Verfahrens: Im Top-Down-Ansatz treten die Nachteile aus dem zuvor vorgestellten Bottom-Up-Verfahren nicht auf: Sind die Daten in der zu untersuchenden Relation $r(R)$ derart verteilt, dass die meisten Quasi-Identifikatoren aus vielen Attributen zusammengesetzt sind bzw. viele negierte QIs existieren, wird der Suchraum früh eingeschränkt. Dadurch wird die Laufzeit der QI-Suche mittels des Top-Down-Ansatzes deutlich reduziert. Im Extremfall, wenn die Kombination aller Attribute aus R für $r(R)$ keinen QI darstellen, terminiert der Algorithmus nach dem ersten vollständigen Scan der Relation.

Eine ungünstige Auswirkung auf die Laufzeit besteht hingegen beim Top-Down-Verfahren, wenn $r(R)$ aus vielen kleinen Quasi-Identifikatoren besteht, da der Suchraum erst spät eingeschränkt werden kann. Ein weiterer Nachteil besteht in einer drastischen Steigerung der benötigten Rechenzeit. Der Grund hierfür ist die gesteigerte Menge an Daten, die vor Einschränkung des Suchraumes eingelesen werden. Beim Bottom-Up-Verfahren werden zunächst alle Attributkombinationen mit einem Attribut eingelesen, bevor die Kombinationen mit mehreren Attributen überprüft werden. Die Überprüfung dieser einelementigen Mengen benötigt eine geringere Rechenzeit,

Algorithmus 4: Erstellung der neuen oberen Testmenge

Data: Aktuelle obere Testmenge **uSet**
Result: Neue obere Testmenge **uSetNew**
$Set\ uSetNew := new\ Set();$
foreach $set \in uSet$ **do**
  foreach $A \in set$ **do**
    if $\nexists o \in optOutSet : set \backslash \{A\} \subseteq o$ **then**
      $uSetNew := uSetNew \cup \{set \backslash \{A\}\};$
    end
  end
end
return $uSetNew;$

da weniger Daten von der Festplatte in den Arbeitsspeicher geladen werden müssen. Zudem ist der Duplikatetest auf diesen Mengen mit einem geringeren Aufwand verbunden. Zwar müssen auch beim Bottom-Up-Verfahren alle Attributkombinationen mit mehreren Attributen geprüft werden, jedoch können durch die Einschränkung des Suchraumes viele der Kombinationen entfallen. Durch den Top-Down-Algorithmus werden vielmehr die kleineren anstatt die größeren Attributkombinationen aus dem Suchraum entfernt. Dadurch ergibt sich ein nicht unerheblicher Unterschied zwischen den Laufzeiten (siehe Tabelle 4.2 und Abbildung 4.5). Um dies zu vermeiden, sollte der Top-Down-Algorithmus nur begrenzt eingesetzt werden. Eine Möglichkeit zum besonnenen Einsatz besteht in der bidirektionalen Breitensuche.

Bidirektionale Breitensuche

Einige der im vorherigen Abschnitt 4.2 eingeführten Verfahren zur Bestimmung von Schlüsseln kombinieren den Bottom-Up- mit dem Top-Down-Ansatz. Dies lässt sich auch auf die Ermittlung der Quasi-Identifikatoren übertragen. Dazu werden die beiden Verfahren alternierend angewendet und die Kenntnis über (negierte) QIs an das jeweils andere Verfahren übergeben. Der Bottom-Up-Algorithmus startet, wie eingeführt, mit den einelementigen Attributkombinationen, der Top-Down-Algorithmus hingegen mit der Attributkombination, welche alle Attribute beinhaltet. Pro Durchlauf werden die Attributkombinationen mit den aktuell wenigsten bzw. meisten Attributen überprüft. Der Algorithmus terminiert, wenn eine der folgenden beiden Bedingungen erfüllt ist:

1. Alle Ebenen von Attributkombinationen wurden entweder durch den Bottom-Up- oder Top-Down-Algorithmus abgearbeitet oder
2. die nächste Bottom-Up-Phase hat keine weiteren Attributkombinationen zu überprüfen.

Die erste Bedingung tritt ein, sobald alle Attributkombinationen abgearbeitet wurden. Dieses Kriterium ist äquivalent zum Bottom-Up- bzw. zum Top-Down-Algorithmus. Die zweite Bedingung tritt in den Fällen auf, wenn in der vorherigen Top-Down-Phase ausschließlich negierte Quasi-Identifikatoren gefunden wurden. Dadurch ist jede Teilmenge ebenfalls ein negierter QI, wodurch keine weiteren Überprüfungen notwendig sind. In Abbildung 4.3 ist die Arbeitsweise des Algorithmus anhand der bekannten Beispielrelation mit fünf Attributen dargestellt.

Gewichtung der Attributanzahl und -typen

Als Erweiterung zu den bestehenden Verfahren aus dem Stand der Technik wurde für die bidirektionale Breitensuche eine Wichtungsfunktion entwickelt, die entscheidet, ob im nächsten Schritt das Bottom-Up- oder Top-Down-Verfahren verwendet werden soll. Die Idee hinter der Wichtungsfunktion ist es, aus den zwei Mengen von zu überprüfenden Attributkombinationen diejenige auszuwählen, welche einen geringeren Aufwand hinsichtlich der Rechenzeit verursacht. Dieser Aufwand ergibt sich aus zwei Komponenten:

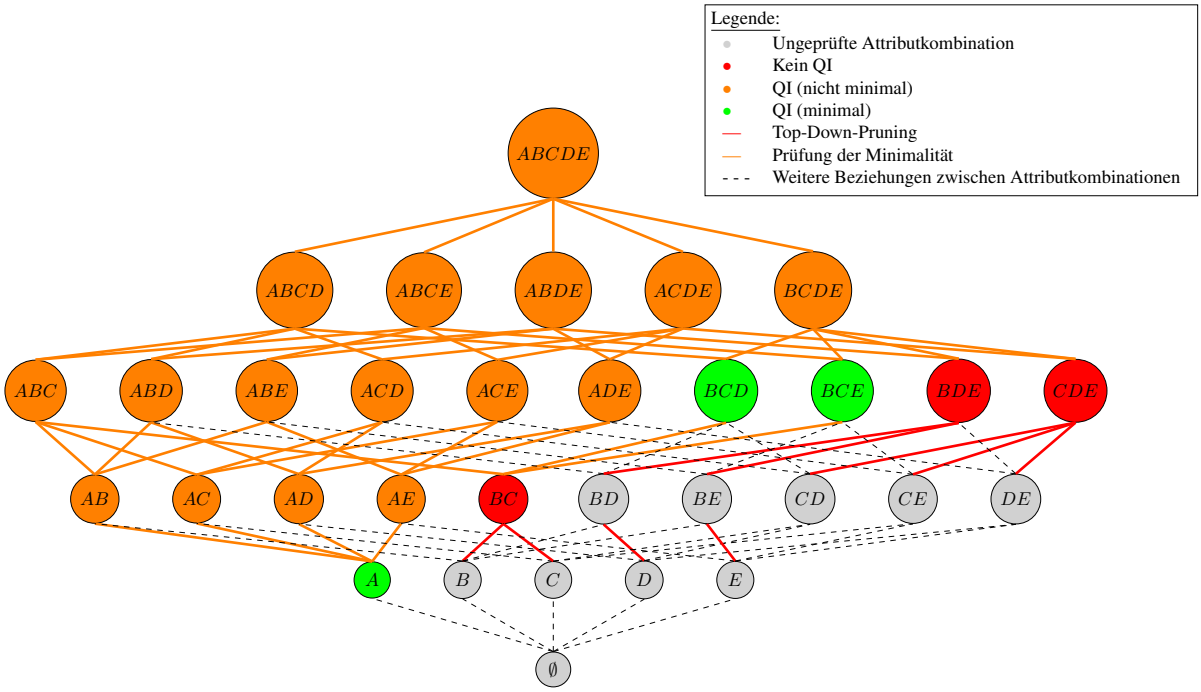


Abbildung 4.3: Die Abbildung zeigt den Ablauf des Top-Down-Algorithmus anhand eines einfachen Beispiels. Im ersten Top-Down-Durchlauf wird die Attributkombination $\{A, B, C, D, E\}$ überprüft. Diese ist ein Quasi-Identifikator, jedoch ist die Minimalität durch das Top-Down-Vorgehen noch nicht sichergestellt. Im nächsten Schritt werden alle vier-elementigen Attributkombinationen, $\{A, B, C, D\}$, $\{A, B, C, E\}$, $\{A, B, D, E\}$, $\{A, C, D, E\}$ und $\{B, C, D, E\}$, überprüft. Diese werden ebenfalls als Quasi-Identifikator erkannt, wodurch $\{A, B, C, D, E\}$ kein minimaler QI sein kann. Anschließend werden die dreielementigen Attributkombinationen überprüft. Dabei erfüllen nur die Mengen $\{B, D, E\}$ und $\{C, D, E\}$ nicht die Anforderungen an einen Quasi-Identifikator. Durch die Beschneidung des Suchraumes können somit die Kombinationen $\{B, D\}$, $\{B, E\}$, $\{C, D\}$, $\{C, E\}$ und $\{D, E\}$ sowie deren Teilmengen ausgeschlossen werden. Zudem können die vier-elementigen Attributkombinationen $\{B\}$, $\{C\}$, $\{D\}$ und $\{E\}$ als Quasi-Identifikatoren ausgeschlossen werden. Von den verbleibenden fünf zwei-elementigen Attributkombinationen erfüllt $\{B, C\}$ nicht die QI-Eigenschaft. Dadurch wird erkannt, dass $\{B, C, D\}$ und $\{B, C, E\}$ minimale Quasi-Identifikatoren sein müssen. Bei den einelementigen Attributkombinationen verbleibt nur $\{A\}$ zur Überprüfung – dies ist der letzte minimale QI und der Algorithmus terminiert.

---

1. Die Anzahl der verbleibenden, zu überprüfenden Attributkombinationen in der jeweiligen Ebene und
2. der „Größe" der konkreten Attributkombination.

Die Überprüfung einer einzelnen Attributkombination zur Erkennung von Duplikaten hat eine Laufzeit von $O(n * log_2(n))$, wobei n die Anzahl der Tupel in der Relation ist[3] und keine vorherige Sortierung der Attributwerte erfolgte. Je mehr Attributkombinationen zu überprüfen sind, desto länger dauert die Überprüfung der aktuellen Ebene. Jedoch ist zu beachten, dass, bezogen auf das laufende Beispiel, die Überprüfung der fünf einelementigen Attributmengen weniger Rechenzeit in Anspruch nimmt als das Testen von fünf vier-elementigen Attributmengen. Dies ist darauf zurückzuführen, dass die Überprüfung der Tupel zusätzlich von der „Größe" der jeweiligen Attributkombination abhängt.

[3]Im günstigsten Fall liegen alle Tupel aus $r(R)$ in einer Gruppe. In diesem Fall beträgt die Laufzeit $O(n)$. Im ungünstigsten Fall bildet jedes Tupel aus $r(R)$ eine eigene Gruppe, wodurch die Komplexität bei $O(n^2)$ liegt.

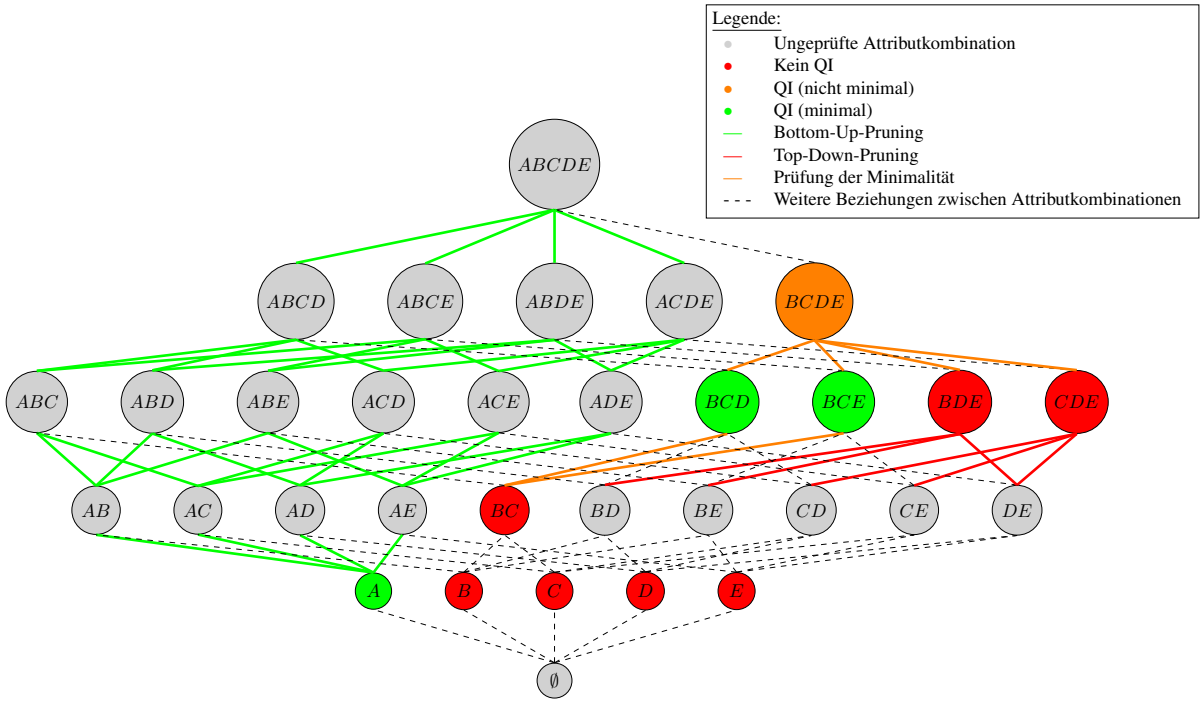


Abbildung 4.4: Die Abbildung zeigt den Ablauf der gewichteten bidirektionalen Breitensuche anhand eines einfachen Beispiels. Im ersten Bottom-Up-Durchlauf werden die atomaren Attribute $\{A\}$, $\{B\}$, $\{C\}$, $\{D\}$ und $\{E\}$ überprüft. Dabei wird festgestellt, dass $\{A\}$ ein Quasi-Identifikator ist. Für die anschließende Top-Down-Phase kann die Attributkombination $\{A, B, C, D, E\}$ auf der obersten Ebene sowie die Attributkombinationen $\{A, B, C, D\}$, $\{A, B, C, E\}$, $\{A, B, D, E\}$ und $\{A, C, D, E\}$ auf der nächsttieferen Ebene ausgelassen werden, da sie $\{A\}$ als Teilmenge enthalten. Es muss demnach nur $\{B, C, D, E\}$ getestet werden. Diese Teilmenge erfüllt im Beispiel die QI-Eigenschaft, jedoch ist in der Top-Down-Phase die Minimalität nicht sichergestellt. Anschließend erfolgt eine erneute Top-Down-Suche mit den verbleibenden Attributkombinationen $\{B, C, D\}$ (möglicher minimaler QI), $\{B, C, E\}$ (möglicher minimaler QI), $\{D, B, E\}$ (kein QI) und $\{C, D, E\}$ (kein QI). Da $\{B, C\}$ die einzige gemeinsame 2-er-Attributkombination aus den QIs der letzten Top-Down-Phase ist, muss nur noch diese abschließend geprüft werden. Da $\{B, C\}$ nicht als QI erkannt wird, müssen $\{B, C, D\}$ und $\{B, C, E\}$ minimale Quasi-Identifikatoren sein. Da nach der letzten Phase alle Ebenen untersucht wurden, terminiert der Algorithmus.

Beispiel: Vergleich der drei Suchstrategien am laufenden Beispiel

Die nachfolgende Tabelle zeigt für die drei vorgestellten Suchstrategien die Anzahl der untersuchten Attributkombinationen (Spalte *Anzahl Vergleiche*) sowie die Anzahl der pro Ebene zu ladenden Attribute als Kosten. Diese ergibt sich aus der Anzahl der Attributkombinationen multipliziert mit der Attributanzahl auf der jeweiligen Ebene.

Algorithmus	**Kosten**						**Anzahl Vergleiche**
	Ebene 1	Ebene 2	Ebene 3	Ebene 4	Ebene 5	**Gesamt**	
Bottom-Up	5*1=5	6*2=12	4*3=12	0	0	29	15
Top-Down	1*1=1	5*2=10	10*3=30	5*4=20	1*5=5	66	22
Bidirektional	5*1=5	1*2=2	4*3=12	1*4=4	0	23	11

Die Zahlen beziehen sich auf das Beispiel aus Abbildung 4.4. Während der Bottom-Up-Algorithmus insge-

samt 29 Attribute in 15 Vergleichen überprüfen muss, benötigt das Top-Down-Vorgehen 22 Vergleiche bei 66 zu überprüfenden Attributen. Die Kombination beider Ansätze als bidirektionale Breitensuche vergleicht elf Attributkombinationen mit lediglich 23 Attributen.

Besteht ein zu überprüfendes Tupel aus vielen Attributen, so muss das Datenbanksystem für die Duplikaterkennung mehr Daten von der Festplatte in den Arbeitsspeicher laden – an dieser Stelle liegt das typische Flaschenhalsproblem [Tan09] vor. Durch die größere Menge an Daten werden bereits geladene Seiten aus dem Arbeitsspeicher verdrängt, obwohl sie für spätere Berechnungen benötigt werden. Dies führt zu einer signifikanten Steigerung der Rechenzeit.

Einbindung der Wichtungsfunktion: Algorithmus 5 zeigt die Einbindung einer (zu diesem Zeitpunkt noch abstrakten) Wichtungsfunktion in einen Algorithmus für die bidirektionale Breitensuche. Den Kern bildet die `while`-Schleife. Diese überprüft zunächst, ob die zuvor eingeführten Abbruchbedingungen erfüllt sind. Solange dies nicht der Fall ist, wird mittels der Wichtungsfunktion `calculateWeights` entschieden, ob im nächsten Schritt der Bottom-Up- (`if`-Zweig) oder der Top-Down-Ansatz (`else`-Zweig) genutzt wird.

Algorithmus 5: Gewichtete bidirektionale Breitensuche

```
Data: Relation r mit Attributen attrList
Result: Eine Menge minimaler Quasi-Identifikatoren qiSet
// Attribute mit nur einem Attributwert entfernen
attrList := attrList.removeConstantAttributes();
// Initiale obere und untere Testmenge erstellen
Set upperSet := new Set({attrList});
Set lowerSet := new Set(attrList);
int bottom := 1;
int top := attrList.size();
while (bottom <= top) ∧ (lowerSet ≠ ∅) do
    isLowerSetNext := calculateWeights();
    if isLowerSetNext then
        // Bottom-Up einmal ausführen, Ebene bottom+1 erstellen, neue QIs
           aus der aktuellen oberen Ebene entfernen
        bottomUp(r, bottom);
        buildNewLowerSet();
        bottom++;
        modifyUpperSet();
    else
        // Top-Down einmal ausführen, Ebene top-1 erstellen, neue
           negierte QIs aus der aktuellen unteren Ebene entfernen
        topDown(r, top);
        buildNewUpperSet();
        top--;
        modifyLowerSet();
    end
end
qiSet := qiLowerSet ∪ qiUpperSet;
return qiSet;
```

Im `if`-Zweig wird zunächst der Bottom-Up-Algorithmus auf der aktuellen Ebene `bottom` ausgeführt und die dort vorkommenden minimalen Quasi-Identifikatoren bestimmt. Anschließend werden die neuen Attributkombinationen mit *bottom*+1 Attributen in der Methode `buildNewLowerSet` gebildet. Dabei werden alle Attri-

butkombinationen aussortiert, welche eine Obermenge der bekannten minimalen Quasi-Identifikatoren sind. Die minimalen QIs werden in der Methode `modifyUpperSet` zudem genutzt, um die aktuell zu testenden Attributkombinationen für den Top-Down-Algorithmus einzuschränken. Der `else`-Zweig wird äquivalent abgearbeitet: Top-Down-Algorithmus ausführen, neue obere Testmenge auf Basis der (nicht zwangsweisen minimalen) neuen Quasi-Identifikatoren bilden (Methode `buildNewUpperSet`) und auf Basis der neu ermittelten negierten QIs die Attributkombinationen für den Bottom-Up-Algorithmus einschränken (Methode `modifyLowerSet`)[4].

Nach jedem Schleifendurchlauf haben sich sowohl die untere als auch obere Menge von Attributkombinationen geändert. Die Wichtungsfunktion `calculateWeights` berechnet somit neue Werte und wählt für den nächsten Schleifendurchlauf die günstigere Suchstrategie aus. Je nachdem, wie stark der Suchraum in einer der beiden Suchrichtungen eingeschränkt wird, kann die gleiche Suchstrategie mehrfach hintereinander angewendet werden. Dies ist insbesondere dann der Fall, wenn durch die Kenntnis über die minimalen Quasi-Identifikatoren fast alle Attributmengen für den Top-Down-Algorithmus entfallen, da diese Obermengen der QIs darstellen[5].

Spezifizierung von Wichtungsfunktionen: Zur Definition einer vereinfachten Wichtungsfunktion gehen wir zunächst davon aus, dass alle Attribute die gleiche Menge an Speicherplatz belegen. Dadurch entspricht die Größe eines Tupels der Anzahl seiner, durch die zu untersuchende Attributkombination gegebenen, Attribute. Dies entspricht in Bezug auf Algorithmus 5 den Werten `bottom` bzw. `top`. Die Anzahl der Attribute bezeichnen wir im Folgenden mit m. Für die Duplikaterkennung ergibt sich für eine Relation mit n Tupeln eine Laufzeit von

$$O((m * n) * log_2(m * n)). \tag{4.5}$$

Sofern sich die Daten in der untersuchten Relation nicht ändern, bleibt die Anzahl der Tupel für jede Duplikaterkennung konstant, wodurch n aus der Kostenabschätzung entfernt werden kann. Die Kosten K für die Überprüfung einer einzelnen Attributkombination mit m Attributen beträgt demnach

$$K := m * log_2(m). \tag{4.6}$$

Sind Metadaten über die voraussichtliche Verteilung der Daten bekannt, lässt sich abschätzen, ob viele oder nur wenige Duplikate auftreten werden. In den beiden möglichen Extremfällen entartet die Laufzeit in Laufzeiten von $O(m*n)$ bis $O((m*n)^2)$. Für die gewichtete bidirektionale Breitensuche wurden dementsprechend folgende drei Wichtungsfunktionen umgesetzt:

1. Wichtungsfunktion für viele Duplikate: $K_{BETTER} := m$
2. Wichtungsfunktion für eine durchschnittliche Anzahl an Duplikaten: $K_{AVG} := m * log_2(m)$
3. Wichtungsfunktion für wenige Duplikate: $K_{WORSE} := m^2$

Um die Gesamtkosten K_{bottom} bzw. K_{top} für die nächsten Bottom-Up- und Top-Down-Phasen zu berechnen, müssen die Kosten der einzelnen Attributkombinationen pro Ebene aufsummiert werden. Für die vereinfachte Wichtungsfunktion entspricht dies dem Produkt aus den Kosten K_i der aktuellen Ebene i multipliziert mit der Anzahl der Attributmengen in `upperSet` bzw. `lowerSet`:

$$\begin{aligned} K_{bottom} &:= |lowerSet| * bottom * log_2(bottom) \\ K_{top} &:= |upperSet| * top * log_2(top) \end{aligned} \tag{4.7}$$

Sind die Kosten für K_{bottom} kleiner oder gleich K_{top}, so wird im nächsten Schleifendurchlauf das Bottom-Up-Verfahren angewendet, andernfalls das Top-Down-Verfahren. Die Bevorzugung des Bottom-Up-Verfahrens bei gleichen Kosten für beide Algorithmen basieren auf der Sicherstellung der Minimalität der Quasi-Identifikatoren.

[4] Da es sich bei den aufgerufenen Methoden $buildNewLowerSet$, $buildNewUpperSet$, $modifyUpperSet$ und $modifyLowerSet$ um einfache Funktionen zur Veränderung von Mengen handelt, werden diese nicht im Pseudocode dargestellt. Die komplette Implementierung dieser Methoden in der Programmiersprache Java befindet sich im Begleitmaterial.

[5] Im Begleitmaterial (siehe Anhang G) befinden sich alle protokollierten Abläufe für die gewichtete bidirektionale Breitensuche. In allen Testszenarien wurde dabei festgestellt, dass die Top-Down-Phase mehrfach übersprungen werden konnte bzw. nur eine geringe, meist einstellige, Zahl an Attributkombinationen zu überprüfen hatte.

Präzisere Wichtungsfunktion: Soll die Wichtungsfunktion präziser sein, so lässt sich der Aufwand abschätzen, indem für jede Attributkombination X die Summe s über die „Größen" der einzelnen Attribute A einer Attributkombination X gebildet wird. Unter „Größe" verstehen wir dabei den benötigten Speicherplatz in Bytes. Dieser richtet sich nach dem Datentyp, welcher beim Erzeugen der jeweiligen Relation festgelegt wurde.

Beispiel: Übersicht über den Speicherbedarf von Attributen in PostgreSQL 12

Die folgende Tabelle enthält eine Übersicht über verfügbare Datentypen (Auswahl) und deren Größe in PostgreSQL 12[a].

Typ	Beschreibung	Größe
smallint	Ganzzahlen mit kleinerem Wertebereich	2 Bytes
integer	Ganzzahlen	4 Bytes
bigint	Ganzzahlen mit größerem Wertebereich	8 Bytes
real	Gleitkommazahlen	4 Bytes
double precision	Gleitkommazahlen mit höherer Genauigkeit	8 Bytes
char	Einzelnes Zeichen	1 Byte
name	Interne Repräsentation der Objektidentität	64 Bytes
timestamp	Datums- und Zeitangabe	8 Bytes
date	Datumsangabe	4 Bytes
time	Tageszeit ohne Zeitzone	8 Bytes
time with time zone	Tageszeit mit Zeitzone	12 Bytes
interval	Zeitintervalle	16 Bytes
boolean	Wahrheitswert	1 Byte
point	Punkt	16 Bytes
line	Linie	32 Bytes
circle	Kreis	24 Bytes

[a]Die Übersicht wurde nach `https://www.postgresql.org/docs/12/datatype.html`, zuletzt aufgerufen am 05.01.2022, erstellt.

Die ermittelten Einzelgewichte der Attribute A werden anschließend zum Gesamtgewicht s eines Tupels bzw. dessen gewählter Attributkombination aufsummiert. Die Summe über alle Einzelsummen aus jeweils einer der beiden Mengen der bidirektionalen Breitensuche, $lowerSet$ und $upperSet$, ergibt anschließend das Gesamtgewicht für die nächste Bottom-Up- bzw. Top-Down-Phase:

$$\begin{aligned} K_{bottom} &:= \sum_{X \in lowerSet} (log_2(s) * s), \\ K_{top} &:= \sum_{X \in upperSet} (log_2(s) * s), \\ \text{mit } s &= \sum_{A \in X} size(A) \end{aligned} \tag{4.8}$$

Beispiel:

Die nachfolgende Tabelle zeigt die Größen der einzelnen Datentypen in der Relation `lineitem` aus der TPC-H-Datenbank. Die mit $\star$ gekennzeichneten Werte der größenvariablen Datentypen stehen für die maximale bzw. tatsächlich benötigte Anzahl an Bytes.

Attribut	Typ	Anzahl Bytes
l_orderkey	bigint	8
l_partkey	bigint	8
l_suppkey	bigint	8
l_linenumber	integer	4
l_quantity	numeric	10*
l_extendedprice	numeric	10*
l_discount	numeric	10*
l_tax	numeric	10*
l_returnflag	character(1)	1
l_linestatus	character(1)	1
l_shipdate	date	4
l_commitdate	date	4
l_receiptdate	date	4
l_shipinstruct	character(25)	25
l_shipmode	character(10)	10
l_comment	character varying(44)	44*
Summe		**161**

Betrachten wir die dreielementigen Attributkombinationen $\{l\_orderkey, l\_partkey, l\_suppkey\}$ und $\{l\_shipinstruct, l\_shipmode, l\_comment\}$, so erhalten wir Tupelgrößen von 24 bzw. 69 Bytes. Je nachdem, welche der Attributkombinationen durch die vorherigen Berechnungen aus dem Suchraum gestrichen wurden, entfallen unterschiedliche der Attributkombinationen im aktuellen Berechnungsschritt. Ist beispielsweise $\{l\_comment\}$ als Quasi-Identifikator bekannt, entfallen viele Attributkombinationen mit hohem Gewicht in der nächsten Top-Down-Phase, da dort das Attribut $l\_comment$ in vielen Kombinationen vertreten ist. In der Bottom-Up-Phase tritt dieses nur in wenigen Kombinationen auf, wodurch der Effekt geringer ist. Dadurch verlagert sich das Kostenverhältnis mehr zu Ungunsten des Bottom-Up-Verfahrens.

Die Entscheidung, welche der beiden Suchstrategien als nächstes ausgeführt wird, erfolgt analog zur einfachen Wichtungsfunktion. Diese Art der Wichtung eignet sich allerdings nur, wenn Zugang zu den Metadaten der betrachteten Datenbankrelation besteht.

Ein weiterer Faktor ist die Komplexität der Duplikattests für verschiedene Datentypen; diese ist unabhängig von der Größe des Dateityps. Dies betrifft insbesondere die Datentypen, welche eine unterschiedliche Darstellung des gleichen Wertes erlauben. Dazu gehören beispielsweise Gleitkommazahlen [DZC08] sowie Datums- und Zeitangaben.

Beispiel: Unterschiedliche Repräsentationen des gleichen Wertes

Die Zahl 42 lässt sich als Gleitkommazahl sowohl durch $42 * 10^0$ als auch $4.2 * 10^2$ bzw. $420 * 10^{-1}$ darstellen. Obwohl der eigentliche Wert unverändert bleibt, unterscheidet sich die Binärdarstellung aller drei Varianten.

Das gleiche Problem tritt auch bei der Darstellung von Zeitstempeln mit Angabe der Zeitzone auf. So entspricht der Zeitstempel $06/11/2019\ 03:42:08.15\ CET$ (nach europäischer Winterzeit) dem Zeitstempel $06/11/2019\ 23:42:08.15\ EDT$ (u. a. Teile von Brasilien).

Eine gesonderte Gewichtung dieser Dateitypen ist in der Implementierung des Verfahrens nicht vorgesehen, da die Verfahren zur Duplikaterkennung sehr vom verwendeten Datenbankmanagementsystem abhängen. Eine mögliche Umsetzung besteht in der individuellen Gewichtung der einzelnen Datentypen. Als weitere Verbesserung des Verfahrens wird im Folgenden ein iteratives Verfahren vorgestellt, welches die Änderungen der Daten in den Relationen überwacht.

Operation	Distinct Ratio		Separation Ratio	
	Fall ↓	Fall ↑	Fall ↓	Fall ↑
Einfügen	$\frac{X}{Y} \rightarrow \frac{X}{Y+1}$	$\frac{X}{Y} \rightarrow \frac{X+1}{Y+1}$	$\frac{X}{Y} \rightarrow \frac{X-1}{Y+1}$	$\frac{X}{Y} \rightarrow \frac{X+1}{Y+1}$
Löschen	$\frac{X}{Y} \rightarrow \frac{X-1}{Y-1}$	$\frac{X}{Y} \rightarrow \frac{X+1}{Y-1}$	$\frac{X}{Y} \rightarrow \frac{X-1}{Y-1}$	$\frac{X}{Y} \rightarrow \frac{X+1}{Y-1}$
Aktualisieren	$\frac{X}{Y} \rightarrow \frac{X-1}{Y}$	$\frac{X}{Y} \rightarrow \frac{X+1}{Y}$	$\frac{X}{Y} \rightarrow \frac{X-2}{Y}$	$\frac{X}{Y} \rightarrow \frac{X+2}{Y}$

Tabelle 4.1: Überblick über die Änderungen der Distinct und Separation Ratio bei der Anwendung von atomaren Einfüge-, Lösch- und Änderungsoperationen auf der bestehenden Relation. Der Fall ↓ gibt wieder, dass für eine bestehende Attributkombination die Werte sinken, wenn durch die Operation der Anteil der Äquivalenzklassen sinkt. Andersrum kennzeichnet der Fall ↑, dass die Werte steigen, wenn der Anteil der Äquivalenzklassen steigt.

4.3.3 Dynamische Berechnung der Distinct bzw. Separation Ratio

Auch wenn die gewichtete bidirektionale Breitensuche eine deutliche Reduzierung der Laufzeit erreicht, ist die Berechnung der Quasi-Identifikatoren in auf Stromdaten basierenden Informationssystemen unpraktikabel. Durch die ständigen Veränderungen im Datenbestand können sich die QIs relativ schnell ändern – bestehende QIs liegen nach einer Abfolge von Aktualisierungs-, Einfüge- oder Löschanweisungen unterhalb des zuvor festgelegten Grenzwertes, andere Attributkombinationen werden hingegen zu Quasi-Identifikatoren. Im Folgenden wird betrachtet, wie sich der Wert eines Quasi-Identifikators im ungünstigsten Fall für jede Art von Operation ändert, ohne Kenntnis über die vorliegende Verteilung der Daten zu besitzen.

Dafür bezeichnen wir mit X die Anzahl der eindeutigen (Separation Ratio) bzw. unterschiedlichen (Distinct Ratio) Tupel; Y steht für die Gesamtzahl aller Tupel ($|r(R)|$) in der jeweiligen Relation. Bei einer Einfügeoperation steigt die Anzahl aller Tupel um den Wert `1`; beim Löschen sinkt die Anzahl hingegen um den Wert `1`. Änderungsoperationen, die auch als eine Folge von einer Einfüge- und einer Löschoperation gesehen werden können, verändern die Anzahl der Tupel nicht.

Für die Werte von X betrachten wir diejenigen Attributkombinationen, die „nahe" am festgelegten Grenzwert t für die Quasi-Identifikatoren liegen. Für „nahe" legen wir zwei Parameter ε_1 und ε_2 fest, welche wahlweise den prozentualen Anteil oder die absolute Zahl von Tupel festlegen, die vom Grenzwert abweichen dürfen:

$$t - \varepsilon_1 \leq p \leq t + \varepsilon_2 \tag{4.9}$$

Beispiel: Intervall um den Grenzwert $t = 90\%$

Sollen beispielsweise die Attributkombinationen überwacht werden, welche jeweils 5% ober- und unterhalb des festgelegten Grenzwertes von $t = 90\%$ liegen, erhalten wir ein Intervall von

$$85\% \leq t \leq 95\% \tag{4.10}$$

Liegt der berechnete Wert p für eine Attributkombination A nach Distinct bzw. Separation Ratio innerhalb dieses Intervalls, so wird A mit überwacht.

Liegt eine, durch Algorithmus 5 berechnete, Attributkombination A innerhalb dieses Bereiches, so wird sie zur Laufzeit des Datenbanksystems überwacht. Für die negierten Quasi-Identifikatoren und die erkannten QIs innerhalb des Intervalls wird stets angenommen, dass durch eine Änderungsoperation auf dem Attribut $a \in A$ – einschließlich Löschen oder Einfügen eines gesamten Tupels – der berechnete Wert p sich hin zum Grenzwert t „bewegt", d. h. abnimmt (QIs) bzw. zunimmt (negierte QIs).

Die möglichen Änderungen am berechneten Wert p sind in Tabelle 4.1 zusammengefasst. Da vorausgesetzt ist, dass kein Wissen über die Verteilung der Daten vorliegt, wird davon ausgegangen, dass die Daten stets so abgeändert werden, dass sich dem Grenzwert t angenähert wird. Zur Erläuterung der Tabelle werden zwei Beispieländerungen angeführt.

Beispiel: Änderungsoperationen und deren Auswirkung auf die Distinct bzw. Separation Ratio

Betrachten wir zunächst folgende Ausgangsrelation mit dem möglichen Quasi-Identifikator $\{Vorname\}$, dem sensitiven Attribut $Titel$ und einem Grenzwert von $t = 70\%$:

Vorname	**Titel**
Holger	Good Morning Blues
Holger	Riverside
Hannes	Dancing Mad

Der berechnete Wert p für den Quasi-Identifikator liegt für die Distinct Ratio bei $66,\overline{6}\%$ und somit unter dem festgelegten Grenzwert. Wird ein neues Tupel $(Andreas, Am\ Fenster)$ in die Relation eingefügt, erhalten wir folgendes Ergebnis:

Vorname	**Titel**
Holger	Good Morning Blues
Holger	Riverside
Hannes	Dancing Mad
Andreas	Am Fenster

Der Wert p liegt nun bei 75% und somit über dem festgelegten Grenzwert. Dies entspricht der Abschätzung aus Tabelle 4.1, bei der die Anzahl aller Tupel und die Anzahl der unterschiedlichen Werte um je `1` gestiegen ist ($\frac{X}{Y} \rightarrow \frac{X+1}{Y+1}$). $\{Vorname\}$ wurde somit zum Quasi-Identifikator. Wird stattdessen die Separation Ratio verwendet, so liegt der Wert p jedoch erst bei 50%, da für das Attribut $Vorname$ nur zwei eindeutige Werte vorliegen. Erfolgt eine Änderung des ersten Tupels, indem `Holger` durch den Nutzer `Dennis` ausgetauscht wird, erhalten wir folgende Relation:

Vorname	**Titel**
Dennis	Good Morning Blues
Holger	Riverside
Hannes	Dancing Mad
Andreas	Am Fenster

Durch die Änderung nimmt das Attribut $Vorname$ vier verschiedene Werte an. Der Wert p steigt für die Separation Ratio somit auf 100% an ($\frac{X}{Y} \rightarrow \frac{X+2}{Y}$). Wird das Attribut $Titel$ geändert, erfolgt keine Neuberechnung der Distinct bzw. Separation Ratio, da der $Titel$ als sensitives Attribut nicht Teil des Quasi-Identifikators ist.

Sobald durch die neuen, approximierten Werte erkannt wird, dass eine Attributkombination den Grenzwert über- bzw. unterschreiten könnte, muss eine Neuberechnung der Quasi-Identifikatoren erfolgen. Die Überschreitung des Grenzwertes ist aus Datenschutzsicht erforderlich; eine Neuberechnung bei Unterschreitung ermöglicht ggf. einen geringeren Informationsverlust, da bei einer kleineren Anzahl an Quasi-Identifikatoren der notwendige Grad der Anonymisierung geringer ausfällt.

Nach Vorstellung des entwickelten Algorithmus zur effizienten Suche nach Quasi-Identifikatoren werden im folgenden Abschnitt die Implementierungen der wichtigsten Konzepte beschrieben. Deren Einbindung in den für PArADISE entwickelten Demonstrators wird in Anhang F näher vorgestellt.

4.4 Implementierung

In diesem Abschnitt wird die Implementierung der gewichteten bidirektionalen Breitensuche beschrieben. Ausgangspunkt ist, analog zur Beschreibung des Konzepts, die Berechnung der Distinct bzw. Separation Ratio. Anschließend werden die notwendigen Methoden zur Umsetzung der bidirektionalen Suche ausführlich erläutert. Abschließend wird die Implementierung der Wichtungsfunktion vorgestellt.

Distinct und Separation Ratio

Als Grundlage zur Bestimmung, ob eine Attributkombination ein Quasi-Identifikator ist, bedarf es der Berechnung der *Distinct Ratio* bzw. *Separation Ratio*. Die folgenden drei Quellcodeausschnitte zeigen, unabhängig von der konkreten Umsetzung in einem Datenbankmanagementsystem, standardkonforme SQL-Anweisungen, welche den gemeinsamen Nenner und die jeweiligen Zähler für beide Metriken berechnen. Dabei wird der Platzhalter `<relation>` für die angefragte Datenbankrelation verwendet; `<attributes>` für die aktuell zu überprüfende Attributkombination. Der Nenner lässt sich durch folgende SQL-Anweisung bestimmen:

```
1 SELECT COUNT(*)
2 FROM <relation>
```

Quellcodeausschnitt 4.1: SQL-Anweisung zur Bestimmung der Anzahl aller Tupel einer Relation

---

Für die *Distinct Ratio* werden zunächst die Duplikate eliminiert. Anschließend werden die verbleibenden Tupel gezählt:

```
1 SELECT COUNT(*)
2 FROM (
3   SELECT DISTINCT <attributes>
4   FROM <relation>
5 ) tmp
```

Quellcodeausschnitt 4.2: SQL-Anweisung zur Bestimmung der *Disinct Ratio*

Im Gegensatz zur *Distinct Ratio* entfernt die Separation Ratio alle Tupel, die mehrfach in der Relation vorkommen. Dies lässt sich durch eine Gruppierung der Daten nach den Attributen der zu überprüfenden Attributkombination realisieren. Dabei erfolgt nach der Gruppierung eine Selektion auf den entstandenen Gruppen, welche jeweils nur ein Tupel enthalten dürfen:

```
1 SELECT COUNT(*)
2 FROM (
3   SELECT <attributes>
4   FROM <relation>
5   GROUP BY <attributes>
6   HAVING COUNT(*) = 1
7 ) tmp
```

Quellcodeausschnitt 4.3: SQL-Anweisung zur Bestimmung der *Separation Ratio*

Das folgende Beispiel zeigt die Zusammenführung der Berechnungen von Zähler und Nenner für die *Distinct Ratio*. Als konkretes Datenbankmanagementsystem wird PostgreSQL verwendet.

Beispiel: Zusammengefasste SQL-Anweisung für die Berechnung der Distinct Ratio

Die folgende Beispielanfrage zeigt die Berechnung der Distinct Ratio für die Attributkombination `{title, artid}` anhand der PostgreSQL-Syntax. In den Zeilen 1 bis 7 erfolgt die Bestimmung des Zählers (Anzahl der durch die Attributkombination bestimmten verschiedenen Tupel). PostgreSQL hält sich syntaktisch an den SQL-Standard [Mel16], wodurch die Unteranfrage in den Zeilen 4 und 5 benötigt wird. Kommt MySQL bzw. MariaDB zum Einsatz, kann die Anfrage auch mit `COUNT(DISTINCT <attributes>) FROM <relation>` abgekürzt werden[a].

In den Zeilen 8 bis 11 wird der Nenner, d. h. die Anzahl aller Tupel in der Datenbank, bestimmt. Abschließend erfolgt in den Zeilen 12 und 13 die Division beider Werte zur Bestimmung der Distinct Ratio.

```
WITH xRel AS (
  SELECT CAST(COUNT(*) AS DOUBLE PRECISION) AS x
  FROM (
    SELECT DISTINCT title, artid
    FROM tracks
  ) tmp
),
yRel AS (
  SELECT COUNT(*) AS y
  FROM tracks
)
SELECT x/y AS p
FROM xRel CROSS JOIN yRel
```

[a]siehe https://dev.mysql.com/doc/refman/8.0/en/aggregate-functions.html#function_count-distinct, zuletzt aufgerufen am 05.01.2022.

Bidirektionale Breitensuche

Zur Erhöhung der Übersichtlichkeit wurde der hier abgebildete Quellcode an einigen Stellen gekürzt. Dies betrifft beispielsweise die Deklaration einiger Variablen und Anweisungen zum vorzeitigen Verlassen von Schleifen. Der vollständige Quellcode befindet sich im Begleitmaterial zu dieser Arbeit. Die Bezeichnung der Variablen und Methoden im Quellcode orientiert sich an den Pseudocodes aus dem vorherigen Abschnitt. Sofern nicht anders angegeben, sind die nicht-deklarierten Variablen als globale Variablen, auf welche alle Methoden zugreifen können, anzusehen.

Bottom-Up: Der folgende Quellcodeausschnitt 4.4 zeigt die Anweisungen für eine einzelne Bottom-Up-Phase. Für jede Attributkombination `set` aus der aktuellen unteren Testmenge `lowerSet` wird ein vollständiger Scan (Methode `getPrecentage`) auf der Tabelle `table`, welche auf die Attribute in `set` projiziert wird, ausgeführt. Wird der festgelegte Grenzwert `threshold` überschritten, so wird die Attributkombination `sets` in die Menge der minimalen Quasi-Identifikatoren `qiLowerSet` übernommen.

```
public void bottomUpTest() {
  for(Set<String> set: lowerSet)
    if(getPrecentage(set, table) >= threshold)
      qiLowerSet.add(set);
}
```

Quellcodeausschnitt 4.4: Java-Code für die Bottom-Up-Phase

Nach der Bottom-Up-Phase werden die neu berechneten Quasi-Identifikatoren zur Reduzierung des Suchraumes der nächsten Top-Down-Phase verwendet. Die Reduktion des Suchraumes übt direkten Einfluss auf das Ergebnis der Wichtungsfunktion aus, die zu Beginn des nächsten Durchlaufes aufgerufen wird. Für jede Attributkombination `set` aus den zuletzt bestimmten, aber ggf. noch nicht minimalen Quasi-Identifikatoren in `possibleQiUpperSet` wird überprüft, ob diese eine Obermenge der ermittelten minimalen Quasi-Identifikatoren aus der Menge `qiLowerSet` oder eine Teilmenge der bekannten negierten QIs aus `optOutUpperSet` sind. Tritt einer der beiden Fälle ein, wird die Attributkombination `set` aus dem Suchraum entfernt. Der folgende Quellcodeausschnitt 4.5 zeigt den dazugehörigen Java-Code:

```
public void modifiyUpperSet() {
  Set<Set<String>> sets = new HashSet<Set<String>>();
  for(Set<String> set: possibleQiUpperSet){
    boolean out = false;
    for(Set<String> pos: qiLowerSet)
      if(set.containsAll(pos))
        out = true;
    for(Set<String> pos: optOutUpperSet)
      if(pos.containsAll(set))
        out = true;
    if(!out)
      sets.add(set);
  }
  possibleQiUpperSet = sets;
}
```

Quellcodeausschnitt 4.5: Java-Code zur Reduzierung des Suchraumes der nächsten Top-Down-Phase

Top-Down: Der Top-Down-Algorithmus ist analog zum Bottom-Up-Verfahren in Quellcodeausschnitt 4.6 implementiert worden, allerdings wird im ersten Schritt zunächst eine Kopie (`possibleQICopy`) der bekannten Quasi-Identifikatoren `possibleQiUpperSet` aus der letzten Top-Down-Phase erzeugt. Bei erfolgreicher Erkennung eines QIs (`set`) durch die Methode `getPrecentage` wird überprüft, ob einige der bekannten Quasi-Identifikatoren (`pos`) durch `set` ersetzt werden können, da $set \subset pos$ gilt und durch das Verfahren nur die minimalen Quasi-Identifikatoren ermittelt werden sollen. Wird `set` hingegen als negierter QI erkannt, so wird diese Attributkombination zur Menge der negierten Quasi-Identifikatoren `optOutUpperSet` hinzugefügt. Abschließend wird `possibleQiUpperSet` auf dessen Arbeitskopie `possibleQICopy` gesetzt.

```
public void topDownTest() {
  Set<Set<String>> possibleQICopy = new HashSet<Set<String>>();
  possibleQICopy.addAll(possibleQiUpperSet);
  for(Set<String> set: upperSet){
    if(getPrecentage(set, table) >= threshold)
      for(Set<String> pos: possibleQiUpperSet)
        if(pos.containsAll(set))
          possibleQICopy.remove(pos);
    else
      optOutUpperSet.add(set);
  }
  possibleQiUpperSet = possibleQICopy;
}
```

Quellcodeausschnitt 4.6: Java-Code für die Top-Down-Phase

Analog zur Einschränkung des oberen Suchraumes erfolgt in Quellcodeausschnitt 4.7 die Reduzierung des unteren Suchraumes. Die Variable `lowerSetCopy` bezeichnet hier die Arbeitskopie von der aktuellen unteren Testmenge `lowerSet`. Eine Attributmenge `set` wird aus der aktuellen unteren Suchraummenge `lowerSet` entfernt, wenn mindestens ein QI (`optOut`) aus der Menge der bekannten minimalen Quasi-Identifikatoren existiert, der eine Teilmenge von `set` ist. Als weiteres, hinreichendes Ausschlusskriterium gilt, dass wenn `set` in keiner der Attributmengen aus der aktuellen oberen Testmenge `upperSet` mehr vorkommt, die Attributmenge `set` auch kein Quasi-Identifikator sein kann. Existiert hingegen mindestens eine solche Attributmenge (gekennzeichnet durch `outEverywhere = false`), so wird `lowerSetCopy` um die Attributmenge `sets` erweitert.

```
public void modifyLowerSet() {
  Set<Set<String>> lowerSetCopy = new HashSet<Set<String>>();
  for(Set<String> set: lowerSet){
    boolean out = false;
    for(Set<String> optOut: qiLowerSet)
      if(set.containsAll(optOut))
        out = true;
    boolean outEverywhere = true;
    for(Set<String> ooU: upperSet)
      if(ooU.containsAll(set))
        outEverywhere = false;
    if(!out&&!outEverywhere)
      lowerSetCopy.add(set);
  }
  lowerSet = lowerSetCopy;
}
```

Quellcodeausschnitt 4.7: Java-Code zur Beschneidung des Suchraumes der nächsten Bottom-Up-Phase

Wichtungsfunktion: Durch die Wichtungsfunktion in Quellcodeausschnitt 4.8 werden die Größen der zu überprüfenden Attributmengen für die nächste Bottom-Up- und Top-Down-Phase bestimmt. Mittels der Mengenfunktion `size()` wird die Anzahl der verbleibenden Attributkombinationen in den Testmengen `lowerSet` bzw. `upperSet` bestimmt und mit den aktuellen Werten für die Anzahl der Elemente in den Attributkombinationen für die Bottom-Up- (`bottom`) und die Top-Down-Phase (`top`) sowie deren Logarithmus[6] multipliziert. Liegen die ermittelten Kosten `weightLow` für die nächste Bottom-Up-Phase unterhalb der Kosten `weightTop` für die nächste Top-Down-Phase, wird durch den Wert `true` der Bottom-Up-Algorithmus ausgewählt; andernfalls wird `false` für den Top-Down-Algorithmus zurückgegeben.

```
public boolean isLowerSetNext(){
  weightLow = Math.log(bottom)*bottom*lowerSet.size();
  weightTop = Math.log(top)*top*upperSet.size();
  if(weightLow <= weightTop)
    return true;
  return false;
}
```

Quellcodeausschnitt 4.8: Java-Code zur Bestimmung der nächsten Suchstrategie

Der abschließende Quellcodeausschnitt 4.9 zeigt die Implementierung der bidirektionalen Breitensuche in stark gekürzter Form. Als globale Variablen liegen vor Aufruf der Funktion `topDownBottomUpDetection()` die Datenbankverbindung einschließlich der zu untersuchenden Datenbankrelation sowie der Grenzwert für die Quasi-Identifikatoren vor. In den Zeilen 2 bis 6 erfolgt die Initialisierung der notwendigen lokalen und globalen Variablen. Die darauffolgenden Zeilen 8 bis 13 bestimmen die grundlegenden Parameter für den Algorithmus. Dazu gehören die Ermittlung der Attributnamen aus der (extern festgelegten) Datenbankrelation `table` und der darin enthaltenen Anzahl an Tupeln. Aus diesen beiden Werten werden die Werte für die Variablen `top` und `bottom` abgeleitet, welche die Zahl der aktuellen Attributebene für den Bottom-Up- und Top-Down-Algorithmus widerspiegeln. Diese entsprechen bei der Initialisierung den Werten `1` für alle einelementigen Attributmengen bzw. der Anzahl der Attribute in `table` für die eine top-elementige Attributmenge.

[6] Im Quellcodeausschnitt wird von einer Normalverteilung der Werte – mit weder zu wenig noch zu vielen Duplikaten – ausgegangen. In diesen Fällen liegt der Faktor bei $bottom$ bzw. top für den günstigen Fall mit wenigen Duplikaten bis hin zu $bottom * bottom$ bzw. $top * top$ für viele Duplikate.

In den Zeilen 15 bis 32 wird der eigentliche Algorithmus beschrieben. Solange nicht alle Ebenen überprüft wurden (Zeile 15) bzw. der Bottom-Up-Algorithmus Attributkombinationen zu überprüfen hat (Zeile 17 und 18), wird zunächst die Wichtungsfunktion (Zeile 16) aufgerufen. Je nach dessen Ergebnis wird entweder die nächste Bottom-Up-Phase (Zeilen 20 bis 24) bzw. die nächste Top-Down-Phase (Zeilen 26 bis 30) ausgeführt.

Beide Phasen verlaufen ähnlich: Die Tests werden ausgeführt (Zeile 20 bzw. 26), anschließend wird die neue untere (Zeile 22) bzw. obere (Zeile 28) Testmenge gebildet. Parallel dazu werden die oberen (Zeile 23) bzw. unteren (Zeile 29) Testmengen auf Basis der neu ermittelten minimalen bzw. negierten Quasi-Identifikatoren erstellt. Abschließend erfolgt das Inkrementieren (Zeile 24) bzw. Dekrementieren (Zeile 30) der beiden Zählvariablen.

```
1  public void topDownBottomUpDetection() throws Exception{
2    possibleQiLowerSet = new HashSet<Set<String>>();
3    possibleQiUpperSet = new HashSet<Set<String>>();
4    optOutUpperSet = new HashSet<Set<String>>();
5    attributes = new HashSet<String>();
6    Set<Set<String>> currentSet = new HashSet<Set<String>>();
7
8    attributes.addAll(getColumnNames(table));
9    numTuple = getNumberOfTuple(table);
10   top = attributes.size();
11   bottom = 1;
12   upperSet = buildCompleteSet();
13   lowerSet = addElementsToSet(currentSet, attributes);
14
15   while(bottom<=top){
16     boolean lowNext = isLowerSetNext();
17     if(weightLowEqualsZero)
18       break;
19     if(lowNext){
20       bottomUpTest();
21       if(!(bottom==top))
22         lowerSet = addElementsToSet(lowerSet, attributes);
23       modifyUpperSet();
24       bottom++;
25     }else{
26       topDownTest();
27       if(!(bottom==top))
28         upperSet = removeElementsFromSet(upperSet);
29       modifyLowerSet();
30       top--;
31     }
32   }
33 }
```

Quellcodeausschnitt 4.9: Java-Code zur Implementierung der bidirektionalen Breitensuche

---

Nach der Implementierung des vorgestellten Verfahrens wurde dessen Leistungsfähigkeit anhand verschiedener Testszenarien untersucht. Die dabei erzielten Ergebnisse sind im nachfolgenden Abschnitt 4.5 festgehalten.

4.5 Testergebnisse

In diesem Abschnitt wird das entwickelte Verfahren zur bidirektionalen Breitensuche nach Quasi-Identifikatoren gegenüber den Bottom-Up- und Top-Down-Ansatz evaluiert. Im Unterabschnitt 4.5.1 werden zunächst die verwendeten Datensätze und Datenbankumgebungen vorgestellt. Anschließend erfolgt in Unterabschnitt 4.5.2 die detail-

lierte Evaluation anhand des Zensus-Datensatzes. Abschließend werden die Ergebnisse der QI-Erkennung für die weiteren Anwendungsfälle in Unterabschnitt 4.5.3 besprochen.

4.5.1 Verwendete Datensätze und Datenbanken

Die Evaluation des Verfahrens erfolgte anhand von drei unterschiedlichen Datensätzen. Zudem wurden unterschiedliche Datenbanksysteme zur Umsetzung des Verfahrens untersucht.

Datensätze

Als Standarddatensatz für das Testen von Anonymisierungsverfahren gilt der sogenannte „Adult"-Datensatz aus dem UCI Machine Learning Repository [KB96], welcher (teil-)anonymisierte Steuerdaten von US-Bürgern enthält. Der Datensatz besteht aus lediglich einer Relation mit insgesamt 32561 Tupeln[7]. In der Relation wurden in 15 Attributen personenbezogene Daten wie Alter, Geschlecht und Staatsangehörigkeit gespeichert. Die Daten lagen zunächst im CSV-Format vor und wurden zum Testen in eine MariaDB- und eine PostgreSQL-Datenbank importiert.

Als zweiter Testdatensatz wurde der im Datenbankbereich etablierte TPC-H-Benchmark[8] mit einem Skalierungsfaktor von `1` gewählt. Die relationale Struktur des Benchmarks ähnelt einem Snowflake-Schema, in dessen Kern die Faktentabelle `lineitem` steht. Durch den gewählten Skalierungsfaktor enthält `lineitem` insgesamt 6 Millionen Tupel. In der verwendeten PostgreSQL-Datenbank entspricht dies ca. 1 GB an Rohdaten verteilt auf 16 Attribute. Über Fremdschlüsselbedingungen wird `lineitem` mit den angrenzenden Dimensionstabellen verknüpft. Diese enthalten Daten zu Kunden, Herstellern und Produkten.

Der dritte Datensatz beruht auf aufgezeichneten Sensordaten, welche im Rahmen des Graduiertenkollegs MuSAMA an der Universität Rostock (siehe Abschnitt 2.1.2) entstanden sind. Zu den verwendeten Sensoren zählen u. a. UbiSense-Tags, smarte Steckdosen und Temperatursensoren. Die Aufzeichnungen der UbiSense-Tags stellen mit insgesamt 1,7 Millionen Tupeln und 22 Attributen den umfangreichsten Testdatenbestand dar. Der Datensatz wurde auf einer MariaDB-Datenbank getestet.

Datenbanksysteme

Als Testumgebung wurde eine Client-Server-Architektur auf Basis von JDBC und Java SE 8 aufgesetzt. Das Client-Notebook wurde mit einer i7-3630QM-CPU mit vier Kernen (je 2,3 GHz und 6 MB Cache) betrieben. Der Arbeitsspeicher in Höhe von 8 GB war für die Tests mehr als ausreichend, da clientseitig keine komplexeren bzw. datenlastigen Berechnungen ausgeführt wurden.

Als primär verwendetes Datenbanksystem wurde MariaDB mit InnoDB als Speichersystem verwendet. MariaDB wurde auf einer virtuellen Maschine aufgesetzt, welche mit einer 64-Bit-CPU (vier Kerne @ 2 GHz und jeweils 4 MB Cache) und 4 GB Arbeitsspeicher ausgestattet ist.

PostgreSQL wurde in Version 11.3 für weitere Tests genutzt. Das Datenbanksystem wurde auf einer virtuellen Maschine mit der gleichen Konfiguration wie der MariaDB-Server aufgesetzt.

4.5.2 Laufzeitvergleiche am Beispiel des Zensus-Datensatzes

Tabelle 4.2 und Abbildung 4.5 zeigen die Messwerte für die Laufzeiten der drei Verfahren Bottom-Up, Top-Down und die gewichtete, bidirektionale Breitensuche mit der mittleren Wichtungsstrategie `AVG` bei unterschiedlichen Grenzwerten. Die Messwerte beziehen sich auf den Zensus-Datensatz auf der MariaDB-Datenbank. Für jede Kombination aus verwendetem Verfahren und Grenzwert wurde eine Messung vorgenommen. Durch Tests bei der Entwicklung der Verfahren wurde im Vorfeld festgestellt, dass die möglichen Abweichungen in der Laufzeit zu gering sind, um signifikant zu sein, wodurch das mehrfache Ausführen der Tests an dieser Stelle vernachlässigt wurde.

[7] In der Beschreibung des Datensatzes werden 48842 Tupel bzw. „Instanzen" erwähnt; der verfügbare Datenbestand enthält jedoch eine geringere Anzahl an Tupeln.

[8] `http://www.tpc.org/tpch/`, zuletzt aufgerufen am 05.01.2022

Grenzwert in %	Laufzeit in Sekunden		
	Bottom-Up	*Top-Down*	*Bidirektional*
50	1143	8482	478
60	1306	8357	451
70	1487	8062	361
80	1703	7742	259
90	2133	7364	53
95	2299	7269	234
96	2337	7120	347
97	2473	6931	783
98	3127	6193	774
99	4126	5397	985

Tabelle 4.2: Vergleich der Laufzeiten für die Algorithmen Bottom-Up, Top-Down und die gewichtete bidirektionale Breitensuche für verschiedene Grenzwerte t zwischen 50% und 99%. Aufgrund von Laufzeitüberschreitungen (> 12 h) liegen keine Messwerte für das Brute-Force-Verfahren (Test aller Attributkombinationen) vor.

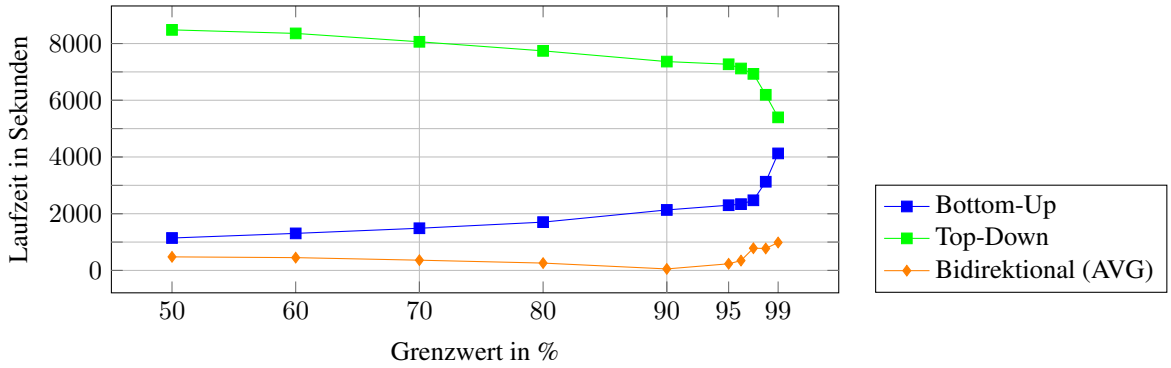


Abbildung 4.5: Laufzeit für die Verfahren Bottom-Up (blau), Top-Down (grün) und die gewichtete bidirektionale Breitensuche (orange) für verschiedene Grenzwerte t (nach Distinct Ratio).

Die Tabelle 4.3 und Abbildung 4.6 zeigen die Anzahl der benötigten Tabellenscans für jedes der drei Verfahren bei einem Grenzwert von $t = 90\%$ nach der Distinct Ratio. Insbesondere Abbildung 4.6 verdeutlicht die Wirkungsweise der bidirektionalen Breitensuche: Stellen wir uns die Zahlen der notwendigen Tabellenscans pro Ebene als gegenläufige „Wellen" für das Bottom-Up- (blaue Linie) bzw. Top-Down-Verfahren (grüne Linie) vor, die bei mittlerer Attributanzahl (7 bzw. 8 Attribute) ihren Höhepunkt erreichen. Treffen in der Natur gegenläufige Wellen aufeinander, so löschen sie sich im günstigsten Fall gegenseitig aus. Gleiches geschieht bei der bidirektionalen Breitensuche: Durch die Reduktion des Suchraumes in beide Richtungen flachen die „Wellen", also die Zahl der notwendigen Tabellenscans stark ab; es verbleibt die in Abbildung 4.6 dargestellte flache, orange Linie. Im Folgenden werden die Messwerte für die eingesetzten Verfahren näher betrachtet.

Top-Down

Das Top-Down-Verfahren zeigt eine Tendenz zu einer verbesserten Laufzeit bei steigendem Grenzwert und nähert sich der Performance des Bottom-Up-Verfahrens an. Die Ursache für dieses Verhalten liegt darin, dass bei hohen Grenzwerten viele Attributkombinationen frühzeitig ausgeschlossen werden können. Da ungeachtet dessen

Anzahl Attribute	Verglichene Attributkombinationen pro Verfahren			
	Brute-Force	*Bottom-Up*	*Top-Down*	*Bidirektional*
1	15	15	0	15
2	105	105	1	105
3	455	442	41	78
4	1365	1108	273	107
5	3003	2120	896	118
6	5005	3084	1929	81
7	6435	3464	2979	32
8	6435	3007	3432	4
9	5005	2002	3003	0
10	3003	1001	2002	0
11	1365	364	1001	0
12	455	91	364	0
13	105	14	91	0
14	15	1	15	2
15	1	0	1	1
Gesamt	**32767**	**16818**	**19007**	**543**

Tabelle 4.3: Anzahl der untersuchten Attributkombinationen für die verschiedenen Suchstrategien im direkten Vergleich. Als Testrelation wurde der bekannte Zensus-Datensatz mit 15 Attributen bei einem Grenzwert von $t = 90\%$ (Distinct Ratio) genutzt. Es wird pro Anzahl an Attributen in den Kombinationen die Zahl der notwendigen Tabellenscans aufgeführt.

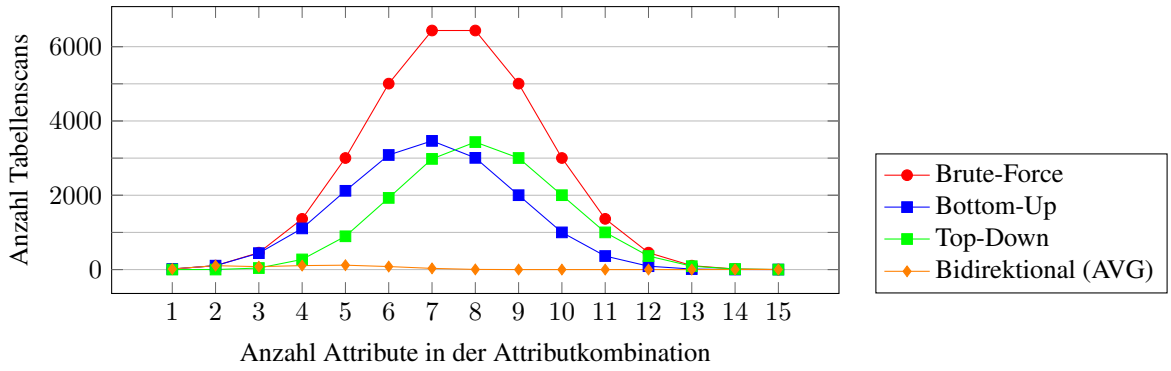


Abbildung 4.6: Anzahl der benötigten Tabellenscans für alle Verfahren pro Größe n der zu testenden Attributkombinationen bei einem Grenzwert von $p = 90\%$ für die Distinct Ratio. Für das Brute-Force-Verfahren werden dabei unabhängig von der Verteilung der Daten stets $\binom{n}{15}$-1 Tablescans für Attributkombinationen der Größe n benötigt.

zunächst die „teuren" Attributkombinationen mit vielen Attributen in den Hauptspeicher geladen werden müssen, liegt die Laufzeit jedoch deutlich über dem Bottom-Up-Verfahren.

Sonderfall $t = 100\%$: Bei einem (hier nicht betrachtetem) Grenzwert von $t = 100\%$ ist das Top-Down-Verfahren am schnellsten, da der Datensatz so strukturiert ist, dass selbst bei Hinzunahme aller Attribute nur ein vollständiger Scan notwendig ist. Das Bottom-Up-Verfahren entspricht in diesem Fall dem Brute-Force-Vorgehen, da alle Attributkombinationen überprüft werden müssen. Die bidirektionale Breitensuche erkennt im dritten Durch-

Grenzwert	**50%**		**90%**		**99%**	
Algorithmus	*Bottom-Up*	*Bidirektional*	*Bottom-Up*	*Bidirektional*	*Bottom-Up*	*Bidirektional*
Anzahl Vergleiche	12501	4076	16818	543	23444	3365
Anteil	32,61%		3,22%		14,35%	
Laufzeit in s	1143	478	2133	53	4126	985
Faktor	$\approx$ 2,5x		$\approx$ 40x		$\approx$ 4x	

Tabelle 4.4: Vergleich der Laufzeit verschiedener Suchstrategien und der benötigten Tabellenscans für ausgewählte Grenzwerte.

lauf – nach Prüfung aller ein- und zwei-elementigen Attributkombinationen – dass die gesamte Relation unter Einbeziehung aller Attributen mindestens ein Duplikat enthält.

Bottom-Up

Die Laufzeiten des Bottom-Up-Verfahrens verhalten sich gegenläufig zum Top-Down-Verfahren: Mit steigendem Grenzwert steigen auch die Laufzeiten des Verfahrens. Ähnlich zum Top-Down-Ansatz schlagen auch hier die Zeiten ab 95% schwunghaft um. Dies ist auf die Verteilung der Daten zurückzuführen. Durch die höheren Grenzwerte wird der Suchraum später eingeschränkt, da weniger Attributkombinationen als Quasi-Identifikatoren erkannt werden. Da zu Beginn des Verfahrens die „kleineren" Attributkombinationen mit geringerem Aufwand geladen werden, beträgt die Laufzeit für die meisten betrachteten Grenzwerte nur einen Bruchteil vom Top-Down-Verfahren.

Bidirektionale Breitensuche

Wie der Tabelle 4.3 bzw. der Abbildung 4.6 zu entnehmen, ist die Laufzeit für die gewichtete, bidirektionale Breitensuche am Beispiel des Zensus-Datensatzes deutlich geringer als für die anderen beiden Verfahren. Interessant zu beobachten ist die Entwicklung der Laufzeit bei steigendem Grenzwert. Während bei $t = 50\%$ die Laufzeit bei 478 Sekunden lag, sank diese schrittweise auf 53 Sekunden bei $t = 90\%$ ab. Anschließend stieg die Laufzeit wieder, bis sie bei $t = 99\%$ bei 985 Sekunden lag. Primärer Faktor für diese Entwicklung ist wiederum die Verteilung der Daten. Je nach gewähltem Grenzwert werden bestimmte minimale bzw. negierte Quasi-Identifikatoren früher oder später erkannt, wodurch der Suchraum mehr oder weniger stark eingeschränkt wird.

Tabelle 4.4 zeigt die Anzahl der benötigten Vergleiche und die Laufzeiten des Bottom-Up-Verfahrens und der bidirektionalen, gewichteten Breitensuche für die Grenzwerte 50%, 90% und 99%. Am stärksten sind die Unterschiede bei $t = 90\%$ ausgeprägt. Bei diesem Grenzwert müssen bei der bidirektionalen Breitensuche nur 3,22% der Tabellenscans ausgeführt werden und die Laufzeit beträgt nur rund $\frac{1}{40}$ des Bottom-Up-Verfahrens.

Als weiterer, möglicher Einflussfaktor auf die Laufzeit kann die Anzahl der vorhandenen minimalen Quasi-Identifikatoren angesehen werden. Während bei $t = 50\%$ 426 minimale QIs erkannt wurden, sank deren Anzahl bis $t = 90\%$ auf 56. Anschließend stieg deren Zahl langsam wieder auf 117 bei $t = 99\%$. Hintergrund hierfür ist allerdings wiederum die Verteilung der Daten. Wird bei einem Grenzwert t_1 ein minimaler Quasi-Identifikator $\{A_1, A_2, \dots A_n\}$ mit n Attributen erkannt, so kann dieser bei einem Grenzwert t_2 mit $t_1 < t_2$ nicht mehr gültig sein. Stattdessen können aus dem bestehenden QI zwei oder mehr Quasi-Identifikatoren der Form $\{A_1, A_2, \dots A_n, A_{n+1}\}$ entstehen. Gleiches gilt für sinkende Grenzwerte, bei denen aus einem QI der Form $\{A_1, A_2, \dots A_n, A_{n+1}\}$ mehrere Quasi-Identifikatoren der Form $\{A_1, A_2, \dots A_n\}$ entstehen.

Zusammenfassung: Die gewichtete bidirektionale Breitensuche ist nach den ermittelten Messwerten für den Zensus-Datensatz sehr gut geeignet, um die notwendige Rechenzeit zur Bestimmung aller minimalen Quasi-Identifikatoren effektiv zu reduzieren. Im folgenden Abschnitt wird kurz drauf eingegangen, welche Auswirkungen der neu entwickelte Algorithmus auf die beiden anderen Anwendungsszenarien hat.

Grenzwert in %	Laufzeit in Sekunden	Anzahl QIs
50	22178	170
60	25576	181
70	30223	194
80	35682	218
90	43240	239
95	55277	261
96	60149	250
97	64711	249
98	70029	271
99	75689	286

Tabelle 4.5: Laufzeit und Anzahl Quasi-Identifikatoren für den TPC-H-Datensatz, Relation $lineitem$.

4.5.3 Weitere Anwendungsszenarien

Als weitere Anwendungsszenarien bzw. Benchmarks wurden der bekannte TPC-H-Benchmark und eine Serie von aufgezeichneten Sensordaten betrachtet. Für den TPC-H-Benchmark liegt der Fokus der Untersuchung auf der Korrelation zwischen Laufzeit und der Anzahl der gefundenen, minimalen Quasi-Identifikatoren. Für die im Rahmen von MuSAMA erzeugten Sensordaten werden die gefundenen QIs und deren Auswirkung auf die Anonymisierung der Daten untersucht. Für beide Datensätze wurde ausschließlich Algorithmus 5 (Gewichtete, bidirektionale Breitensuche) verwendet. Die Gründe hierfür liegen in der Verteilung und der Menge an Daten. Bei ersten Tests der Bottom-Up- und Top-Down-Verfahren zeigte sich, dass die Quasi-Identifikatoren derart verteilt sind, dass die Laufzeiten wie bereits beim Zensus-Datensatz um mehrere Größenordnungen auseinanderliegen.

TPC-H-Benchmark

Auf dem TPC-H-Benchmark wurden mittels der gewichteten bidirektionalen Breitensuche die minimalen Quasi-Identifikatoren für alle acht Relationen bestimmt. Im Folgenden wird lediglich die Relation `lineitem` betrachtet, da diese hinsichtlich der vorhandenen Tupel und Attribute die umfangreichste Relation darstellt.

In Tabelle 4.5 und Abbildung 4.7 werden die Messwerte für die Laufzeit und die Anzahl der ermittelten Quasi-Identifikatoren aufgeführt. Im Gegensatz zum Zensus-Datensatz steigt die Laufzeit stetig an. Während bei einem Grenzwert von $t = 90\%$ die Laufzeit bei 22178 Sekunden ($\approx$ 6,16 Stunden) liegt, beträgt die Laufzeit bei einem Grenzwert von 90% bereits 43240 Sekunden ($\approx$ 12,01 Stunden) und bei 99% schließlich 75689 Sekunden ($\approx$ 21,02 Stunden). Aufgrund der hohen Laufzeiten wurde die Messung nur einmal durchgeführt.

Die Ursache für die erhöhte Laufzeit lässt sich auf drei Faktoren zurückführen: Zum einen ist die Tupelanzahl in der Relation `lineitem` mit 6 Millionen Tupeln rund 184 Mal so hoch wie im Zensus-Datensatz (32561 Tupel), wodurch bereits ein einzelner Tabellenscan inkl. Duplikaterkennung deutlich länger dauert. Weiterhin besteht die `lineitem`-Relation aus 16 Attributen und somit aus einem Attribut mehr als der Zensus-Datensatz. Dies führt zu einer Verdopplung des potentiellen Suchraums und somit der zu erwartenden Berechnungszeit.

Als dritter Faktor nimmt wieder die im Vorfeld unbekannte Verteilung der Daten eine wichtige Rolle ein. Wie in Abbildung 4.7 gut zu erkennen ist, steigt die Zahl der minimalen Quasi-Identifikatoren parallel zur benötigten Berechnungszeit[9]. Diese Korrelation lässt sich auf die Verteilung der Daten zurückführen. In Tabelle 4.6 wird näher auf die Auswirkungen der Datenverteilung auf die Einschränkung des Suchraumes durch die Quasi-Identifikatoren eingegangen. Wie der Gesamtzahl von notwendigen Tabellenscans zu erkennen ist, wurde auf Grund der Datenverteilung bei einem gleichen Grenzwert von $t = 90\%$ (Distinct Ratio) der Suchraum weniger stark eingeschränkt als im Zensus-Szenario. Dies trägt zu einer deutlich erhöhten Laufzeit bei.

[9]Zwei Ausnahmen bilden die Werte für einen Grenzwert von 96% bzw. 97%, bei denen die Zahl der minimalen Quasi-Identifikatoren leicht gesunken ist.

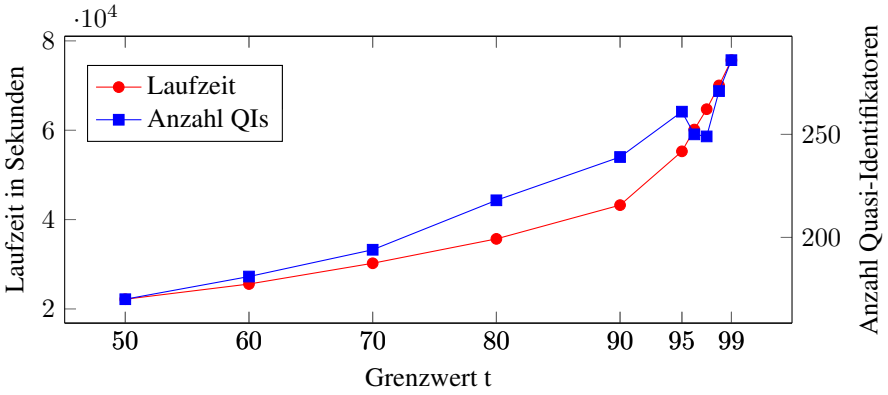


Abbildung 4.7: Laufzeit und Anzahl Quasi-Identifikatoren für den TPC-H-Datensatz, Relation $lineitem$.

Anzahl Attribute	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	**Gesamt**
Relation **lineitem**	16	120	337	525	361	77	65	75	57	11	13	1	0	0	0	1	**1659**
Relation **Zensus**	15	105	78	107	118	81	32	4	0	0	0	0	0	2	1	—	**543**

Tabelle 4.6: Anzahl der notwendigen Tabellenscans für je eine Relation aus dem TPC-H-Benchmark und dem Zensus-Datensatz. Insbesondere bei den mittelgroßen Attributkombinationen mit sieben bis neun Attributen werden auf der Relation $lineitem$ deutlich mehr Scans benötigt. Die Relation liegt bei der Gesamtzahl an Tabellenscans deutlich vor dem Zensus-Datensatz, selbst wenn der Faktor 2 für den erweiterten Suchraum abgezogen wird.

Sensordaten in MuSAMA

Die bidirektionale Breitensuche wurde auf verschiedene Sensordaten, die im Rahmen des Graduiertenkollegs MuSAMA entstanden, angewendet. Dabei stellte sich heraus, dass in den betrachteten Sensordaten die Daten derart verteilt sind, dass die Quasi-Identifikatoren in den meisten Fällen aus maximal zwei Attributen bestanden. Daher terminierte die bidirektionale Breitensuche meist nach dem dritten Durchlauf, nachdem in den ersten beiden Bottom-Up-Phasen einige minimale Quasi-Identifikatoren gefunden und die anschließende Top-Down-Phase lediglich negierte QIs entdeckte.

Für die eingesetzten UbiSense-Sensoren, die von mehreren Teilnehmern während der Aufzeichnungen getragen wurden, konnte der Zeitstempel[10] als 99%-iger Quasi-Identifikator erkannt werden. Da die Daten vom Sensor mit 120 Hz erfasst und mit einer Genauigkeit innerhalb des Millisekundenbereichs gespeichert werden, ist es über den Zeitstempel möglich, eine Person über die gesamte Laufzeit zu identifizieren. Als zusätzliches Hintergrundwissen bedarf es zusätzlich entweder des genauen Aufzeichnungsstarts des UbiSense-Tags oder des „Startpunkts" der jeweiligen Person im Raum. Die Koordinaten `x`, `y` und `z` bilden jeweils einen Quasi-Identifikator bis zu einem Grenzwert von 97%. Dadurch lässt sich die Bewegung einer Person im Raum ebenso verfolgen, da die Daten mit sehr hoher Genauigkeit erfasst werden.

Als weiterer, interessanter Sensor stellten sich die vernetzten Steckdosen heraus. Die Attribute $\{MilliAmperes, EventType, id\}$ bilden einen 74%-igen Quasi-Identifikator. Das Attribut `id` stellt dabei die Identifikationsnummer der Steckdose dar, welche sich den einzelnen Sitzplätzen bzw. weiteren Positionen im vernetzten Besprechungsraum zuordnen lässt. `EventType` beschreibt die Art des Ereignisses, d. h., ob ein Gerät angeschlossen bzw. abgenommen wurde und ob eine Änderung des Stromverbrauchs (Attribut `MilliAmperes`) erfolgte. Sind durch Beobachtungen die Geräte der Teilnehmer und deren Sitzgewohnheiten (im Sinne von

[10] Im MuSAMA-Datensatz wird dieser durch die Attribute $EventMillis$ und $timestamp$ doppelt repräsentiert. $EventMillis$ spiegelt den Zeitpunkt des Sensorereignisses dar, $timestamp$ hingegen den Zeitpunkt des Speicherns des Tupels.

„Stammplätzen" o. Ä.) bekannt, lässt sich daraus eine Art unintendierte „Anwesenheitskontrolle" realisieren, die nicht im Sinne der Betroffenen ist.

Letzterer Aspekt ist nach Strufe [Kos19] sehr problematisch, „weil [das Internet der Dinge] eine immer umfassendere Beobachtung und potenzielle Überwachung der Benutzer einzieht", insbesondere „weil diese Geräte häufig keine Interfaces haben, über die man die Benutzer sinnvoll aufklären oder über die der Benutzer sinnvoll Schutzeinstellungen konfigurieren könnte". Im Speziellen gilt dies für adaptive Umgebungen, bei denen neue (Sensor-)Systeme im laufenden Betrieb hinzugefügt werden können.

Zusammenfassende Betrachtung

Der entwickelte Ansatz, mittels einer gewichteten, bidirektionalen Breitensuche Quasi-Identifikatoren zu erkennen, wurde speziell für hochdimensionale Sensordaten, wie sie bei MuSAMA erzeugt werden, entwickelt. In diesem Anwendungsszenario läuft der Algorithmus und dessen Implementierung äußerst effizient. Auch in den anderen beiden Testszenarien konnten bessere Laufzeiten als mit dem bestehenden Bottom-Up-Ansatz erreicht werden.

Überraschend fielen die eigentlichen Ergebnisse in Form der erkannten Quasi-Identifikatoren aus. Während im MuSAMA-Szenario nur wenige QIs erkannt wurden, verfügen der Zensus- und der TPC-H-Datensatz über eine große Menge an Quasi-Identifikatoren, die zudem stark überlappen. Dies wirft die Frage auf, inwieweit sich Datensätze mit vielen Quasi-Identifikatoren sinnvoll anonymisieren lassen. Eine Analyse zu dieser Fragestellung wird im folgenden Abschnitt 4.6 durchgeführt.

4.6 Anwendung für Anonymisierungsverfahren

Basierend auf den Evaluationsergebnissen im vorherigen Abschnitt können bereits für niedrigdimensionale Relationen mit weniger als 20 Attributen Hunderte von minimalen Quasi-Identifikatoren existieren. Für jeden dieser QIs ist es notwendig, mindestens ein Attribut durch Anonymisierungstechniken zu generalisieren, zu verrauschen bzw. mit verwandten Techniken zu anonymisieren. Um einen hohen Informationsgehalt zu gewährleisten, wird in der Literatur zu Anonymisierungsverfahren (siehe u. a. [MW04] und [MX07]) vorgeschlagen, eine minimale Teilmenge der Attribute zu bestimmen, welche alle Quasi-Identifikatoren überdeckt.

4.6.1 Bestimmung der minimalen Attributmenge

Als formale Grundlage hinter der Auswahl der minimalen Teilmenge verbirgt sich das *Vertex Cover Problem* auf Hypergraphen. Ein Hypergraph H ist ein Tupel der Form $H := (V, E)$, wobei durch V eine Menge von n Knoten (engl.: *vertices*) und E eine Menge von Kanten (engl.: *edges*) beschrieben werden. Die Kanten enthalten dabei eine beliebig große Teilmenge der Knoten aus V^n.

Beim ungewichteten Vertex-Cover-Problem wird eine minimale Teilmenge $V' \subseteq V$ gesucht, sodass jede Kante $e \in E$ mindestens einen Knoten $v' \in V'$ enthält. Übertragen auf die Anonymisierungsverfahren entspricht V der Attributmenge eines Relationenschemas R und E ist die Menge der Quasi-Identifikatoren über $r(R)$.

Beispiel: Quasi-Identifikatoren als Hypergraph

Betrachten wir folgende Quasi-Identifikatoren mit jeweils vier Attributen aus der Amarok-Datenbank für die Relation `tracks` ($t = 90\%$):

- $QI_1 = \{bitrate, length, tracknumber, yid\}$
- $QI_2 = \{cid, length, tracknumber, yid\}$
- $QI_3 = \{artid, cid, gid, title\}$
- $QI_4 = \{discnumber, length, tracknumber, yid\}$
- $QI_5 = \{artid, discnumber, length, yid\}$

Der Hypergraph $H = (V, E)$ setzt sich wie folgt zusammen:

- $V = \{artid, bitrate, cid, discnumber, gid, length, title, tracknumber, yid\}$
- $E = \{QI_1, QI_2, QI_3, QI_4, QI_5\}$

Nach Halperin [Hal02] entspricht das *Vertex Cover Problem* angewendet auf Hypergraphen dem *Set Cover Problem*. Für die Transformation des Problems wird in einem Eintrag C_{v_i} einer invertierten Liste pro Knoten $v_i \in V$ erfasst, in welchen Quasi-Identifikatoren $qi_j \in E$ dieser vorkommt.

Beispiel: Invertierte Liste für Quasi-Identifikatoren

Ausgehend von dem Hypergraphen H aus dem letzten Beispiel ergibt sich folgende invertierte Liste:

- $C_{artid} = \{QI_3, QI_5\}$
- $C_{bitrate} = \{QI_1\}$
- $C_{cid} = \{QI_2, QI_3\}$
- $C_{discnumber} = \{QI_4, QI_5\}$
- $C_{gid} = \{QI_3\}$
- $C_{length} = \{QI_1, QI_2, QI_4, QI_5\}$
- $C_{title} = \{QI_3\}$
- $C_{tracknumber} = \{QI_1, QI_2, QI_4\}$
- $C_{yid} = \{QI_1, QI_2, QI_4, QI_5\}$

Beim Set-Cover-Problem kann jedem Eintrag C_i ein Gewicht w_i zugeordnet werden, um dessen Kosten zu repräsentieren. Im Fall von Anfragen auf relationalen Strukturen können diese Kosten bzw. Gewichte die Relevanz der einzelnen Attribute für das Anfrageergebnis widerspiegeln. Für die Auswahl der Attribute als Parameter für die Anonymisierungsverfahren sind demnach die aufsummierten Kosten zu minimieren. O. B. d. A. setzen wir für die folgenden Betrachtungen die Kosten für jeden Knoten auf den Wert `1`, da die semantische Bedeutung der Attribute für die Anfragen unbekannt ist.

Für die Ermittlung der minimalen Kosten (bzw. der minimalen Anzahl an Attributen) zur Überdeckung aller Quasi-Identifikatoren wurde ein bereits existierender Algorithmus zur Lösung des Set-Cover-Problems von Adham Zaki verwendet[11]. Wird dem Algorithmus eine invertierte Liste mit den Attributen und den dazugehörigen Hyperkanten übergeben, berechnet dieser eine minimale Teilmenge an Attributen, welche alle Hyperkanten überdecken.

Beispiel: Minimale Überdeckungen für die Quasi-Identifikatoren

Mögliche minimale Überdeckungen für das laufende Beispiel sind beispielsweise `(yid, gid)`, `(length, title)` oder `(tracknumber, artid)`.

4.6.2 Ergebnisse

In diesem Abschnitt wird kurz auf die minimalen Überdeckungen im Zensus- und TPC-H-Datenbestand eingegangen. Der Zensus-Datensatz enthält beispielsweise die Attribute `fnlwght` (klassifiziert als sensitive Information) und `age` (als Generalisierung vom genauen Geburtsdatum der jeweiligen Person). Diese Attribute bilden bei einem Grenzwert von $t = 60\%$ die minimale Überdeckung für alle Quasi-Identifikatoren.

[11] Der in Java geschriebene Algorithmus ist unter `https://github.com/azakiio/Set-Cover-Problem-Java`, zuletzt aufgerufen am 05.01.2022, verfügbar. Im Begleitmaterial (siehe Anhang G) befindet sich eine angepasste Version, um aus den Quasi-Identifikatoren die für den Algorithmus notwendigen Datenstrukturen zu erzeugen.

Da es sich beim Zensus-Datensatz um bereits anonymisierte Daten handelt, ist das Ergebnis nicht überraschend: In der Regel sind nur noch die als sensitive Informationen gekennzeichneten Attribute (hier: `fnlwght`), welche nicht verändert wurden, Teil der QIs. Weiterhin können Attribute, die nicht ausreichend anonymisiert wurden (hier: `age`), in der Überdeckung der Quasi-Identifikatoren enthalten sein.

Bei höheren Grenzwerten ab 70% entfällt das Attribut `age` aus der minimalen Überdeckung. Da somit nur das sensitive Attribut `fnlwght` verbleibt, braucht der Datenbestand für $t \geq 70\%$ nicht weiter anonymisiert zu werden.

Für Rohdaten, d. h. Daten ohne vorherige Anonymisierung, sieht dies komplexer aus: Im TPC-H-Szenario sind in der Relation `lineitem` (Grenzwert $t = 50\% \ldots 90\%$) beispielsweise 10 der 16 Attribute notwendig, um eine minimale Überdeckung der Quasi-Identifikatoren zu erreichen. Bei höheren Grenzwerten ($t = 95\% \ldots 99\%$) werden immerhin noch neun Attribute zur minimalen Überdeckung benötigt. Die neun bzw. zehn Attribute müssen – beispielsweise – durch Generalisierung in ihrem Informationsgehalt reduziert werden.

Dadurch können allerdings neue QIs aus der Kombination der bestehenden Quasi-Identifikatoren mit weiteren Attributen entstehen [Sam01], wodurch weitere Anonymisierungen einzelner Attribute notwendig sind. Der daraus entstehende Informationsverlust kann, je nach Anwendungsszenario, zu groß sein. Dadurch kann die Funktionsfähigkeit des auf den Daten aufbauenden Informationssystems stark beeinträchtigt werden.

Eine vollständige Liste der Quasi-Identifikatoren für die Grenzwerte von 50% bis 99%, inkl. des während der Berechnung erstellten Logs, ist für alle drei Anwendungsszenarien ist im Begleitmaterial zu dieser Arbeit verfügbar. Weiterhin sind für ausgewählte QI-Mengen angepasste Versionen zur Berechnung der minimalen Überdeckung hinterlegt.

4.7 Zusammenfassung

Das in diesem Kapitel entwickelte Verfahren zur bidirektionalen Suche nach Quasi-Identifikatoren in hochdimensionalen Daten ist nicht vollständig neu entwickelt worden. In Abgrenzung zum Stand der Technik im Abschnitt 4.2 wurden folgende neue Erkenntnisse gewonnen:

- Durch die in Abschnitt 4.3 eingeführte Wichtungsfunktion wurde eine kostenbasierte Abschätzung des Berechnungsaufwands zwischen dem Bottom-Up- und dem Top-Down-Vorgehen geschaffen. Die Wichtungsfunktion erlaubt eine effizientere Traversierung innerhalb des aufgespannten Suchraums.
- Die in Abschnitt 4.3.3 vorgestellten Algorithmen zur dynamischen Berechnung der Distinct bzw. Separation Ratio garantieren, dass (negierte) Quasi-Identifikatoren trotz einer Folge von Änderungsoperationen auf dem Datenbestand weiterhin gültig sind. Dies erlaubt es, dass der Datenbestand nur periodisch auf Quasi-Identifikatoren überprüft werden muss.
- Durch die Verwendung von Java und SQL als Programmiersprachen konnte der Algorithmus zur bidirektionalen Breitensuche direkt in den JDBC-Proxy-Treiber von PArADISE (siehe Anhang F) integriert werden. Durch die Einbettung in die JDBC-Middleware wurde die Unabhängigkeit vom Datenbanksystem als auch von der konkreten Anwendung erreicht, wodurch sich das entwickelte Verfahren flexibel einsetzen lässt.

Die Evaluation des Algorithmus für die Suche nach Quasi-Identifikatoren mittels bidirektionaler Breitensuche ergab gute Messwerte bzgl. der Performance des Verfahrens. Jedoch sind die Ergebnisse der Analysen – im Sinne der gefunden Quasi-Identifikatoren – sehr ernüchternd. Wie im vorherigen Abschnitt 4.6 gezeigt, existieren für die meisten Datenbestände sehr viele Quasi-Identifikatoren mit einer sehr starken Überlappung in den einzelnen Attributen. Dadurch wird eine ausreichende Anonymisierung nur unter hohem Informationsverlust erreicht.

In Kapitel 6 wird ein alternativer Ansatz zur Anonymisierung vorgestellt: Durch die stufenweise, lokale Vorberechnung von Zwischenergebnissen einer Analyse werden unter dem Aspekt der Datensparsamkeit Daten derart vorgefiltert und -aggregiert, sodass der Personenbezug aus den Daten entfernt wird. Das vorgestellte Verfahren kann dabei mit Anonymisierungstechniken verknüpft werden, um auch für die erzeugten Zwischenergebnisse Anonymität zu gewährleisten. Zuvor geht Kapitel 5 auf die theoretischen Grundlagen für dieses Verfahren ein.

Kapitel 5

Query Containment: Stand der Forschung

Im Folgenden werden theoretische und praktische Ansätze zur Lösung des Query-Containment- und Query-Rewriting-Problems vorgestellt. Dabei werden insbesondere Konzepte zum Überprüfen von Anfragen auf Äquivalenz betrachtet. Für spezielle Anfrageklassen werden Konzepte für Anfragen mit Rekursion, Aggregaten und verschiedene Arten von Abhängigkeiten vorgestellt. Abschließend werden als Abgrenzung zum eigenen entwickelten Ansatz verwandte Konzepte für kapazitätsbeschränkte und verteilte Anfragen sowie das Prinzip *Design by Contract* vorgestellt.

5.1 Basisdefinitionen und -algorithmen für das Query Containment

Die formalen Grundlagen für das *Query Containment* und die *Query Equivalence* wurden von Chandra und Merlin in [CM77] eingeführt:

Definition: Query Containment Problem (QCP) nach [CM77]
Eine Anfrage Q_2 ist in einer Anfrage Q_1 enthalten ($Q_2 \sqsubseteq Q_1$), genau dann, wenn für jede Instanz d über dem Datenbankschema D das Ergebnis der Anfrage $Q_2(d)$ eine Teilmenge des Ergebnisses der Anfrage $Q_1(d)$ ist ($\forall d(D) : Q_2(d) \subseteq Q_1(d)$). Zwei Anfragen gelten als äquivalent ($Q_1 \equiv Q_2$), wenn sowohl $Q_2 \sqsubseteq Q_1$ als auch $Q_1 \sqsubseteq Q_2$ gilt.

In dem Beitrag wird zudem der Beweis für die Nicht-Berechenbarkeit für allgemeine Anfragen und die NP-Vollständigkeit für konjunktive Anfragen angegeben[1]. Für Anfragen mit einfachen Selektionen sowie Projektionen und Verbundbedingungen zeigen Chandra und Merlin [CM77], dass das Problem auch unter Verwendung der Mengensemantik berechenbar ist. Für Spezialfälle, z. B. wenn eine Relation maximal zweimal im Rumpf einer Anfrage auftaucht, existieren angepasste Algorithmen, die das Problem in polynomialer Zeit bzgl. der Anfragegröße lösen können [KV98]. Die NP-Vollständigkeit stellt jedoch in vielen Fällen kein Problem dar, da Anfragen, im Vergleich zu den Daten in der dazugehörigen Datenbank, vergleichsweise klein ausfallen [DHI12].

Die Probleme des Query Rewritings und des Query Containments haben in der Datenbankforschung eine zentrale Bedeutung. Vielen Forschungsfragen, wie die Optimierung von Anfragen und die Datenintegration, lassen sich auf diese beiden Probleme zurückführen.

Beim *Query Containment* wird für zwei Anfragen Q_1 und Q_2 geprüft, ob eine Anfrage in der anderen enthalten ist. Durch *Query Rewriting* wird eine Anfrage Q_2 berechnet, die eine Transformation einer gegebenen Anfrage Q_1 darstellt. Dabei soll Q_2 eine möglichst große Teilmenge des Ergebnisses von Q_1 zurückgeben und keine zusätzlichen Tupel enthalten.

[1] Für zwei Anfragen Q_1 und Q_2 lassen sich Teilabbildungen von jedem der m Literale aus Q_1 auf jedes der n Literale aus Q_2 bilden. Jede dieser m^n Kombinationen lässt sich in Polynomialzeit überprüfen. Das Query Containment Problem liegt somit in der Komplexitätsklasse NP-vollständig.

Die Abbildung zweier konjunktiver Anfragen Q_1 und Q_2 wird durch sogenannte *Containment Mappings* beschrieben. Zwecks besserer Lesbarkeit führen wir an dieser Stelle nicht die ursprüngliche Version aus Chandra und Merlin [CM77] an, sondern benutzen die Notation aus einem aktuellen Lehrbuch zur Informationsintegration [DHI12], die auf Datalog (siehe Anhang A.4) basiert:

Definition: Containment Mapping nach [DHI12]
Seien Q_1 und Q_2 konjunktive Anfragen und ψ eine Abbildung der Variablen von Q_1 auf Q_2. ψ heißt *Containment Mapping* von Q_1 auf Q_2, falls

- $\psi(X) = Y$, wobei X und Y die Kopfvariablen von Q_1 bzw. Q_2 sind.
- Für jedes Teilziel $g(X_i)$ im Rumpf von Q_1 ist $\psi(g(X_i))$ ein Teilziel in Q_2.

$Q_1 \sqsupseteq Q_2$ gilt genau dann, wenn ein Containment Mapping von Q_1 auf Q_2 vorliegt.

Formal kann das Verhältnis von Q_1 zu Q_2 nach Türker [Tür99] eine der folgenden Ausprägungen annehmen:

1. **Disjunkt:** In den Ergebnissen beider Anfragen existiert kein gemeinsames Tupel (siehe Abbildung 5.1a):
$$Q_1(R) \cap Q_2(R) \equiv \emptyset \tag{5.1}$$

2. **Äquivalent:** Im Ergebnis beider Anfragen sind die gleichen Tupel vorhanden (siehe Abbildung 5.1b):
$$\begin{aligned} Q_1(R) \equiv Q_2(R) \Leftrightarrow Q_1(R) \cap Q_2(R) \equiv Q_1(R) \\ \wedge\, Q_1(R) \cap Q_2(R) \equiv Q_2(R) \end{aligned} \tag{5.2}$$

3. **Überlappend:** Einige, aber nicht alle Tupel des Ergebnisses der Anfragen sind identisch. Die Mengen sind dabei weder äquivalent noch disjunkt (siehe Abbildung 5.1c):
$$\begin{aligned} Q_1(R) \Cap Q_2(R) \Leftrightarrow Q_1(R) \not\equiv Q_2(R) \\ \wedge\, Q_1(R) \cap Q_2(R) \not\equiv \emptyset \end{aligned} \tag{5.3}$$

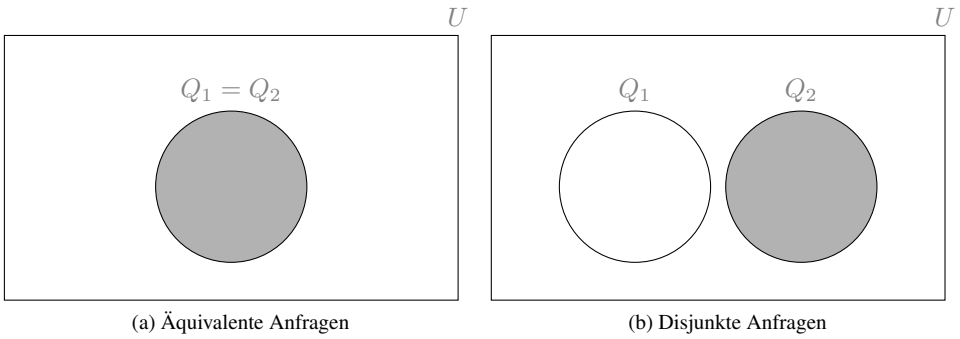
4. **Echte Teilmenge:** Das Ergebnis der Anfrage Q_1 ist in der Anfrage Q_2 enthalten, d. h., Q_2 enthält alle Tupel aus Q_1 (siehe Abbildung 5.1d):
$$\begin{aligned} Q_1(R) \subset Q_2(R) \Leftrightarrow Q_1(R) \cap Q_2(R) \equiv Q_1(R) \\ \wedge\, Q_1(R) \cap Q_2(R) \not\equiv Q_2(R) \end{aligned} \tag{5.4}$$

Überlappende Beziehungen treten nach den Ergebnissen von Türker [Tür99] schon bei einfachen Anfragetransformationen auf. Dies zeigt sich beispielsweise, wenn zwei oder mehr einfache Aggregatfunktionen kombiniert werden oder die Zählung der unterschiedlichen Tupel auf zwei verschiedenen Attributen erfolgt.

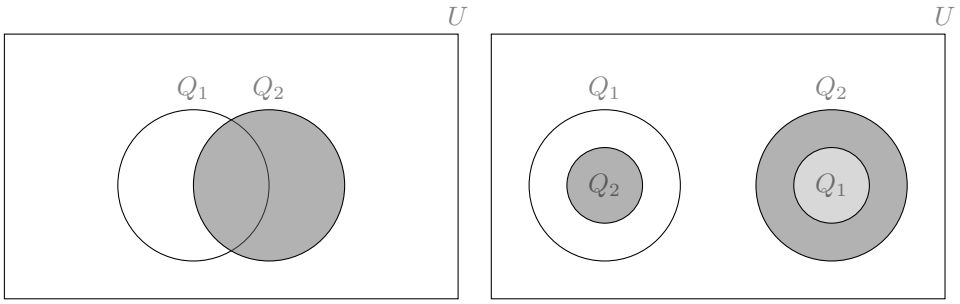
5.1.1 Answering Queries using Views

In [LMSS95] wird ein Standardfall des Query-Containment-Problems vorgestellt: *Answering Queries using Views* (AQuV). Für eine gegebene Anfrage Q_1, die auf einer Datenbank D ausgeführt wird, soll eine Anfrage Q_2 berechnet werden, die ausschließlich Sichten V_i verwendet, die auf den Relationen von D definiert wurden.

Definition: Answering Queries using Views Problem nach [LMSS95]
Sei Q_1 eine Anfrage und $V = \{V_1, \ldots, V_n\}$ eine Menge von Sichtdefinitionen. Eine Anfrage Q_2 ist eine äquivalente Umschreibung von Q_1 unter Berücksichtigung von V, wenn Q_2 nur die Sichten aus V verwendet und $Q_2(D)$ äquivalent zu $Q_1(D)$ ist.



(a) Äquivalente Anfragen (b) Disjunkte Anfragen



(c) Überlappende Anfragen (d) Enthaltene Anfragen

Abbildung 5.1: Mögliche Verhältnisse zwischen zwei Anfragen.

Äquivalente Transformationen sind für beliebige Datenbankschemata und deren Instanzen nicht immer möglich. Um eine Anfrage unter Nutzung der Sichten möglichst vollständig zu beantworten, wird nach einer Anfragetransformation von Q_1 nach Q_2 gesucht, welche die größtmögliche Teilmenge des Anfrageergebnisses zurückliefert. Dieses wird in der englischsprachigen Literatur als *Maximally Contained Rewriting*[2] bezeichnet:

Definition: Maximally Contained Rewriting nach [LMSS95]

Sei Q_1 eine Anfrage, $V = \{V_1, \ldots, V_n\}$ eine Menge von Sichtdefinitionen und L eine Anfragesprache. Eine Anfrage Q_2 ist ein *maximally contained rewriting* von Q_1 unter Beachtung von L, falls Q_2 eine in L formulierte Anfrage ist, die nur Sichten aus V verwendet, $Q_2 \sqsubseteq Q_1$ gilt und keine Anfrage $Q' \in L$ existiert, sodass $Q_2 \sqsubset Q' \sqsubseteq Q_1$ gilt.

Das folgende Beispiel verdeutlicht, inwieweit durch die Wahl der Sichten die Vollständigkeit des Anfrageergebnisses beeinflusst wird:

Beispiel: Anfragetransformationen für die Musikdatenbank

Ausgehend von den Relationen der Amarok-Musikdatenbank wurden folgende drei Sichten erzeugt:

- Sicht `jazzVar` – Länge und Titel aller Jazz-Tracks von „Various Artists”

$$\pi_{title,length}(\sigma_{aname='VariousArtists' \wedge gname='Jazz'}(tracks \bowtie artists \bowtie genres)) \tag{5.5}$$

[2]Ein deutschsprachiger Begriff hat sich hierfür nicht etabliert. Eine mögliche Übersetzung ist *maximal zurücklieferbare Ergebnismenge einer Anfragetransformation*.

- Sicht `jazzOnly` – Titel aller Jazz-Tracks

$$\pi_{title}(\sigma_{gname='Jazz'}(tracks \bowtie genres)) \tag{5.6}$$

- Sicht `varFarm` – Titel, Genre und Länge aller Tracks von „Various Artists" und der „Farm AG"

$$\pi_{title,gname,length}(\sigma_{aname='VariousArtists' \vee aname='FarmAG'}(tracks \bowtie artists \bowtie genres)) \tag{5.7}$$

Als Anfrage auf der Datenbank sollen alle Titel mit einer Laufzeit von mehr als 300 Sekunden ermittelt werden. Auf den Basisrelationen lässt sich die Anfrage wie folgt formulieren:

$$\pi_{title}(\sigma_{length>300}(tracks)) \tag{5.8}$$

Dürfen nur die zuvor erstellten Sichten anstatt der Basisrelationen zur Beantwortung der Anfrage genutzt werden, so muss die Anfrage wie folgt umformuliert werden, um das größtmögliche Ergebnis zu erhalten:

$$\pi_{title}(\sigma_{length>300}(varFarm)) \tag{5.9}$$

Die Sicht `jazzOnly` kann offensichtlich nicht zur Beantwortung der Anfrage verwendet werden, da auf ihr keine Selektion auf dem Attribut `length` mehr erfolgen kann. Auf der Sicht `jazzVar` ist diese Selektion weiterhin möglich, allerdings stellt sie eine Teilmenge von `varFarm` dar, da sie ein zusätzliches Selektionsprädikat enthält und das bestehende Prädikat auf dem Attribut `aname` in `varFarm` durch die Oder-Verknüpfung weniger restriktiv ist.

Der in [LMSS95] vorgestellte Algorithmus sucht einerseits nach minimalen Transformationen, d. h., nach transformierten Anfragen mit möglichst wenigen Verbundoperationen. Außerdem sollen die transformierten Anfragen, sofern umsetzbar, vollständig sein, d. h., vollständig auf den materialisierten Sichten statt auf den Basisrelationen ausgeführt werden. Im Kern bildet der Algorithmus die Sichten auf die Anfragen durch spezielle *Containment Mappings* ab. Dabei wird untersucht, wie die Attribute und Prädikate der Anfrage und der Sichten aufeinander abgebildet werden können. Aufgrund der Zahl der möglichen *Containment Mappings* und der Literale in der Originalanfrage, welche nicht mehr für die Sichten benötigt werden, ist das Problem nach Analyse der Autoren selbst für konjunktive Anfragen ohne Vergleichsprädikate NP-vollständig.

Einsatz von AQuV: Answering Queries Using Views hat viele Anwendungsgebiete: In der Datenintegration wird es als Grundtechnik eingesetzt, um möglichst viele, lokal vorhandene Daten in eine globale Datenbank zu überführen. Zudem können bei der Informationsintegration die Sichten für Wrapper und Mediatoren genutzt werden. In Data Warehouses werden diese Techniken genutzt, um Datenbankanfragen auf materialisierten Sichten zu formulieren. Dies führt zu einer Effizienzsteigerung des Datenbanksystems, da auf vorberechnete Zwischenergebnisse zurückgegriffen werden kann.

Komplexität von AQuV: In der nachfolgenden Literatur zum Stand der Forschung erfolgt meistens eine Klassifizierung bezüglich der unterstützten Anfragesprachen. Zu den am häufigsten betrachteten Klassen gehören konjunktive Anfragen (Projektion, mehrere UND-verknüpfte Selektionen auf Gleichheit, Verbund), sowie die Klassen SPJ (alle Selektionen, Projektion, Verbund), SPJU (zusätzlich mit Vereinigung) und SPJUA (zusätzlich Aggregation und Gruppierung). Außerdem verwendet SQL eine Multimengensemantik, die nicht durch die vorgestellten Verfahren abgedeckt wird.

Der Grund für die Unterscheidung in die verschiedenen Klassen ist die Nichtberechenbarkeit der Äquivalenz zweier beliebiger Anfragen. Für einfache, konjunktive Anfragen ist der Test auf Äquivalenz ohne Probleme möglich. Komplexe Anfragen bestehen jedoch aus Mengenoperationen, Aggregaten und Integritätsbedingungen. Die Terminierung der Algorithmen zur Bestimmung der Äquivalenz, beispielsweise durch Anwendung des CHASEs

auf Tableaus, ist in diesen Fällen nicht gesichert [FKMP03]. In der Literatur werden die vorgestellten Lösungsansätze daher nur auf eine bestimmte Anfrageklasse angewendet, bzw. die Klassen werden durch die vorgestellten Ansätze feiner unterteilt, da neue Möglichkeiten zur Bestimmung des Verhältnisses einzelner Spezialfälle entdeckt wurden.

5.1.2 Weiterführende Algorithmen für AQuV

Im Folgenden wird ein kurzer Überblick über bestehende Konzepte und Algorithmen zu Answering Queries using Views gegeben. Für eine detaillierte Übersicht sei auf [Hal01] und [Vas09] verwiesen.

Bucket: Der Bucket-Algorithmus [LRO96a] transformiert eine gegebene konjunktive Anfrage, sodass statt den Basisrelationen nach der Anfragetransformation ausschließlich vordefinierte, relevante Sichten verwendet werden. Ausgehend von den einzelnen Teilzielen der Anfrage wird bestimmt, welche Sichten für das Erreichen des jeweiligen Teilzieles notwendig sind. Dadurch wird die Anzahl der möglichen Anfragetransformationen reduziert, da nur relevante Sichten betrachtet werden.

Im ersten Schritt des Algorithmus wird für jedes Datenbankprädikat der Anfrage Q ein *Bucket* erzeugt. Jedem der n Buckets werden die zur Beantwortung des Prädikates relevanten Sichten zugeordnet. Im zweiten Schritt des Bucket-Algorithmus wird das kartesische Produkt der Buckets gebildet, indem je ein Prädikat p_j aus jedem Bucket j zu einer konjunktiven Anfrage $Q_i := p_1 \wedge p_2 \cdots \wedge p_n$ verknüpft wird. Gilt für Q_i, dass $Q_i \sqsubseteq Q$ erfüllt ist, so wird Q_i als Teil des Ergebnisses mit allen anderen Teilanfragen, welche diese Bedingung erfüllen, vereinigt: $Q' := \bigcup_i Q_i$. Q' stellt somit das *Maximally Contained Rewriting* nach [LMSS95] dar.

Das Verfahren hat diverse Nachteile: Einerseits werden viele unnötige Tests ausgeführt, da die Zusammenhänge zwischen den einzelnen Prädikaten nicht berücksichtigt werden. Dadurch kann z. B. die Kombination von zwei Prädikaten zu einer Kontradiktion führen, wodurch keine Tupel zurückgegebenen werden. Diese Arten von Sichten sind daher zur Beantwortung der Anfrage unbrauchbar. Eine Verbesserung des Verfahrens wird durch den MiniCon-Algorithmus eingeführt.

MiniCon: Die Idee hinter dem MiniCon-Algorithmus [PL00, PH01] ist es, die Kombination unnötiger Teilanfragen oder Regeln, wie es beim Bucket- [LRO96a] bzw. beim Inverse-Rules-Algorithmus [DG97, DGL00] (siehe Abschnitt 5.2) der Fall ist, zu vermeiden. Es wird stattdessen berücksichtigt, wie die Variablen aus der Originalanfrage in den Sichten verwendet werden können. Durch diesen Ansatz müssen weniger Kombinationen von Sichten zur Bestimmung der transformierten Anfrage beachtet werden, wodurch der potentielle Suchraum stark eingeschränkt wird.

In [PL00] wird der MiniCon-Algorithmus zur Anfragetransformation für SPJ-Anfragen mit einfachen Vergleichsoperatoren unter Betrachtung der Mengensemantik vorgestellt. Im ersten Schritt werden, wie beim Bucket-Algorithmus, die relevanten Sichten bestimmt. Dabei wird jedoch vermerkt, welche Teilziele von der Originalanfrage zu den Teilzielen der Sichten gehören. Anschließend werden über gleiche Variablen Verbundbedingungen zu anderen Prädikaten gesucht und die Prädikate sowie Abbildungen von Variablen pro Sicht v_i in *MiniCon Descriptions* (MCD) erfasst.

Definition: MiniCon Description nach [PL00]

Eine MiniCon Description (MCD) für eine Anfrage Q besteht pro Sicht $v_i \in V$ aus

- einem Kopf-Homomorphismus h über v_i,
- einer partiellen Abbildung φ, welche die Prädikate von Q in die Prädikate aus v_i abbildet und
- einer Teilmenge G der Teilziele aus Q ($G \subseteq pred(Q)$), die durch ein Teilziel in $h(V)$ und die Abbildung φ abgedeckt werden.

Zur Vermeidung von redundanten Transformationen von Q müssen zusätzlich folgende Bedingungen an eine MCD gelten:

1. Für jede Kopfvariable x von Q, welche im Wertebereich von φ liegt, gilt, dass $\varphi(x)$ eine Kopfvariable in $h(V)$ ist.
2. Für jedes Teilziel $g_i \in G$ mit einer existenzquantifizierten Variable $\varphi(x)$ in $h(V)$ müssen alle Variablen aus g_i im Wertebereich von φ liegen. Weiterhin gilt, dass $\varphi(G)$ Element von $h(V)$ ist.

Durch die Erstellung der MCDs werden unbrauchbare Sichten sofort eliminiert, um die Anzahl an zusätzlichen Teilzielen zu minimieren. Dies verringert den Suchraum für die Anfragetransformation massiv, wodurch der Algorithmus deutlich effizienter arbeitet als der Bucket-Algorithmus.

Im zweiten Schritt erfolgt die Kombination der Abbildungen zur transformierten, konjunktiven Anfrage. Ein zusätzlicher Test bzgl. der Containment-Eigenschaft ($\sqsubseteq$) ist nicht notwendig, da durch die Konstruktion der MCDs der MiniCon-Algorithmus diese Eigenschaft gewährleistet. Abschließend erfolgt die Vereinigung der erzeugten konjunktiven Anfragen, wodurch ein *maximally-contained rewriting* – bei Beibehaltung der korrekten Semantik – von Q zu Q' erreicht wird.

Beispiel: Ablauf des MiniCon-Algorithmus

In diesem Beispiel betrachten wir den Ablauf des MiniCon-Algorithmus in Anlehnung an die Musikdatenbank. Als Notationsform wird zur besseren Nachvollziehbarkeit die Datalog-Schreibweise genutzt. Wir gehen nachfolgend von einer Anfrage Q und drei Sichtdefinitionen, V_1, V_2 und V_3, aus:

- $Q(x) := cover(x, y) \wedge cover(y, x) \wedge gleichesGenre(x, y)$
- $V_1(a) := cover(a, b) \wedge cover(b, a)$
- $V_2(c, d) := gleichesGenre(c, d)$
- $V_3(e, g) := cover(e, f) \wedge cover(f, g)$

Die Anfrage Q besteht aus den drei Teilzielen $g_1 = cover(x, y)$, $g_2 = cover(y, x)$ und $g_3 = gleichesGenre(x, y)$. Daraus ergeben sich folgende MCDs:

v_i	φ	h	$h(v_i)$	$g_i \in G$
V_1	—	—	—	—
V_2	$x \to c, y \to d$	$c \to c, d \to d$	$V_2(c, d)$	g_3
V_3	$x \to e, y \to f$	$e \to e, g \to e$	$V_3(e, e)$	g_1, g_2

φ bildet zunächst die Variablen der Anfrage (x und y) auf die Variablen der jeweiligen Sichten ab. Durch h werden die Variablen in den Sichten abgebildet; die einzige Änderung erfolgt hierbei in V_3, um das gegenseitige Covern abzubilden.

Die Sicht V_1 erfüllt nicht die Anforderungen an eine MCD, da im Kopf der Anfrage nicht alle zum Verbund notwendigen Variablen enthalten sind. Werden die Sichten $h(V_2)$ und $h(V_3)$ konjunktiv zur neuen Sicht $V_{2,3}$ vereinigt, so deckt diese alle drei Teilziele der Anfrage Q ab. Da $V_{2,3}$ keine zusätzlichen Teilziele enthält, ist sie äquivalent zur Anfrage Q.

Erweiterungen von MiniCon: In [BCH13] wird der Einsatz des MiniCon-Algorithmus zur Durchsetzung von Zugriffskontrollen durch zusätzliche Selektionen und Projektionen in Form von materialisierten Sichten vorgestellt. Über eine im Datenbanksystem hinterlegte Menge von Sichten hinaus werden weitere feingranulare Zugriffsrechte als Template für zusätzliche Zugriffssichten generiert. Wird eine Anfrage gestellt, so werden nur Daten zurückgegeben, welche über die Zugriffssichten auf den materialisierten Sichten erfragbar sind. Die Autoren bezeichnen dieses Vorgehen als *double rewriting*, da die gestellte Anfrage zweifach transformiert werden muss, damit sie auf den Basisrelationen ausführbar ist.

Der Umfang der erlaubten Anfrageklassen beschränkt sich dabei, ähnlich zum MiniCon-Algorithmus, auf konjunktive Anfragen und den Gleichheits-Operator. Im Unterschied zum MiniCon-Algorithmus können im Anfragekopf auch Variablen vorkommen, die nicht durch die partielle Abbildung φ verwendet wurden. Dadurch sichert der Algorithmus ab, dass das Anfrageergebnis maximal im Sinne der zurückgegebenen Attributmenge ist.

Anwendung von Anfragetransformationen auf materialisierten Sichten: Query-Containment-Probleme spielen eine wichtige Rolle bei der Erzeugung von materialisierten Sichten. Materialisierte Sichten sollen dazu dienen, Anfragen zu beschleunigen, indem häufig genutzte Teilausdrücke der Anfragen vorberechnet werden. Ein konkretes Verfahren, welches häufig genutzte Teilbäume in einer Menge von Anfragen identifiziert, um daraus materialisierte Sichten zu erzeugen, ist in [JKRP18] beschrieben.

Das Verfahren beruht auf induktiver logischer Programmierung, um aus gegebenen Beispielanfragen einen bipartiten Graphen zu erzeugen. In dem Graphen wird anschließend parallel und iterativ nach gleichen, maximalen Teilbäumen gesucht. Die Teilbäume repräsentieren diejenigen Teile der Anfragen, die als materialisierte Sichten gespeichert werden. Durch eine Kosten-Nutzen-Funktion erfolgt eine zusätzliche Filterung, um nur die Anfragen zu materialisieren, die zu einer wirklichen Verringerung der Anfrageauswertung führen. Die Teilanfragen werden abschließend im internen Anfrageoptimierer hinterlegt, um Anfragetransformationen für zukünftige Anfragen bereitzustellen. Laut den Evaluierungsergebnissen von [JKRP18] lässt sich durch das Verfahren die Laufzeit von Anfragen um bis zu 40% reduzieren.

Query Folding: Query Folding beschreibt den Vorgang, wie eine Anfrage beantwortet werden kann, wenn lediglich materialisierte Sichten oder die gecacheten Ergebnisse von vorherigen Anfragen zur Verfügung stehen. Wie die anderen bisher vorgestellten Ansätze auch, kann durch Query Folding die Optimierung von Datenbankanfragen auf zentralisierten, parallelen oder föderierten Datenbanken realisiert werden. In [Qia96] wird ein Algorithmus mit exponentieller Laufzeit vorgestellt, welcher alle vollständigen oder zumindest partiellen Foldings (zu Deutsch: Entfaltungen) findet. Qian unterscheidet dafür vier Arten von Anfragetransformationen: partielle, maximale, vollständige und strenge Transformationen.

Eine partielle Transformation der Anfrage Q unter der Ausnutzung der Sichten V ist eine konjunktive Anfrage Q', sodass $Q' \subseteq Q$ gilt und der Kopf von Q' mindestens ein Prädikat aus V enthält. Die Transformation ist maximal, wenn keine weitere partielle Transformation existiert, die aus weniger Literalen besteht. Eine Transformation von Q unter der Ausnutzung von V ist vollständig, wenn im Rumpf der Anfrage nur Prädikate aus V vorkommen. Eine Transformation von Q unter der Ausnutzung von V heißt streng, wenn sie partiell ist und Q vollständig enthält.

Im ersten Schritt wird die Anfrage Q in seinen Hypergraphen G_Q überführt. Dabei wird für jedes Label L_e einer Hyperkante $e \in G_Q$ und dem dazugehörigen Literal eine Relation mit den Attributen aus p zugeordnet. Die Abbildungen der Sichten auf die Basisrelationen, im Artikel von Qian als *Folding Rules* bezeichnet, werden als zusätzliche Tupel mit neu ausgezeichneten Variablen in die Labels übernommen.

Der abschließende zweite Schritt erzeugt die eigentliche Anfragetransformation. Dazu werden die Labels des Hypergraphen schrittweise durch mehrere Verbunde miteinander vereinigt. Da die Sichten zuvor als Labels eingefügt wurden, ersetzen diese nun, sofern möglich, die Basisrelationen im Hypergraph[3].

Der entwickelte Algorithmus findet in Exponentialzeit alle vollständigen und partiellen Transformationen zur Beantwortung von konjunktiven Anfragen auf Sichten. Zudem wurde ein Algorithmus entwickelt, der in Polynomialzeit prüft, ob eine partielle oder vollständige Transformation für azyklische Anfragen vorliegt.

5.2 Transformation von Anfragen mit Rekursion

In diesem Abschnitt wird mit dem *Inverse-Rules*-Ansatz ein Verfahren zur Optimierung von rekursiven Anfragen beschrieben. Die zweite vorgestellte Technik verbindet rekursive Anfragen mit Aggregatfunktionen.

[3]Damit nicht die Basisrelationen die Sichten ersetzen, werden die Labels der Sichten laut Qian [Qia96] *bevorzugt* bei der Vereinigung verwendet.

Answering Recursive Queries Using Views: In den Artikeln *Answering Recursive Queries Using Views* [DG97] und *Recursive Query Plans for Data Integration* [DGL00] von Duschka et al. wird die Beantwortung von konjunktiven Anfragen auf (materialisierten) Sichten um den Aspekt der Rekursion erweitert. Der vorgestellte Algorithmus erzeugt aus einer Anfrage Q und einer Menge von Sichten V eine neue Anfrage Q_V, sodass (i) als Prädikate in Q_V nur die Sichten aus V zum Einsatz kommen, (ii) $Q_V \sqsubseteq Q$ gilt und (iii) für jede weitere Anfrage Q'_V gilt, dass $Q'_V \sqsubseteq Q_V$ erfüllt ist.

Diese Konstruktion der transformierten Anfrage liefert – sofern sie existiert – stets eine äquivalente Anfrage. Die Konstruktion besteht aus zwei Schritten: Im ersten Schritt werden sogenannte inverse Regeln aufgestellt, welche pro Relation aufzeigen, in welchen Sichten die Regeln verwendet werden.

Definition: Inverse Regeln nach [DGL00]

Sei p eine Sicht oder Anfrage mit n Prädikaten der Form $p(X) :- p_1(X_1), \ldots p_n(X_n)$. Dann ist für $\forall i \in \{1, \ldots, n\} : p_i(X'_i) :- p(X)$ eine inverse Regel von p. In X'_i werden die Variablen aus X direkt übernommen. Alle Variablen X_j, die im Rumpf, aber nicht im Kopf von p vorkommen, werden durch ein Funktionssymbol $f_{p,j}(X)$ in X'_i ersetzt.

Dabei werden ggf. Funktionssymbole als Skolemfunktionen in den Köpfen der Regeln für existenzquantifizierte, nichtausgezeichnete Variablen eingeführt. Diese Funktionssymbole zeigen somit auf, welche Informationen aus den Sichten gewonnen werden können. Aus der ursprünglichen Anfrage Q werden anschließend alle Prädikate entfernt, die in keiner Sichtdefinition vorkommen. Jede inverse Regel stellt dabei ein Teilziel aus der Sichtdefinition dar. Bei der abschließenden Auswertung der Anfrage werden in die verbleibenden Prädikate die inversen Regeln eingesetzt, wodurch die Basisrelationen durch die Sichten ersetzt werden. Aus der konjunktiven Verknüpfung von der Originalanfrage Q zusammen mit den inversen Regeln ergibt sich eine maximale Transformation von Q.

Beispiel: Ablauf des Inverse-Rules-Algorithmus

Wir gehen zunächst von folgenden Anfragen und Sichten aus:

- $Q(X,Y) := cover(X,Y), gleichesGenre(X,Y)$
- $Q(X,Y) := cover(X,Z), gleichesGenre(X,Z), Q(Z,Y)$
- $V_1(X,Y) := cover(X,Y), gleichesGenre(X,Y)$
- $V_2(X,Y) := cover(X,Z), Q(Z,Y)$

Daraus ergeben sich folgende invertierte Sichten V^{-i}:

- $V^{-1} := cover(X,Y) \leftarrow V_1(X,Y)$
- $V^{-2} := gleichesGenre(X,Y) \leftarrow V_1(X,Y)$
- $V^{-3} := cover(X,Y) \leftarrow V_2(X,Y)$
- $V^{-4} := cover(f_{V_2}(X,Y),Y) \leftarrow V_2(X,Y)$

Werden die invertierten Sichten zur Anfrage hinzugefügt, ergibt sich folgende transformierte Anfrage:

- $Q(X,Y) := cover(X,Y), gleichesGenre(X,Y)$
- $Q(X,Y) := cover(X,Z), gleichesGenre(X,Z), Q(Z,Y)$
- $cover(X,Y) := V_1(X,Y)$
- $gleichesGenre(X,Y) := V_1(X,Y)$
- $cover(X,Y) := V_2(X,Y)$
- $cover(f_{V_2}(X,Y),Y) := V_2(X,Y)$

Werden die zusätzlichen Regeln zur Beantwortung von Q eingesetzt und anschließend das dabei entstandene Funktionssymbol $f_{V_2}(X, Y)$ aufgelöst, so entstehen mehrere konjunktive Anfragen, welche lediglich aus den Sichtdefinitionen bestehen. In diesem Beispiel deckt die Sicht V_1 sowohl die erste Regel von Q vollständig ab, die zweite Regel nur teilweise. Die Sicht V_2 deckt die restlichen Teilziele der zweiten Regeln von Q ab.

Durch die Einbindung von funktionalen Abhängigkeiten als Integritätsbedingungen wird die Anfragetransformation zwar erschwert, ermöglicht es aber, weiterführende Anfragetransformationen zu realisieren. Zudem wird gezeigt, dass nicht immer ein nicht-rekursiver, maximal enthaltender Anfrageplan existiert.

Für die Erweiterung um rekursive Anfragen auf konjunktiven Sichten kommen die Autoren zu dem Schluss, dass das *Query Containment Problem* für die Äquivalenz von Anfragen unentscheidbar ist. Für praktische Anwendungen, wie sie beispielsweise bei der Informationsintegration vorkommen, reichen aber *maximally-contained* Anfragen aus, die sich in Polynomialzeit bzgl. der Anzahl der Teilziele erstellen lassen. Diese enthalten nicht das gesamte Ergebnis, jedoch eine ausreichend große Teilmenge.

Fixpoint Semantics and Optimization of Recursive Datalog Programs with Aggregates: Zaniolo et al. [ZYI+17] beschreiben eine Erweiterung von Datalog um Aggregatfunktionen und Rekursion. Sie verbinden dazu die Fixpunktsemantik von Horn-Klauseln mit bestehenden Optimierungstechniken von Datalog-Programmen. Während sich die Aggregatfunktionen *Minimum* bzw. *Maximum* in rekursiven Anfragen auf Multimengen problemlos verwenden lassen, können die Aggregate *Summe* nur über einem positiven Wertebereich[4] und *Anzahl* nur unter Mengensemantik effizient ausgewertet werden, da ansonsten kein Fixpunkt aufgrund der fehlenden Monotonie-Eigenschaft gefunden werden kann[5].

5.3 Transformation von Anfragen mit Aggregaten und komplexen Vergleichen

In diesem Abschnitt werden Ansätze zur Lösung des Query-Containment-Problems für komplexere Arten von Anfragen vorgestellt. Der Fokus liegt dabei auf Aggregatfunktionen und arithmetischen Operatoren.

Semantic Integrity Constraints in Federated Database Schemata: In seiner Doktorarbeit [Tür99] zeigt Can Türker, wie für zwei Mengen von komplexen Integritätsbedingungen Φ und Ψ das Verhältnis zueinander bestimmt werden kann. Dabei wird zunächst das Verhältnis von zwei einzelnen Bedingungen $c_1 \in \Phi$ und $c_2 \in \Psi$ überprüft. Für c_1 und c_2 existieren folgende fünf Beziehungen:

1. Disjunktheit: $c_1 \cap c_2 = \emptyset$,
2. Äquivalenz: $c_1 \equiv c_2$,
3. c_2 enthält c_1: $c_1 \sqsubset c_2$,
4. c_1 enthält c_2: $c_2 \sqsubset c_1$,
5. c_1 und c_2 überlappen, d. h., das Verhältnis kann nur aus dem konkreten Datenbestand, nicht jedoch aus der Anfrage selbst bestimmt werden.

Als Klasse von Anfragen betrachtet Türker zunächst linear-arithmetische Bedingungen (engl.: Linear Arithmetic Constraints, LAC) mit den Vergleichsoperatoren $<, \leq, =, \neq, \geq$ und $>$. Diese unterteilt er in die vier Unterklassen (i) Attribut-Wert-Vergleiche ($LAC1$), (ii) Attribut-Attribut-Vergleiche ($LAC2$), (iii) Attribut-Wert- und Attribut-Attribut-Vergleiche mit Multiplikation über $\mathbb{Z}$ ($LAC3$) und (iv) Attribut-Wert- und Attribut-Attribut-Vergleiche mit Multiplikation und Addition von Konstanten über $\mathbb{Z}$ ($LAC4$).

[4] Über einen positiven Wertebereich verfügen z. B. die Menge der natürlichen Zahlen und die Teilmenge der reellen Zahlen, deren Werte größer-gleich 0 sind.

[5] Allgemeine Terminierungskriterien für Anfragen bzw. Trigger lassen sich u. a. in [WH95] und [AHW95] finden.

Beispiel: Äquivalenz zweier arithmetischer Vergleichsoperatoren

Zu der Klasse LAC1 gehört folgende äquivalente Transformation, bei der die Länge eines Musikstückes mit der Konstanten `42` verglichen wird:

$$\begin{aligned} x < c &\equiv x \leq c \wedge x \neq c \\ length < 42 &\equiv length \leq 42 \wedge length \neq 42 \end{aligned} \tag{5.10}$$

Ist die Länge kleiner als 42, dann gilt auch, dass die Länge kleiner-gleich 42 *und* ungleich 42 ist.

Um die Beziehung zwischen zwei Mengen von $LAC2$-Bedingungen zu bestimmen, führt Türker einen graphbasierten Ansatz ein. Der Algorithmus testet die Integritätsbedingungen auf sogenannte *stark verbundene Komponenten*. Jede Komponente stellt einen Knoten im Graph dar, der wiederum eine Variable aus der Anfrage repräsentiert. Existiert zwischen zwei Variablen x und y ein $\leq$- oder $\geq$-Operator, so werden diese mittels einer gerichteten Kante verbunden. Die stark verbundenen Komponenten zeichnen sich dadurch aus, dass diese durch eine Rückkante (ggf. auch durch transitive Beziehungen) miteinander verknüpft sind. Existiert parallel dazu eine Verbindung zwischen x und y mittels $\neq$-Operator, so sind die Integritätsbedingungen nicht erfüllbar.

Für das Testen der Integritätsbedingungen der $LAC3$- und $LAC4$-Klassen führt Türker gewichtete Kanten ein. Das Gewicht entspricht den jeweiligen Konstanten der arithmetischen Bedingungen. Ist die Summe der kürzesten Pfade zwischen je zwei eng verbundenen Knoten negativ, so sind die Bedingungen unerfüllbar. Diese Ansätze werden in späteren Kapiteln der Arbeit von Türker für Anfragen mit einfachen Aggregatfunktionen, Inklusionsabhängigkeiten und funktionalen Abhängigkeiten erweitert.

Definition: Verhältnis zweier Mengen von Integritätsbedingungen nach Türker [Tür99]

Nach Türker sind zwei Mengen von Integritätsbedingungen Φ und Ψ äquivalent, wenn die Mengen der Datenbankzustände U, welche Φ bzw. Ψ erfüllen, äquivalent sind:

$$\Phi \equiv \Psi :\Leftrightarrow U^{\Phi} = U^{\Psi}.$$

Sind die Datenbankzustände U^{Φ} und U^{Ψ} disjunkt, so sind die dazugehörigen Integritätsbedingungen Φ und Ψ ebenfalls disjunkt:

$$\Phi \oslash \Psi :\Leftrightarrow U^{\Phi} \cap U^{\Psi} = \emptyset.$$

Ähnliches gilt für die Containment-Eigenschaft: Eine Integritätsbedingung Ψ ist in einer Integritätsbedingung Φ enthalten, wenn die Menge der Datenbankzustände, welche Φ erfüllen, eine Teilmenge der Datenbankzustände, welche Φ erfüllen, ist:

$$\Phi \supset \Psi :\Leftrightarrow U^{\Phi} \subset U^{\Psi}.$$

Gilt keine der vier Beziehungstypen (Äquivalenz, Disjunkheit und Containment in beide Richtungen), so gelten die Integritätsbedingungen Φ und Ψ nach Türker als überlappend:

$$\Phi \Cap \Psi :\Leftrightarrow \neg((\Phi \equiv \Psi) \vee (\Phi \oslash \Psi) \vee (\Phi \supset \Psi) \vee (\Phi \subset \Psi)).$$

Ausgehend von den obigen Definitionen erarbeitete Can Türker ein Regelwerk, mit denen das Verhältnis zwischen verschiedenen Integritätsbedingungen für verschiedene Anfrageklassen bestimmt werden kann. Im Folgenden beschränken wir uns auf die zwei für diese Arbeiten wichtigsten Anfrageklassen: Die linear arithmetischen Bedingungen und die Bedingungen an Aggregatsfunktionen (engl.: Aggregate Constraints). Für letzter Klasse wird an dieser Stelle nur ein exemplarischer Ausschnitt der Regelmenge aufgeführt:

1. $min(X) = c \Rightarrow max(X) \geq c$
2. $min(X) = c \wedge c \geq 0 \Rightarrow sum(X) \geq c$
3. $\forall x \in X : x = c \wedge c \geq 0 \Rightarrow sum(X) \geq c$
4. $min(X) \phi c \wedge \phi \in \{>, \geq\} \Rightarrow \forall x \in X : x \phi c$

Die erste Regel sagt aus, dass wenn das Minimum für ein Attribut X dem Wert c entspricht, dann muss auch gelten, dass das Maximum von X größer oder gleich c ist. Gleiches gilt für die Summe; hierbei muss jedoch zusätzlich gelten, dass der Wert von c größer als 0 ist. Die Regeln 3 und 4 beschäftigen sich mit Allquantoren und deren Abbildung auf bzw. von Aggregatfunktionen.

Eine vollständige Auflistung der Transformationsregeln befindet sich im Anhang C. Diese wurden dort bereits so aufbereitet, dass sie mit dem eigenen entwickelten Ansatz, skizziert in Abschnitt 6.1, kompatibel sind.

Large-Scale Content-Based Publish/Subscribe Systems: In [Müh02, MFB02] werden viele Ansätze zur Überprüfung von Query Containment-Eigenschaften für Subskriptionen in großen Publish/Subscribe-Systemen gezeigt. Ziel für diese Systeme ist es, durch das Erkennen gemeinsamer, inhaltsbasierter Filter die Größe von Routing-Tabellen zu verkleinern und die Anzahl der zu übertragenden Kontrollnachrichten zu verringern. Die Visualisierung der Containment-Eigenschaften dieser Filter ist in Abbildung 5.2 dargestellt.

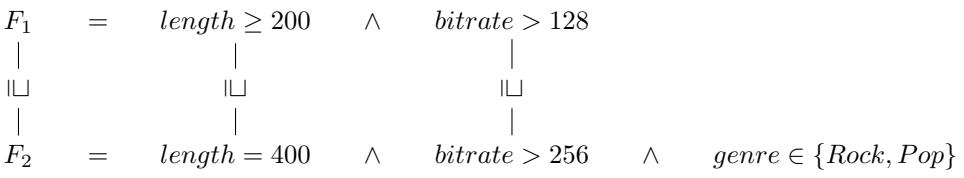


Abbildung 5.2: Visualisierung eines überdeckenden Filters in Anlehnung an Mühl [Müh02].

Die von Mühl entwickelten Ansätze lassen sich dabei leicht auf relationale Strukturen übertragen. Ähnlich zu der Arbeit von Türker [Tür99] stellt Mühl Query-Containment-Techniken für Vergleichsprädikate, inkl. Tests auf Gleichheit, Ungleichheit und Intervalle, vor. Weiterhin führt Mühl Containment-Tests für die Beziehungen zwischen Mengen, Funktionen auf Zeichenketten (Prä- und Postfixe, sowie Teilzeichenketten) und Beziehungen zwischen Ober- und Untertypen auf.

Die in Abschnitt 2.1 vorgestellten Assistenzsysteme basieren schwerpunktmäßig auf der Analyse numerischer Daten. Das in Kapitel 5 entwickelte Konzept vernachlässigt daher Operatoren auf Zeichenketten. Die Ansätze von Mühl [Müh02, MFB02] eignen sich aber für die Weiterentwicklung des in Anhang C vorgestellten Regelwerks durch zukünftige Arbeiten.

Decidability of Equivalence of Aggregate Count-Distinct Queries: In [HT15] wird ein Polynomialzeitalgorithmus ($O(n^3)$) zur Überprüfung der Äquivalenz von konjunktiven `COUNT-DISTINCT`-Anfragen, einschließlich Vergleichsoperatoren, vorgestellt. Der von den Autoren vorgeschlagene Algorithmus überprüft zunächst die sogenannte *Kern*-Eigenschaft (engl.: *core*), bei der die gegebenen konjunktiven Anfragen ohne die Aggregatfunktion auf eine isomorphe Abbildung getestet werden. Dies stellt jedoch weder eine notwendige noch eine ausreichende Bedingung für die Äquivalenz der `COUNT-DISTINCT`-Anfragen dar. Als notwendige und ausreichende Bedingung wird die Umkehrung (engl.: *flip*) der beiden Anfragen benötigt. Dazu werden die Vergleichsoperatoren in den Anfragen umgekehrt. Besteht eine isomorphe Abbildung zwischen der ersten Anfrage und der zweiten Anfrage bzw. deren Umkehrungen, so sind die `COUNT-DISTINCT`-Werte beider Anfragen äquivalent.

Die in [HT15] vorgestellten Ansätze wurden mit in das in Anhang C aufgeführte Regelwerk eingearbeitet. Dies betrifft die Transformationsregeln, bei denen Allquantoren durch Aggregationen ersetzt werden.

Privacy-Preserving Query Execution using a Decentralized Architecture and Tamper Resistant Hardware: In [TNP14] wird ein Verfahren vorgestellt, mit dem SQL-Anfragen an Edge-Geräte verteilt werden können. Das Verfahren konzentriert sich auf Anfragen mit Aggregation und Gruppierung, unterstützt jedoch nicht den Verbundoperator. Der Anfragesteller erhält das Ergebnis für die vorgegebene Anfrage; er hat jedoch keinen Zugriff auf die Rohdaten oder mögliche Zwischenergebnisse.

Das für den Algorithmus entwickelte Protokoll arbeitet in zwei Phasen: In der ersten Phase werden die erfassten Sensordaten solange gefiltert und in Partitionen gesammelt, bis eine Mindestzahl an Tupeln erreicht ist. Fallen in einem gewissen Zeitraum zu wenige Tupel an, so können die Partitionen auch mit Dummy-Tupeln aufgefüllt werden. Vor dem Versenden der Daten an den nächst höheren Knoten werden die Daten verschlüsselt.

In der zweiten Phase werden die Daten der einzelnen Partitionen auf dem übergeordneten Knoten zusammengeführt. Dabei können die Dummy-Tupel herausgefiltert werden, wenn genügend Daten vorhanden sind. Beide Phasen werden solange wiederholt, bis das Ergebnis der Berechnung dem Anfragenden vorliegt.

Die Autoren erweitern den Ansatz zusätzlich um Aggregatfunktionen. Dafür wird die zweite Phase leicht modifiziert: Nach Entfernung der Dummy-Tupel werden, sofern die Aggregatfunktion verteilbar ist, partielle Aggregate berechnet. Die wird solange wiederholt, bis alle Tupel pro Gruppe verarbeitet wurden. Werden die Dummy-Tupel und die Verschlüsselung der Daten außen vor gelassen, so entspricht das modifizierte Protokoll in seinen Grundzügen TinyDB (siehe [MFHH02] und Anhang A.5). Eine Ausweitung dieses Konzepts für komplexere Aggregatfunktionen wird in Unterabschnitt 6.1.4 des nachfolgenden Kapitels vorgestellt.

Acceleration of SQL Restrictions and Aggregations through FPGA-Based Dynamic Partial Reconfiguration: In [DZT13] wird die Beschleunigung von Datenbankanfragen in Data Warehouses durch die Verwendung von FPGAs untersucht. Im Speziellen werden Selektions- und Aggregatanfragen in der Stromdatenverarbeitung betrachtet. Die Kombination aus FPGAs zur Vorverarbeitung und Data-Warehouse-Technologien bildet somit ein Hardware/Software co-Design. Im Vergleich zur reinen Data-Warehouse-Lösung konnte durch die Verwendung von FPGAs zur Vorberechnung im TPC-DS-Benchmark (Anfrage 52) eine Beschleunigung um den Faktor 4,8 erreicht werden.

In diesem Ansatz entscheidet das DBMS darüber, welche Teile einer Anfrage auf den FPGA ausgelagert werden können. Der FPGA verfügt dabei über eine beschränkte Anzahl an Operatoren (siehe auch Abbildung 5.3 und [OYM+13]), die zur Laufzeit dynamisch konfiguriert werden können. Somit sind auch komplexere Operationen, wie z. B. mehrere Und-verknüpfte Bereichsanfragen, realisierbar.

Zu den verfügbaren Operatoren in diesem Ansatz gehören Selektionen mit Arithmetik ($+$, $-$, $*$) und den Vergleichsoperatoren ($<$, $>$, $=$, $\neq$), bitweise logische Verknüpfungen (AND, OR, NOT, XOR, $NAND$, NOR) sowie Aggregationen ($Summe$, $Anzahl$, $Minimum$ und $Maximum$) inklusive Gruppierungen. Die Operatoren lassen sich zudem beliebig miteinander kombinieren.

Dieser Beitrag zeigt, dass sich auch komplexere Operatoren auf Hardwareebene umsetzen lassen. Im Gegensatz zum eigenen Ansatz (siehe 6.1) wird hier nur ein lokales, zweistufiges Konzept (bestehend aus FPGA und Datenbankserver mit gemeinsam genutztem Arbeitsspeicher) zur Verteilung von Anfragen vorgestellt. Während alle notwendigen Vergleichsoperatoren unterstützt werden, beschränkt sich der in [DZT13] vorgestellte Ansatz lediglich auf einfache Aggregatfunktionen, wie Summen- und Maximumsberechnungen.

5.4 Transformation von Anfragen mit Abhängigkeiten

In diesem Abschnitt werden insbesondere Techniken vorgestellt, die auf dem CHASE und verwandten Techniken beruhen. Das Chase&Backchase-Verfahren [DPT06] kann zur Bestimmung äquivalenter Anfragen unter Bewahrung einer Menge von Integritätsbedingungen C verwendet werden. C kann dabei u. a. Tupel-generierende Abhängigkeiten (engl.: *tuple-generating dependencies*, TGDs) und Gleichheit erzeugende Abhängigkeiten (engl.: *equality-generating dependencies*, EGDs) beinhalten – sofern diese azyklisch sind. Während der Chase-Phase wird ein universeller Anfrageplan erzeugt, welcher alle Alternativen zur Beantwortung der Originalanfrage enthält. Anschließend reduziert die Backchase-Phase den Anfrageplan auf eine minimale Anfrage, die weiterhin äquivalent zur Originalanfrage ist.

Query Rewriting Using Views in the Presence of Functional and Inclusion Dependencies: Gryz stellt in [Gry99] Verfahren zur Transformation von konjunktiven Anfragen unter Berücksichtigung von funktionalen und Inklusionsabhängigkeiten vor. Diese auf den Basisrelationen formulierten Integritätsbedingungen werden genutzt, um weitere Anfragetransformationen zu finden, die mit den bisherigen Techniken nicht erkennbar waren. Für beide

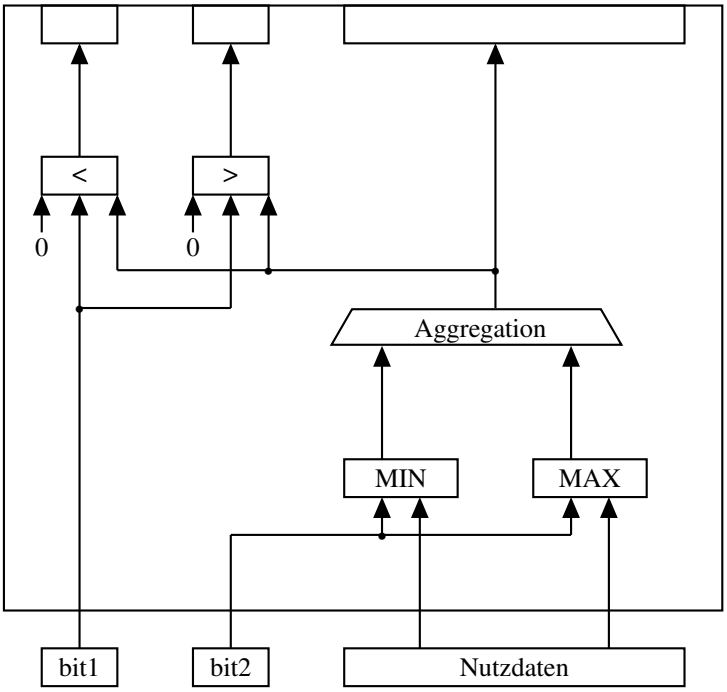


Abbildung 5.3: Beispielhafter Aufbau für ein Field Programmable Gateway Array in Anlehnung an [OYM+13]. In dieser Konfiguration unterstützt der FPGA die beiden Aggregatfunktionen *Minimum* und *Maximum* sowie die Vergleichsoperationen *kleiner* und *größer*, welche den aggregierten Wert gegen die Konstante 0 prüfen. Die Auswahl der Aggregatfunktion und des Vergleichsoperators wird über das Setzen der Bits *bit1* bzw. *bit2* gesteuert.

Arten von Abhängigkeiten werden, ähnlich zu [Qia96], zunächst Regeln zum Abbilden der Anfrage auf die Sichten gebildet. Die Integration der Abhängigkeiten erfolgt anschließend mittels des Chase, wie es auch in [DLN05] für weitere Arten von Integritätsbedingungen vorgeschlagen wird.

Funktionale Abhängigkeiten werden durch ein neues Prädikat $e(v_1, v_2)$ dargestellt. Dieses stellt sicher, dass die Werte für v_1 und v_2 unter den gegebenen funktionalen Abhängigkeiten (engl.: *functional dependencies*, FDs) gleich sind. Dadurch wird die Ergebnismenge entsprechend der FDs eingeschränkt.

Werden Anfragen unter Berücksichtigung von Inklusionsabhängigkeiten (engl.: *inclusion dependencies*, INDs) transformiert, unterscheidet Gryz zwei Fälle: Einerseits existieren Inklusionsabhängigkeiten, welche ein äquivalentes Ergebnis liefern, andererseits auch solche, die nur eine Teilmenge der Ergebnisse zurückgeben. Für beide Fälle werden zunächst in der Anfrage die Prädikate entsprechend der Inklusionsabhängigkeiten ausgetauscht. Danach erfolgt eine Minimierung der entstandenen Anfrage, um überflüssige Prädikate zu entfernen.

Abschließend wird von Gryz untersucht, in wieweit der entwickelte Ansatz für Anfragen mit sowohl funktionalen als auch Inklusionsabhängigkeiten genutzt werden kann. Dafür werden die oben eingeführten Algorithmen abwechselnd auf die Anfrage angewendet, bis keine der spezifizierten Abhängigkeiten mehr auf die Anfrage angewendet werden können. Unabhängig davon, ob zunächst die INDs oder FDs angewendet werden, kann es laut Gryz passieren, dass der Algorithmus nicht terminiert bzw. nicht alle möglichen Transformationen abdeckt, da neue FDs und INDs entstehen können.

Query containment for data integration systems: In [MHF03] wird der Begriff des *relativen Enthaltenseins* (engl.: *relative containment*) eingeführt. *Relativ* bezieht sich dabei auf die Query-Containment-Eigenschaft in Hinblick auf die verfügbaren Datenquellen aus Sicht eines Systems zur Datenintegration unter dem local-as-view-Ansatz [LRO96a]. Zusätzlich zu der Abbildung der Anfrage von den Basisrelationen auf die von den Quelldaten-

banken bereitgestellten Sichten werden auch Zugriffsbeschränkungen in Form von zusätzlichen Integritätsbedingungen, den sogenannten *binding patterns*, beachtet. In der Evaluation ihres Ansatzes zeigen die Autoren, dass das Containment auch für Anfragen und Sichten mit Vergleichsprädikaten und Rekursion entscheidbar ist.

Rewriting Queries Using Views with Access Patterns Under Integrity Constraints: Deutsch et al. untersuchen in [DLN05, DLN07] das Problem der Anfragetransformation unter Berücksichtigung von Zugriffsbeschränkungen, Integritätsbedingungen sowie den booleschen Operatoren Disjunktion und Negation. Die in dem Artikel vorgestellten Algorithmen behandeln die Fälle des Enthaltenseins (in beide Richtungen) und der Äquivalenz zweier Anfragen. Ausgehend von dem Standard AQuV-Anwendungsfall reduzieren sie diesen auf die Transformation von Anfragen mit (i) Zugriffsbeschränkungen und weiteren Integritätsbedingungen ohne Nutzung von Sichten und (ii) Integritätsbedingungen auf Sichten ohne Zugriffsbeschränkungen. Die Zugriffsbeschränkungen werden zunächst in Integritätsbedingungen umgeformt. Sämtliche Integritätsbedingungen sowie die Sichten werden anschließend mittels des Chase&Backchase-Verfahrens in die Anfrage integriert.

Durch die Zugriffsbeschränkungen werden nicht erfüllbare Teilziele aus einer Anfrage entfernt. Dadurch liefert die modifizierte Anfrage mehr Tupel zurück. Im Gegensatz zum eigenen entwickelten Ansatz (siehe Abschnitt 6.1) erfolgt keine Ersetzung der Teilziele durch alternative Teilziele, die evtl. eine Obermenge der erwarteten Ergebnismenge erzeugen.

Ähnliche Verfahren sind beispielsweise in [CY14] für eine Menge von schwach azyklischen TGDs zu finden. Diese werden ebenfalls mittels CHASE in eine gegebene Anfrage und einer Menge von konjunktiven Sichten integriert. Als speziellen Anwendungsfall betrachten die Autoren den Datenzugriff in DBMS, insbesondere wenn der Zugriff auf die Daten durch Zugriffskontrollrichtlinien in Form von TGDs eingeschränkt wird.

Query Rewriting and Optimization for Ontological Databases: Eine mögliche Umsetzung der Anfragetransformation mit TGDs und dem CHASE wird im System *XRewrite* [GOP14a] beschrieben. XRewrite formt eine gegebene Anfrage Q in eine Menge einfacherer Anfragen $Q_1, \ldots, Q_m$ um, die getrennt voneinander transformiert werden können. Durch die Anwendung von Q auf die transformierten Teilanfragen Q'_i und der abschließenden Vereinigung wird die transformierte Anfrage Q' von Q erzeugt:

$$Q' := \bigcup_{1 \leq i \leq m} Q(Q'_i). \tag{5.11}$$

Die einzelnen Teilanfragen Q_i können dabei parallel transformiert werden. Als Vorteile des Verfahrens führen die Autoren die kompaktere Form der transformierten Anfrage, die geringe Anzahl an Anfrageprädikaten pro Teilanfrage sowie die Reduzierung der notwendigen Verbundoperationen an. Der Nachteil dieses Verfahrens ist, dass die Terminierung für beliebige TGDs nicht sichergestellt ist. Terminiert die Transformation für eine Teilanfrage Q_i nicht, so fließt dessen Ergebnis nach einer festgelegten Zeitüberschreitung nicht in die Vereinigung mit ein.

Axiomatic Foundations and Algorithms for Deciding Semantic Equivalences of SQL Queries: In [CCS18] wird ein Verfahren und dessen Implementierung beschrieben, mit dem die Äquivalenz von in SQL formulierten SPJU-Anfragen mit zusätzlichen Integritätsbedingungen formal bewiesen werden kann. Als formale Grundlage führen die Autoren den U-Semiring ein, eine algebraische Struktur, welche den normalen Semiring um zusätzliche Konstrukte erweitert. Dazu gehören z. B. Projektionen, Duplikateliminierungen und Quantoren. Zusätzlich kann das Verfahren weitere Integritätsbedingungen, wie Primär- und Fremdschlüssel, Sichten und Indexe, bei der Überprüfung der Anfrageäquivalenz berücksichtigen.

5.5 Transformation weiterer Klassen von Anfragen

Ein guter Überblick zu weiterführenden *Answering-Queries-using-Views*-Techniken ist im gleichnamigen Lehrbuch [AC17] von Foto Afrati und Rada Chirkova zu finden. Darin werden etablierte und neue Konzepte und Algorithmen zur Transformation einer Anfrage zu einer äquivalenten oder *maximally-contained*-Anfrage vorgestellt.

Die dort vorgestellten Konzepte reichen von einfachen Algorithmen für konjunktive Anfragen unter Bag- und Set-Semantik mit Arithmetik und Negation bis hin zu Anfragen mit Vereinigungen und Aggregationen. Zudem wird die Anfragetransformation unter Bewahrung von Abhängigkeiten, wie *tuple generating dependencies* und *equality generating dependencies*, betrachtet.

Im Folgenden werden weitere Techniken und Ansätze vorgestellt, welche unterschiedliche Klassen von Anfragen abdecken. Dabei liegt der primäre Fokus auf der Kombination von Stromdaten mit Aggregatfunktionen.

IBMs Big SQL: Die Firma IBM hat in ihrem Produkt *Big SQL*[6] für die Anfrageoptimierung eine Engine mit mehr als 140 verschiedenen Regeln zur Transformation von Anfragen integriert [GOP+14b]. Zu diesen Regeln gehören u. a.:

- die Auflösung von Unteranfragen zu Verbund-Operationen,
- die Ersetzung von korrelierendem Unteranfragen durch abgeleitete Tabellen und
- die Zerlegung komplexer Aggregate.

Für die Realisierung der letzten Klasse von Anfragen werden zusammengesetzte Aggregate, wie der Durchschnitt, in ihre Bestandteile (hier: Summe und Anzahl) zerlegt, sofern einer dieser Bestandteile für mindestens ein weiteres Aggregat in der gleichen Anfrage benötigt wird. Dieses Prinzip wird in Big SQL als *shared aggregation* bezeichnet:

Beispiel: Anfragetransformation mit *shared aggregation* in IBMs Big SQL

Die folgende Anfrage berechnet sowohl die Gesamtlänge als auch die durchschnittliche Länge der Musikstücke pro Genre:

```
SELECT name, SUM(length) AS total, AVG(length) AS average
FROM tracks NATURAL JOIN genres
GROUP BY name
```

Da der Durchschnitt als zusammengesetzte Aggregatfunktion die Summe beinhaltet, kann diese zusammen mit der Anzahl der Musikstücke für beide Aggregate in einer Unteranfrage vorberechnetet werden[a]:

```
SELECT name total, total/cnt AS average
FROM (
  SELECT SUM(length) AS total, COUNT(length) AS cnt
  FROM tracks NATURAL JOIN genres
  GROUP BY name
) AS tmp_relation
```

[a] In Edge-basierten Systemen ist diese Zerlegung aus Sicht des Datenschutzes nicht immer unproblematisch. In Anhang D.1 wird gezeigt, wie die berechneten Zwischenergebnisse zweckentfremdet genutzt werden können.

Zudem werden in Big SQL kostenbasierte Optimierungstechniken angewendet, darunter Strategien zur Vorberechnung aggregierter Werte, wie es bereits in TinyDB umgesetzt wurde (siehe Anhang A.5). Weiterhin werden in Big SQL Entscheidungen zur Lokalisierung des Operators getroffen, beispielsweise ob ein Push-down von Operatoren bzw. Vergleichsprädikaten für eine föderierte Anfrageverarbeitung oder eine Multi-Core-Parallelisierung erfolgt.

HoTTSQL: Proving Query Rewrites with Univalent SQL Semantics: Das System *HoTTSQL* [CWCS16] bietet einen Validierungsmechanismus zur Überprüfung von Transformationsregeln auf SQL-Anfragen. Im Gegensatz

[6] Db2 Big SQL: `https://www.ibm.com/products/db2-big-sql`, zuletzt aufgerufen am 05.01.2022.

zu den meisten Ansätzen kann mittels HoTTSQL ein breiteres Spektrum an Anfragekonstrukten abgedeckt werden. Dies deckt u. a. Multimengen, korrelierende Unteranfragen und Aggregationen auf konjunktiven Anfragen ab. HoTTSQL unterstützt neben den standardmäßigen algebraischen Optimierungen, wie der Vertauschung von Selektionen mit anderen Operatoren der Relationenalgebra, weiterführende Optimierungsregeln für Aggregationen, Gruppierungen und dem Ausnutzen von Indexen.

Data Stream Sharing: In [KSK05] wird ein Verfahren zur Optimierung von Anfragen auf Stromdaten in Peer-to-Peer-Systemen (P2P) vorgestellt. Dieses Verfahren beruht auf zwei Prinzipien: Einerseits wird die Anfrageverarbeitung derart abgeändert, dass sie für die Verteilung und Ausführung von neuen, kontinuierlichen Anfragen innerhalb des P2P-Netzwerkes ausgelegt ist. Außerdem erfolgt eine Optimierung zu einer *multi-subscription*, wodurch Zwischenergebnisse über mehrere Anfragen hinweg geteilt und wiederverwendet werden können.

Dazu stellen die Autoren die Anfragesprache *Windowed XQuery* vor, welche Anfragen, inkl. Fensterfunktionen, auf XML-Datenströmen ermöglicht. Auf Basis dieser Anfragesprache werden Query-Containment-Techniken angewendet. Dabei wird wie bei Mühl [Müh02] geprüft, ob auf Basis von Selektionsprädikaten, Projektionsattributen, etc. eine Teilmengenbeziehung zwischen zwei Anfragen besteht. Für die Anwendung von Aggregatfunktionen muss sichergestellt werden, dass für zwei Anfragen Q_1 und Q_2 zunächst gleiche Selektionsbedingungen vorliegen. Zudem müssen Q_1 und Q_2 mit den gleichen Aggregatfunktionen vorverarbeitet worden sein. Als weitere Bedingung müssen Q_1 und Q_2 in der abschließenden Projektion die gleichen Aggregatfunktionen enthalten.

Speziell für Stromdaten müssen weitere Bedingungen erfüllt sein: Q_1 und Q_2 sind miteinander kompatibel, wenn die Schrittweite bzw. die Fenstergröße von Q_2 ein Vielfaches der Schrittweite bzw. der Fenstergröße von Q_1 sind. Zudem muss die Fenstergröße von Q_1 ein Vielfaches der Schrittweite von Q_1 sein, da es ansonsten bei Q_2 zu unerwünschten Überlappungen kommt.

Der in [KSK05] vorgestellte Algorithmus arbeitet in vier Schritten: Zunächst werden mögliche teilbare Datenströme entdeckt und für die gegebenen Anfragen die Anfragepläne erstellt. Anschließend erfolgt ein Matching der Eigenschaften und Prädikate zwischen einzelnen Teilen der verschiedenen Anfragepläne. Zum Schluss wird ein Matching der Aggregatoperatoren ermittelt.

Das Verfahren lässt sich für verschiedene Aggregatfunktionen verwenden. Neben den klassischen Aggregatfunktionen werden zum Beispiel auch holistische Aggregate, wie Quantile, unterstützt. Für das Verfahren werden weiterhin verschiedene Erweiterungen und Optimierungen vorgestellt, wie beispielsweise das Caching von einzelnen Matching-Ergebnissen oder die Optimierung von Berechnungen in Schleifen.

Scalable Delivery of Stream Query Result: In [ZSA09] stellen Zhou et al. einen Middleware-Ansatz zur Optimierung von Anfragen an ein Stromdatensystem vor. In ihrem Ansatz untersuchen sie, welche Teile von mehreren Anfragen zusammengefasst werden können um die Gesamtanzahl an Anfragen an das System zu minimieren. Hierfür erweitern sie Query-Containment-Techniken für kontinuierliche Datenströme, indem Sichten auf diesen Strömen verschmolzen werden, die anschließend durch einfache Filtertechniken eines Publish/Subscribe-Systems weiterverarbeitet werden können. Ausgehend vom Query-Containment-Problem für relationale Datenbanken führen die Autoren eine Semantik für Fensterfunktionen ein, um das Problem auf Stromdaten zu übertragen:

> **Definition:** Query Containment für Fensterfunktionen nach [ZSA09]
> Eine kontinuierliche Datenstromanfrage Q_1 ist in einer Anfrage Q_2 enthalten, gekennzeichnet durch $Q_1 \sqsubseteq Q_2$, falls für alle Instanzen des Datenstroms $s \in S$ und für jeden Zeitpunkt $\tau \in \mathrm{T}$ gilt, dass $Q_1(s, \tau) \subseteq Q_2(s, \tau)$. Q_1 und Q_2 sind äquivalent, wenn $Q_1 \sqsubseteq Q_2$ und $Q_2 \sqsubseteq Q_1$ gilt.

Hierfür führen sie Bedingungen ein, die für einen Fenster-basierten Verbundoperator notwendig sind. Für eine Anfrage auf zwei Datenströmen s_1 und s_2 und den dazugehörigen Fensterbreiten $interval_1$ bzw. $interval_2$, einer Slide-Länge $slide$ und einer Startzeit $begin$, kann für zwei Tupel $t_1 \in s_1$ und $t_2 \in s_2$ ein Ergebnistupel t durch einen Verbund erzeugt werden, wenn folgende Bedingungen gelten:

1. Das Verbundprädikat ist erfüllt,
2. -1 $* interval_1 \leq t_1.ts - t_2.ts \leq interval_2$,
3. $t_1.ts > n_2 * slide - interval_1$, wobei $n_2 = \lceil (t_2.ts - begin)/slide \rceil$ und
4. $t_2.ts > n_1 * slide - interval_2$, wobei $n_1 = \lceil (t_1.ts - begin)/slide \rceil$.

Nach diesen Bedingungen müssen die Tupel innerhalb des betrachteten Intervalls des Fensters liegen. Außerdem muss mindestens ein Paar temporärer Relationen zu einem bestimmten Fenster existieren, welches beide Tupel enthält. Ausgehend von diesen Bedingungen formulieren die Autoren das Query-Containment-Problem für SPJ-Anfragen auf Datenströmen: Eine Select-Project-Join-Anfrage (SPJ-Anfrage) Q_1 auf einem Datenstrom ist in einer anderen SPJ-Anfrage Q_2 enthalten, wenn sie folgende Bedingungen erfüllen:

1. $Q_1^\infty \sqsubseteq Q_2^\infty$, wobei Q_i^∞ eine Anfrage ist, bei der die Größe des Fensters von Q_i auf ∞ gesetzt wurde;
2. $begin(Q_1) \geq begin(Q_2)$;
3. $\forall i : interval_i(Q_1) \leq interval_i(Q_2)$, wobei $interval_i(Q_j)$ die Größe des Fensters des i-ten Streams von Q_j ist;
4. Außerdem muss eine der beiden folgenden Bedingungen gelten:
   (a) $\forall i : 1 \leq slide(Q_2) \leq max(1, interval_i(Q_2) - interval_i(Q_1))$;
   (b) oder
      i. $\exists m \in [1, \infty)$,
      sodass $begin(Q_2) + m * slide(Q_2) - begin(Q_1) \leq min_i(interval_i(Q_2) - interval_1(Q_1))$;
      ii. $slide(Q_1) = k * slide(Q_2), k \in \mathbb{N}^+$;

Umgangssprachlich formuliert enthält die Q_2 die Anfrage Q_1, wenn (1) die Basisdaten von Q_1 in Q_2 enthalten sind, (2) Q_2 vor Q_1 beginnt und (3) die Intervalle von Q_2 mindestens genauso groß sind wie die Intervalle von Q_1. Zudem muss entweder (4.a) die Sliding-Größe von Q_2 kleiner sein als der Abstand zwischen den Intervallen oder (4.b) die beiden Fenster müssen sich synchron bewegen.

Abschließend stellen die Autoren einen Algorithmus zur Vereinigung zweiter SPJ-Anfragen auf Datenströmen vor. Ziel der Vereinigung ist eine verringerte Anfragelast auf den Stromdaten durch die Vermeidung von zusätzlichen Berechnungskosten. Die Vereinigung der Anfragen enthält die Attribute beider Anfragen und Selektionsbedingungen, die durch das logische Oder verknüpft werden. Entsprechend der zusätzlichen Containment-Eigenschaften wird der Beginn des Fensters und dessen Intervall sowie die Größe der Slides auf das Minimum (`begin` und `slide`) bzw. Maximum (`interval`) gesetzt.

Das Verfahren stellt eine Ergänzung zu den bisher vorgestellten Query-Containment-Verfahren dar, da mit dem von [ZSA09] entworfenen Ansatz Anfragetransformationen auf Stromdaten ermöglicht werden. Für das im Rahmen dieser Arbeit entworfene Konzept zur Transformation von Anfragen (siehe Kapitel 6) wurde der Ansatz jedoch nicht mit aufgenommen, da in den konkreten Anwendungsszenarien (siehe Abschnitt 2.1) die zu Grunde liegenden Analysen zwar auf Stromdaten, jedoch nicht mit fenster-basierten Ansätzen durchgeführt wurden.

Eine Umsetzung dieses Verfahrens wird in [ABR21] skizziert. Die Autoren zeigen, wie die `SELECT`-, `WHERE`- und `WINDOW`-Klausel von SQL genutzt werden können, um nutzerdefinierte Datenschutzrichtlinien auf smarten Geräten zu realisieren.

QTor: A Flexible Publish/Subscribe Peer-to-Peer Organization Based on Query Rewriting: Dufromentel et al. stellen in [DCLL15] ein weiteres Verfahren zur Anfragetransformation in Publish/Subscribe-Systemen für stromdatenbasierte Anwendungen vor. Das entwickelte Verfahren, *QTor*, organisiert die Anfragen der Teilnehmer in sogenannten *Communities*, welche durch Anfragen auf die gleichen Teilmengen eines Datenbestandes zugreifen. Die Communities stellen somit eine Menge von Sichten dar, auf welche die Anfragen anstatt der Basisrelationen zugreifen. Zudem müssen die Communities folgende Eigenschaften berücksichtigen:

1. Konsistenz mit den Rechenkapazitäten der Teilnehmer,
2. Verträglichkeit mit den Richtlinien zum Ressourcen-Management,
3. Effizienz hinsichtlich Rechenkapazität, Netzwerkauslastung sowie Energiebedarf und
4. Bereitstellung der Ergebnisse mit geringer Latenz.

Basierend auf diesen Eigenschaften wird eine Anfragetransformation der Originalanfrage vorgenommen, sodass diese auf den Communities effizient ausgeführt werden kann. Die Organisation der Communities erfolgt dabei adaptiv auf Basis der aktuellen Netzwerktopologie und kann sowohl in zentralen als auch dezentralen Netzwerken verwendet werden.

Der eigene Ansatz verfolgt neben der Wahrung des Datenschutzes ähnliche Ziele zur Sicherstellung einer geringen Latenz und einer hohen Effizienz. Während in diesem Ansatz die Wiederverwertung von Zwischenergebnissen im Vordergrund steht, konzentriert sich das in Kapitel 6 vorgestellte Konzept auf die Transformation einzelner Anfragen. Beide Ansätze können prinzipiell miteinander gekoppelt werden; dies steht aber nicht im Fokus dieser Arbeit.

Query Containment für verschiedene Klassen von Anfragen: In [WL03, ALM04] und [KMRS14] werden zusätzlich negierte Prädikate in den Selektionsbedingungen eingeführt (Klasse $SPJ^{\neg}$). Klug [Klu88] erweitert die Anfrageprädikate um zusätzliche arithmetische Vergleichsoperatoren ($<$, $\leq$, $\geq$, $>$).

Chaudhuri und Vardi [CV93] übertragen die Forschungsergebnisse aus dem Bereich Query Containment von der Mengen- auf die Multimengensemantik. Die Komplexität für allgemeine Anfragen unter Multimengensemantik ist gegenwärtig noch ein offenes Problem [Coh06]. Für einen aktuellen Überblick zu den Query Containment-Ansätzen unter Mengen- und Multimengensemantik sei auf Kolaitis in [Kol13] bzw. auf Afrati [ADG14] für die Klasse der Pfadanfragen verwiesen.

Neben den verschiedenen Semantiken sind auch Aggregatfunktionen, Gruppierungen und Selektionen auf diesen Gruppen für komplexe Anfragen von Interesse. Klug zeigt in [Klu82] die Integration von einfachen Aggregaten (Summe, Anzahl, Minimum und Maximum) in bestehende Ansätze, indem eine Erweiterung der relationalen Algebra erfolgt. In [CNS07, CNS99] wird dieser Ansatz zudem auf die Multimengensemantik übertragen. Zusätzlich werden weitere Integritätsbedingungen, wie z. B. funktionale Abhängigkeiten, abgedeckt. Zudem werden die Summe, das Maximum, das Minimum und die Anzahl als Vergleichsprädikate erlaubt. Grumbach [GRT04] führt mit dem Durchschnitt und der Prozentrechnung zusätzliche Aggregatfunktionen ein.

In [BPV15] wird eine Anfragesprache für Anfragen höherer Ordnung, inklusive Rekursion und Komplexitätsbetrachtungen, vorgestellt. Eine gute Übersicht zu den Komplexitäten des Query-Containment-Problems für einzelne Anfrageklassen, inkl. funktionaler Abhängigkeiten und TGDs, ist in dem Artikel von Benedikt [BBtCP15] zu finden.

5.6 Anfragetransformation in ressourcenbeschränkten Umgebungen

Um für das später vorgestellte Konzept (siehe Kapitel 6) die Anfragekapazitäten der verschiedenen Knoten vollständig auszunutzen, müssen die bestehenden Techniken für das *Anwering-Queries-using-Views*-Problem (AQuV) auf die neue Problemstellung adaptiert werden: Anstatt einer Menge von Sichten werden für den in Kapitel 6 eingeführten Schichten-Ansatz abstrakte Anfragekapazitäten benötigt. Diese beschreiben, welche Operatoren auf den einzelnen Knoten zur Beantwortung der Anfrage zur Verfügung stehen. Im Folgenden werden verschiedene Ansätze zur Integration dieser Kapazitäten vorgestellt.

Querying Heterogeneous Information Sources Using Source Descriptions: In [LRO96b] wird mit dem *Information Manifold*-System ein Verfahren beschrieben, welches die Kapazitäten von Quellen durch annotierte Sichtdefinitionen beschreibt. Als Quellenbeschreibung dienen sogenannte *capability records*, welche die Sichtdefinitionen und die Kapazitäten für jede Quelle enthalten.

Die Kapazitäten werden als 5-Tupel $(S_{in}, S_{out}, S_{Sel}, min, max)$ angegeben. S_{in} beinhaltet die Eingabeattribute der Anfrage auf die jeweilige Quelle (im klassischen Sinne: die Attribute im Rumpf der Sichtdefinition), S_{out} hingegen die Ausgabeparameter (im klassischen Sinne: die Attribute im Kopf der Sichtdefinition). Für die Selektionsbedingungen S_{sel} gilt, dass die dort verwendeten Attribute eine Teilmenge der Ein- und Ausgabeparameter sein müssen ($S_{sel} \subseteq S_{in} \cup s_{out}$). Als mögliche Vergleichsoperatoren können $<, \leq, =$ und $\neq$ verwendet werden. min und max geben die mögliche minimale bzw. maximale Anzahl von belegbaren Eingabeparametern an.

Ausgehend von den *capability records* und einer gegebenen Anfrage wird eine Modifikation des Bucket-Algorithmus (siehe [LRO96a] und Abschnitt 5.1) zur Ermittlung der passendsten Quelle angewendet. Die Modifizierung bezieht die Anfragekapazitäten mit ein, wodurch nur relevante Buckets pro Teilziel zur Beantwortung der Anfrage in Betracht gezogen werden.

Capability-Sensitive Query Processing on Internet Sources: In [GLY99, GMLY99] wird ein Algorithmus zur effizienten Optimierung von konjunktiven als auch disjunktiven Anfrageplänen unter Einbeziehung der zur Verfügung stehenden Anfragekapazitäten eingeführt. Der Ansatz garantiert, dass kapazitätsbeschränkte Datenquellen Anfragen ausführen können, indem eine Transformation auf die zur Verfügung stehenden Anfragesprache erfolgt.

Eine mögliche Kapazitätsbeschränkung ist, dass die Anzahl der vorkommenden Bedingungen in den Anfragen begrenzt wird. Außerdem kann die Struktur der Anfrage auf atomare oder konjunktiv verknüpfte Bedingungen sowie Anfragen mit einer bestimmten Template-Struktur beschränkt werden. Zudem können einige Anfragen nur dann umgesetzt werden, wenn die Attributwerte für bestimmte Attribute bzw. Attributkombinationen als Eingabeparameter angegeben wurden.

Die Autoren stellen in [GLY99] zunächst eine neu entwickelte Beschreibungssprache, die *Simple Source-Description Language* (SSDL) für die Anfragekapazitäten vor. Zur Formulierung der Kapazitäten wird die gegebene Anfrage zunächst wahlweise in die disjunktive oder konjunktive Normalform gebracht. Die folgende Definition zeigt den Aufbau der SSDL für Anfragen in der disjunktiven Normalform:

Definition: Aufbau der SSDL nach [GLY99]

Eine SSDL-Beschreibung für eine Datenquelle R wird durch ein Triplet (S, G, A) charakterisiert, wobei

- S eine Menge von Nichtterminalen,
- G eine Menge von Ableitungsregeln und
- A eine Menge von Abbildungen von Nichtterminalsymbolen auf Attributmengen aus R ist.

Dabei muss jedes Nichtterminalsymbol $s_i \in S$, mit Ausnahme des Startsymbols $s \rightarrow s_1 \vee \cdots \vee s_m$ auch in A als Abbildung der Form $s_i \rightarrow \{a_{i_1}, \ldots a_{i_n}\}$ vorkommen.

Beispiel: Einfache Anfrage mit Kapazitätsbeschränkungen

Die folgende SSDL-Beschreibung kennzeichnet zwei parametrisierbare Sichten, s_1 und s_2, die zur Beantwortung einer Anfrage s genutzt werden können.

1. $s \rightarrow s_1 \vee s_2$
2. $s_1 \rightarrow artist = \$a \wedge bitrate < \$b$
3. $s_2 \rightarrow artist = \$a \wedge genre = \$g$
4. $attributes :: s_1 : \{artist, year, length, genre\}$
5. $attributes :: s_2 : \{artist, year, length\}$

Die Zeichen $\$a$, $\$b$ und $\$g$ stehen dabei für Konstanten, welche als Werte für die Attribute $artist$, $bitrate$ bzw. $genre$ eingesetzt werden können.

Aus dieser Beschreibung für die jeweiligen Quelldatenbanken werden anschließend umsetzbare Anfragepläne bestimmt. Die durch den entwickelten Algorithmus, *GenCompact*, generierten Anfragepläne für kapazitätsbeschränkte Quelldatenbanken bieten diverse Vorteile: Zum einen werden die generierten Anfragen vollständig von der Quelldatenbank unterstützt und dort effizient mit den bereitstehenden Kapazitäten ausgeführt. Zudem wird durch GenCompact ein größerer Suchraum für die Anfragepläne aufgespannt, wodurch potentiell effizientere Pläne gefunden werden können.

Dafür werden die Anfrageprädikate so angeordnet, dass die ausführbaren Operatoren als erstes ausgeführt werden. Der Algorithmus zur Anfragetransformation bestimmt dabei Teilziele der Anfrage, welche auf den Quelldatenbanken ausgeführt werden können. Als realisierbare Teilziele werden diejenigen parametrisierbaren Sichten betrachtet, welche die entsprechenden Anfrageprädikate unterstützen und hinsichtlich der Projektionsliste alle Attribute enthalten, welche für die Ausführung der verbleibenden Anfrage benötigt werden.

Zur Optimierung des Verfahrens wird ein Kostenmodell im Zusammenspiel mit Strategien zur Einschränkung des Suchraumes eingesetzt. Dabei werden lokale Teilpläne verworfen, wenn deren Kosten bereits als nicht-optimal bekannt sind oder Pläne existieren, die einen größeren Teil der Anfrage abdecken. Abschließend erzeugt der Plangenerator für jede Bedingung einen einzelnen Plan und verarbeitet sie separat für jede $\vee$- und $\wedge$-Verknüpfung.

Capabilities-Based Query Rewriting in Mediator Systems: Ein ähnlicher Ansatz wird in [PGH98] mit dem *Capabilities-based Query Rewriting* verfolgt. Ausgehend von einer Reihe möglicher Operationen und Anfragen, die auf einem gegebenen Layer L ausgeführt werden sollen, bestimmt eine kapazitätsbasierte Anfragetransformation partielle SPJ-Anfragen, die auf L ausgeführt werden können. Zur Darstellung der Kapazitäten schlagen die Autoren eine neue Sprache vor, die *Relational Query Description Language* (RQDL).

Die RQDL beschreibt die Kapazitäten bzw. unterstützen Operatoren der einzelnen Knoten durch Anfrage-Templates, welche parametrisierbare Sichten darstellen. Diese Templates bilden die unterstützten Anfrageprädikate und die verwendbaren Basisrelationen ($Table$) ab.

Beispiel: Einfache Anfrage als RQDL

Die Sicht

```
1  SELECT *
2  FROM tracks
3  WHERE length > 42000
```

entspricht in RQDL den folgenden Prädikaten:

$$Table(tracks), subset(R, tracks), length > 42000 \tag{5.12}$$

Sofern die Datenquelle eine Anfrage Q nicht direkt durch die Kapazitäten des Wrappers unterstützt, überprüft der *Capabilities-Based Rewriter* (CBR), ob durch die verfügbaren Operatoren Q äquivalent dargestellt werden kann. Der CBR-Algorithmus durchläuft dazu drei aufeinanderfolgende Phasen, um alle algebraisch optimalen Anfragepläne zu erzeugen:

1. *SubQuery Discovery*: In dieser Phase wird nach Anfragen gesucht, welche mindestens ein Teilziel von Q unterstützen und der Anfragekapazität des Wrappers entsprechen. Dabei werden nur diejenige Anfragen ausgegeben, welche die meisten Selektions- und Verbundbedingungen bzgl. Q unterstützen. Dies reduziert den aufgespannten Suchraum. Die Anfragen werden dafür beginnend von den „kleinsten" Anfrage-Templates her aufgebaut. Kommen diese in rekursiv aufgerufenen Templates vor, so terminiert die Discovery-Phase spätestens nach Überschreiten der Anzahl der Teilziele aus Q.
2. *Plan Construction*: Die ermittelten Teilanfragen werden in dieser Phase kombiniert, sodass Q berechnet wird. Dazu müssen alle Selektions- und Verbundbedingungen sowie alle in Q vorkommenden Konstanten enthalten sein. Dabei werden die Containment-Eigenschaften zwischen den Anfragen beachtet, damit keine unnötigen Teilziele erzeugt werden.
3. *Plan Refinement*: Ähnlich zu den Techniken der Anfrageoptimierung (siehe Abschnitt 5.8) werden zusätzliche Projektionen in den entstandenen Anfrageplan hinzugefügt.

Ausgehend von der RQDL-Beschreibung erzeugt der CBR-Algorithmus Teilanfragen (ComponentSubQuerys; CSQs), welche auf der aktuellen Ebene realisierbar sind. Zudem wird eine Anfrage erzeugt, welche die Teilresultate im Wrapper zusammenführt.

Beispiel: CSQ für das vorherige Beispiel

Folgende Anfrage lässt sich somit nicht vollständig umsetzen:

```
1 SELECT *
2 FROM tracks
3 WHERE length > 42000
4 AND title LIKE '%Cosmo%',
```

Auf Basis der im letzten Beispiel formulierten Prädikaten kann nur ein einziger CSQ ermittelt werden, wenn das Anfrage-Template nur einfache Vergleiche auf Integer-Werten zulässt. Zwar kann eine partielle Anfrage entwickelt werden, jedoch wird das Prädikat `substring(title, 'Cosmo')` nicht lokal unterstützt. Eine Grammatik, die `substring`-Vergleiche zulässt, würde ebenfalls eine entsprechende Teilanfrage entdecken bzw. eine, die gar beide Anfrageprädikate unterstützt.

Aus der Menge der algebraisch optimalen Pläne wird anschließend über eine kostenbasierte Optimierung der Plan mit dem geringsten Aufwand ermittelt. Dieser wird daraufhin durch den Wrapper ausgeführt.

Expressive capabilities description languages and query rewriting algorithms: In [VP00] wird eine Erweiterung bzw. Kombination der RQDL um weitere Sprachkonstrukte aus Datalog beschrieben. Der von Vassalos und Papakonstantinou entwickelte Sprachvorschlag *parameterized Datalog*, kurz p-Datalog, ist einerseits einfacher zu formulieren, da er an die Datalog-Notation angelehnt ist. Zudem werden weitere Sprachkonstrukte, wie Disjunktionen und Negationen, unterstützt. Beliebige rekursive Anfragen werden hingegen nicht unterstützt. Für p-Datalog wird zudem ein Capability-based Rewriter vorgestellt, welcher die Anfragetransformation übernimmt. Abschließend wird, basierend auf einer erweiterten Fassung von RQDL um Vektorvariablen, eine Reduktion von RQDL auf p-Datalog vorgestellt. Durch die Vektorvariablen können eine beliebige Anzahl an Attributen dargestellt werden, wodurch beliebige relationale Strukturen in RQDL bzw. p-Datalog abgebildet werden können.

Answering Queries Using Limited External Query Processors: In [LRU99] wird untersucht, inwieweit die durch Anfragetemplates aufgespannte, unendliche Menge an parametrisierbaren Sichten für die Anfragetransformationen effizient durchsucht werden kann. Eine parametrisierte Sicht in V steht stellvertretend für eine Menge von Sichten gleicher Struktur, bei denen einzelne Konstanten und Variablen durch Platzhalter ersetzt wurden. Die Autoren kommen zu dem Ergebnis, dass die Sichten in eine endliche Menge an Äquivalenzklassen unterteilt werden können. Dadurch muss nur jeweils eine Sicht stellvertretend für jede Äquivalenzklasse betrachtet werden. Der von den Autoren entwickelte Algorithmus arbeitet auf einfachen konjunktiven Anfragen und Sichten, welche aus Prädikaten mit den Vergleichsoperatoren $<$, $\leq$, $=$ und $\neq$ zusammengesetzt sind.

Zur Überprüfung, ob eine konjunktive Anfrage Q' eine gültige Transformation von Q, welche lediglich die Sichten aus V nutzt, ist, muss eine Erweiterung Q'' von Q' gebildet werden, die äquivalent zu Q ist. Dazu müssen entsprechende Homomorphismen gefunden werden. Können durch diese Homomorphismen zwei Sichten V_1 und V_2 beliebig ausgetauscht werden, so gelten diese als äquivalent.

5.7 Vorverarbeitung von Informationen

In diesem Abschnitt werden Verfahren zur Vorverarbeitung von Informationen auf lokalen Geräten vorgestellt. Der Schwerpunkt liegt dabei auf Sensornetzwerken und vorberechneten Aggregationen.

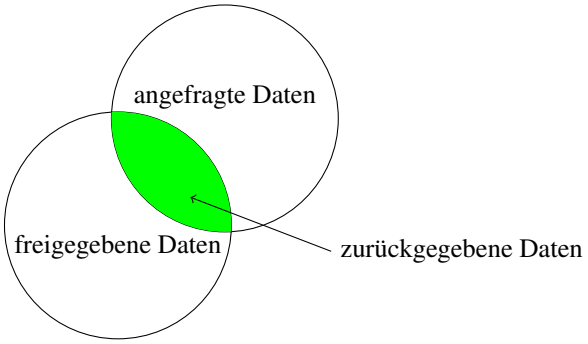


Abbildung 5.4: Zurückgegebene Daten bei der Verwendung von Zugriffsrechten (Abbildung nach [SW74])

Access control in a relational data base management system by query modification: In [SW74] wird das Zugriffskontrollsystem für das relationale Datenbankmanagementsystem INGRES (INteractive Graphics and REtrieval System) – eines der ersten RDBMS – vorgestellt. Die Grundlagen dieses Zugriffskontrollsystems wurden später von vielen weiteren Forschungs- und kommerziellen Systemen übernommen.

Der Grundgedanke hinter dem Mechanismus zur Zugriffskontrolle liegt darin, eine Benutzeranfrage so zu modifizieren, dass Verletzungen hinsichtlich der zuvor festgelegten Zugriffsrechte ausgeschlossen werden. Für eine gegebene Relation $r(R)$, eine Anfrage Q und eine Menge von Zugriffsrechten S wird für jedes Tupel $t \in r(R)$ überprüft, ob die in Q und S verwendeten Prädikate auf t zutreffen. Dazu wird die Anfrage Q mit den Prädikaten aus S `UND`-verknüpft. Dadurch werden nur Tupel herausgegeben, die sowohl durch die Anfrage, als auch durch die vergebenen Zugriffsrechte abgedeckt werden (siehe Abbildung 5.4).

Für aggregierte Werte wird angenommen, dass diese unbedenklich sind, da nur statistische Werte herausgegeben werden. Dennoch unterscheiden Stonebreaker et al. [SW74] folgende vier Fälle:

1. Aggregationen sind ohne Einschränkungen erlaubt,
2. Aggregationen sind nur dann erlaubt, wenn eine Mindestanzahl an Tupeln aggregiert werden,
3. über unqualifizierte Relationen, d. h. ohne vorherige Selektion auf bestimmte Werte, sind Aggregationen erlaubt und
4. Aggregationen über Relationen, die mit den Zugriffsrechten vorselektiert wurden, sind erlaubt.

Der Zugriff auf die Daten zum Zwecke der Aggregation nimmt entsprechend der vier Fälle ab, da die Zugriffsbedingungen verschärft werden.

Privacy-Preserving Data Aggregation in Two-Tiered Wireless Sensor Networks with Mobile Nodes: Yao et al. stellen in [YLX14] ein zweistufiges Verfahren zum Aggregieren von Sensordaten in mobilen Umgebungen vor. In ihrem Beitrag konzentrieren sie sich auf die Bildung von Summen. Die Ergebnisse lassen sich aber auch auf weitere Aggregatfunktionen, wie Durchschnitt, Minimum und Maximum, übertragen. Jeder Aggregator bildet die lokalen Summen und schickt diese an eine globale Senke. Zwischen Sensor und Aggregator bzw. Aggregator und Senke werden die Daten mit zusätzlichem, aber festem Rauschen versehen. Die Werte für das Rauschen werden zu Beginn der Anfrageverarbeitung durch die Senke generiert. Dadurch wird es dem Aggregator und anderen Sensoren unmöglich, die Originalwerte eines einzelnen Sensors zu ermitteln.

A robust and privacy-preserving aggregation scheme for secure smart grid communications in digital communities: Ein ähnliches Verfahren wird in [FMLJ16] für den Anwendungsfall des Smart Grid vorgestellt. Zusätzlich wird hier eine homomorphe Verschlüsselung auf Basis von elliptischen Kurven (engl.: Elliptic Curve Cryptography; ECC) eingesetzt, um speziell gegen Man-in-the-Middle-Angriffe geschützt zu sein. In dem dreistufigen

Verfahren werden die Rohdaten zusätzlich auf den lokalen Geräten voraggregiert, bevor sie an die Aggregatoren außerhalb des eigenen Netzes geschickt werden.

Privacy-Assured Aggregation Protocol for Smart Metering: A Proactive Fault-Tolerant Approach: Ebenfalls für den Anwendungsfall Smart Grid stellen Won et al. in [WMYR16] ein Protokoll zur Integration von Differential Privacy in Auswertungen mit Aggregation vor. Sie erweitern dabei bestehende Ansätze, um zu große Abweichungen vom realen Energieverbrauch bei möglichst geringen Zusatzinformationen zu vermeiden. Das notwendige Rauschen für die einzelnen Geräte und Aggregatoren wird dabei über die verteilte Berechnung einer Laplace-Verteilung erzeugt. Zusätzlich wird eine leichtgewichtige Verschlüsselungsmethode basierend auf homomorpher Verschlüsselung angewandt. Dadurch ist das Protokoll auch auf leichtgewichtigen Systemen, die beispielsweise TinyOS verwenden, lauffähig. Das Verfahren kann zudem dynamisch neue Geräte in das Smart Grid aufnehmen bzw. wieder von dort entfernen.

Distributed Service Deployment in Mobile Ad-Hoc Networks: In [AFB07] wird ein Verfahren zur Aufteilung von Anwendungen auf Basis Service-orientierter Architekturen präsentiert. Anwendungen werden dazu aus bestehenden Services zusammengesetzt. Diese können beispielsweise zum Aggregieren und Aufbereiten der Daten speziell in verteilten Umgebungen genutzt werden. Die Kommunikation der Anwendung mit den Diensten erfolgt über eine extra Schicht, welche die Verbindung zu einem Gruppenanführer herstellt. Dieser Anführer repräsentiert dabei eine Gruppe lokaler, homogener Geräte gleicher Leistungsfähigkeit. Zur Parametrisierung der Anfrage werden einerseits die Anforderungen der Anwendung (z. B. minimaler Durchsatz) als auch die Limitierungen der Hardwareeigenschaften (bzgl. Energieverbrauch etc.) in Betracht gezogen.

5.8 Äquivalente Anfragen für die Anfrageoptimierung

Die Hauptanwendung der Anfragetransformation in Datenbanksystemen ist es, dass mittels der Optimierung die Zeit zur Berechnung des Ergebnisses einer Anfrage minimiert wird. Ein Teilgebiet der Anfrageoptimierung umfasst die algebraische Optimierung. Die algebraische Optimierung basiert auf Regeln zur Termersetzung durch Äquivalenzen in der Relationenalgebra, d. h., die Anfrage wird so umgeformt, dass die umgeformte Anfrage unabhängig von dem Datenbankzustand immer das gleiche Ergebnis wie die Originalanfrage liefert. Dabei soll eine Anfrage derart transformiert werden, dass Teilanfragen möglichst wenig Daten, d. h., wenige Tupel und Attribute, zurückgeben. Dieser etablierte Ansatz wurde u. a. im Volcano- [GM93] und Starburst-Optimierer [HFLP89] sowie – als neueres Framework – in Apache Calcite[7] verwendet.

Basisoperatoren der Relationenalgebra, wie Selektion und Projektion, sollen möglichst früh auf den Basisrelationen ausgeführt werden, bevor komplexere Operationen, wie der Verbund, größere Zwischenergebnisse erzeugen. Dies führt dazu, dass die Seitenzugriffe auf die Festplatte minimiert werden, da dies einen Flaschenhals in der Datenverarbeitung darstellt [Tan09]. Die vier wichtigsten Teilziele der Optimierung umfassen die frühestmögliche Ausführung von Selektions- und Projektionsbedingungen, das Zusammenfassen von Basisrelation zur Eliminierung von unnötigen Zwischenergebnissen, die Entfernung von redundanten Operationen, Idempotenzen oder leeren Zwischenergebnissen und das Wiederverwenden von Zwischenergebnissen durch das Zusammenfassen gleicher Teilausdrücke. Im Folgenden werden kurz die wichtigsten Optimierungsregeln nach [SSH11] vorgestellt. Eine genauere Betrachtung, wie die Regeln für den Aspekt der Datensparsamkeit genutzt werden können, wird in Abschnitt 6.1 vorgenommen.

Kommutativität von Selektion und Verbund: Die Algebraoperatoren Selektion und Verbund kommutieren unter bestimmten Bedingungen. Diese Einschränkungen betreffen hierbei Selektionsprädikate, die entweder nur in einer der beiden zu verknüpfenden Relationen vorkommen oder bei denen die Selektionsattribute bei Attribut-Attribut-Vergleichen auf beide Relationen verteilt sind. Im allgemeinen Fall können die Selektionsprädikate zweier Relationen, r_1 und r_2, in drei Teilmengen eingeteilt werden:

[7] `https://calcite.apache.org/`, zuletzt aufgerufen am 05.01.2022

- X: Selektionsprädikate, die vollständig in r_1 anwendbar sind,
- Y: Selektionsprädikate, die vollständig in r_2 anwendbar sind und
- Z: Selektionsprädikate, die erst nach dem Verbund von r_1 und r_2 anwendbar sind

Als äquivalente Anfragetransformation ergibt sich hieraus folgende Regel:

$$\begin{aligned}\sigma_{X \wedge Y \wedge Z}(r_1 \bowtie r_2) \Leftrightarrow \sigma_Z(\sigma_X(r_1) \bowtie \sigma_Y(r_2));\\ \text{mit } attr(X) \subseteq R_1,\\ \wedge\, attr(Y) \subseteq R_2,\\ \wedge\, attr(Z) \subseteq R_1 \cup R_2,\\ \wedge\, attr(Z) \not\subseteq R_1,\\ \wedge\, attr(Z) \not\subseteq R_2.\end{aligned} \tag{5.13}$$

Der Verbund und die Selektion kommutieren, sofern die Attribute der Selektionsprädikate dies erlauben. Ein Selektionsprädikat kann nur auf die Relationen hinuntergezogen werden, die alle Attribute des Selektionsprädikates vollständig enthalten. Insbesondere wenn Attribut-Attribut-Vergleiche auftreten, bei denen die Attribute Teil von unterschiedlichen Relationen sind, kann die Selektion nicht mit dem Verbund vertauscht werden.

Für eine spätere algebraische Optimierung zur Umsetzung der Datensparsamkeit kann diese Regel verwendet werden, damit die Selektionen möglichst nahe an die Rohdaten bzw. die IoT-Endknoten verlagert werden. Insbesondere wenn Daten von mehreren Sensoren miteinander via Verbundoperationen auf einem übergeordneten Knoten kombiniert werden, wird hierdurch verhindert, dass unnötige Tupel an diesen gelangen.

Kommutativität von Projektion und Verbund: Projektion und Verbund kommutieren, wenn in der hineingezogenen Projektion die Verbundattribute ($R_1 \cap R_2$) erhalten bleiben, da diese erst nach dem Verbund herausprojiziert werden können. Die verbleibenden inneren Projektionen mit den Attributlisten Y und Z enthalten zudem alle Attribute der ursprünglichen Projektionsliste X, die in R_1 respektive R_2 vorkommen:

$$\begin{aligned}\pi_X(r_1 \bowtie r_2) \Leftrightarrow \pi_X(\pi_Y(r_1) \bowtie \pi_Z(r_2));\\ \text{mit } Y := (X \cap R_1) \cup (R_1 \cap R_2),\\ Z := (X \cap R_2) \cup (R_1 \cap R_2).\end{aligned} \tag{5.14}$$

Eine ähnliche Regel existiert für die Kommutativität von Projektion und Vereinigung (siehe Anhang C.3, Regel `K18`). Aus Sichtweise des Datenschutzes können durch vorzeitige Projektionen die Attribute eines Quasi-Identifikators eingeschränkt werden. Kann für jeden minimalen QI mindestens ein Attribut entfernt werden, so lassen sich einzelne Tupel nicht mehr ausreichend identifizieren. Da für den Verbund häufig Schlüssel bzw. Fremdschlüssel genutzt werden, können verständlicherweise nicht alle identifizierbaren Merkmale entfernt werden. Durch die vorzeitigen Projektionen wird jedoch gewährleistet, dass nur das Minimum der benötigten Daten an den nächsten Auswertungsoperator bzw. -knoten übermittelt wird.

Kommutativität von zwei Selektionen: Werden innerhalb einer Selektion zwei Prädikate P_1 und P_2 mit dem logischen Und verknüpft, so können die beiden Prädikate auch in zwei hintereinander ausgeführte Selektionen eingesetzt werden. Durch die Kommutativität des logischen Unds lässt sich zudem die Reihenfolge der Prädikate vertauschen:

$$\begin{aligned}\sigma_{P_1}(\sigma_{P_2}(r)) &\Leftrightarrow \sigma_{P_1 \wedge P_2}(r)\\ &\Leftrightarrow \sigma_{P_2}(\sigma_{P_1}(r)).\end{aligned} \tag{5.15}$$

Für die Vorfilterung von Daten in ressourcenbeschränkten Umgebungen lässt sich diese Regel wie folgt ausnutzen: Wird ein Operator aus dem Prädikat P_1 nicht durch die Kapazitäten eines lokal eingesetzten Gerätes unterstützt, so lässt sich das Prädikat P_2 getrennt von P_1 ausführen. Voraussetzung hierfür ist, dass jeder Operator aus P_2 lokal unterstützt wird. Das Zwischenergebnis $r' := \sigma_{P_2}(r)$ lässt sich anschließend weiterleiten, damit auf einem externen Knoten das verbleibende Selektionsprädikat P_1 ausgeführt werden kann und das Ergebnis der Anfrage $r'' := \sigma_{P_1}(r')$ berechnet wird.

Kommutativität von Selektion und Projektion: Selektion und Projektion kommutieren, falls alle Attribute im Selektionsprädikat P in der Projektionsliste X vollständig vorhanden sind:

$$\sigma_P(\pi_X(r)) \Leftrightarrow \pi_X(\sigma_P(r)), \quad \text{falls } attr(P) \subseteq X. \tag{5.16}$$

Ist dies nicht möglich, kann die Projektionsliste um eine weitere Attributmenge Y ergänzt werden, welche die restlichen Attribute aus P enthält:

$$\pi_X(\sigma_P(\pi_{X,Y}(r))) \Leftrightarrow \pi_X(\sigma_P(r)), \quad \text{wobei } attr(P) \subseteq X_1 \cup X_2. \tag{5.17}$$

Aus Datenschutzsicht lassen sich diese Regeln derart nutzen, dass auf dem lokalen Gerät, je nach Anfragekapazität, die Projektion oder Selektion lokal ausgeführt wird. Der verbleibende Operator verarbeitet anschließend auf einem externen Knoten das Zwischenergebnis weiter.

Weitere Regeln: Neben den genannten Regeln existieren noch viele weitere Optimierungen, die aber vorwiegend auf Distributivität und Kommutativität zwischen Mengenoperatoren und anderen Operationen sowie Idempotenzen beruhen. Im Anhang C, insbesondere im Abschnitt C.3, wird eine Übersicht zu den gängigen Optimierungsregeln angeführt. Dabei wird schwerpunktmäßig darauf eingegangen, wie diese Regeln zur Umsetzung der Datensparsamkeit genutzt werden können.

Anwendung der Regeln: Aus den einzelnen Regeln lässt sich ein einfacher Algorithmus zur Optimierung von Anfragen ableiten. Im ersten Schritt werden komplexe Selektionsprädikate aufgelöst, indem die de Morganschen Regeln zur Auflösung von Negation und Oder-Verknüpfungen angewandt werden. Die erzeugte konjunktive Normalform lässt sich anschließend durch die Kommutativität in mehrere kleine Selektionsprädikate aufspalten. Der zweite Schritt schiebt die entstandenen Selektionsprädikate durch die mehrfache Anwendung der Regeln zum Vertauschen mit Verbunden und Mengenoperatoren soweit wie möglich in Richtung der Blätter des Anfragebaumes. Abschließend werden die Projektionen ebenfalls nahe an die Blätter geschoben. Dabei entstandene überflüssige Projektionen werden abschließend entfernt.

In Anhang D wird ein komplexeres Beispiel zur Anwendung der Regeln aufgezeigt. Dabei kommen auch Transformationen zum Einsatz, welche auf der Ausnutzung von funktionalen und Inklusionsabhängigkeiten basieren sowie die Einbindung von Aggregatfunktionen betrachten.

5.9 Design by Contract

Bertrand Meyer stellt in [Mey92] eine Methode bzw. Richtlinie zur Verbesserung der Verlässlichkeit von objektorientierter Software vor. In der Praxis wird der im Folgenden vorgestellte Ansatz in den objektorientierten Programmiersprachen Eiffel und C++[8] angewendet. *Design by Contract* soll Entwicklern dabei helfen, die Qualität ihrer Softwareprodukte durch die Vermeidung von Fehlern zur Laufzeit zu erhöhen.

[8]Contracts sind ab dem Standard C++20 verfügbar, siehe `https://isocpp.org/std/the-standard`, zuletzt aufgerufen am 05.01.2022.

Ein Kontrakt besteht aus Vorbedingungen, Nachbedingungen und Invarianten. Die *Vorbedingungen* einer Funktion müssen vor dessen Aufruf erfüllt sein, ansonsten wird dessen Ausführung verhindert. Dies kann beispielsweise die Forderung sein, dass die übergebenen Parameter an die Methode keine Nullwerte enthalten. *Nachbedingungen* beinhalten alle Forderungen, die nach der Ausführung einer Methode gelten müssen. So wird beispielsweise sichergestellt, dass die Ausgabe der Methode nicht mit dem Nullwert übereinstimmt. *Invarianten* gelten für alle Objekte einer Klasse während deren gesamten Lebensdauer. Sie werden erstmalig beim Erzeugen eines Objektes überprüft und anschließend bei jedem Test von Vor- und Nachbedingungen mit überprüft. So kann beispielsweise sichergestellt werden, dass die Summe zweier Kontostände vor und nach einer Überweisung gleich bleibt. Während der Ausführung einer Methode können die Invarianten hingegen auch verletzt sein.

Beispiel: Design by Contract in C++20

Der folgende Quellcodeausschnitt zeigt die Verwendung von Contracts in C++20. Die Funktion `calculateFilesize` überprüft, ob das Produkt von `length` und `bitrate` mit der global deklarierten, privaten Variable `filesize` übereinstimmt. Durch die Schlüsselwörter `expects` und `ensures` werden Vor- bzw. Nachbedingungen spezifiziert (Test auf positive Werte); das Schlüsselwort `assert` bestimmt hingegen eine Invariante (Vergleich der Werte).

```
1  class Track {
2    public:
3      int calculateFilesize(int length, int bitrate)
4      [[expects: length > 0]]
5      [[expects: bitrate > 0]]
6      [[ensures: length * bitrate > 0]]{
7        [[assert: length * bitrate == filesize]];
8        return length * bitrate;
9      }
10
11   private:
12     int filesize;
13 };
```

Kontrakte können über Vererbung auch an abgeleitete Klassen vererbt werden, um auf überschriebene Methoden, Polymorphie und dynamisches Binden reagieren zu können. Dabei werden sogenannte Subkontrakte erstellt, welche nur für die abgeleiteten Unterklassen gelten.

Kontrakte bilden zudem eine Alternative zu klassischen Fehlerbehandlungen, wie sie aus anderen Programmiersprachen bekannt sind. Dazu können, neben dem Abfangen oder Weitergeben von Fehlern, sogenannte Rettungsanweisungen formuliert werden, welche das Objekt zurück in einen konsistenten Zustand bringen sollen. Dazu kann beispielsweise versucht werden, eine Methode im Fehlerfall erneut auszuführen, oder eine Rückführung in den zuletzt bekannten, konsistenten Zustand erfolgen.

In der objektorientierten Programmiersprache Java werden ab Version 1.4 sogenannte *Assertions* unterstützt[9]. Über das Schlüsselwort `assert` können im Programmcode Invarianten spezifiziert werden. Über externen Bibliotheken, wie *Contracts for Java* von Google[10], lassen sich über Annotationen zudem Vor- und Nachbedingungen im Programmcode angeben. Java-Derviate, wie Kotlin, bieten zudem auch die direkte Unterstützung von Design by Contract[11] an.

[9] `https://web.archive.org/web/20080616233205/http://www.jcp.org/en/jsr/detail?id=41`, zuletzt aufgerufen am 05.01.2022

[10] `https://opensource.googleblog.com/2011/02/contracts-for-java.html`, zuletzt aufgerufen am 05.01.2022

[11] `https://github.com/JetBrains/kotlin/blob/master/libraries/stdlib/src/kotlin/util/Preconditions.kt`, zuletzt aufgerufen am 05.01.2022

5.10 Stand der Forschung und Stand der Technik: Zusammenfassung

In diesem Abschnitt wurden Konzepte zur Lösung des Query-Containment-Problems für Anfragen mit Rekursion, Aggregaten und verschiedenen Arten von Abhängigkeiten vorgestellt. Was in den bisherigen Ansätzen bisher fehlt, sind Erweiterungen für relationale Anfragen mit beliebigen Kombinationen aus komplexeren Aggregatfunktionen, Gruppierungen sowie Fensterfunktionen. Diese Operatoren sind zentraler Bestandteil von Auswertungen in smarten, assistiven IoT-Umgebungen bzw. -Anwendungen. Mit bestehenden Ansätzen lässt sich die Klasse von komplexen Anfragen nur unzureichend analysieren.

Zudem müssen die Anfragekapazitäten der einzelnen Geräte bzw. Knoten als Parameter in den Transformationsprozess der Anfrage integriert werden. Die bisherigen Ansätze zur kapazitätsbasierten Anfragetransformation wurden bisher nur für einfache SPJ-Anfragen genutzt. Dadurch sind die bisher entwickelten Techniken zur Verarbeitung und Optimierung von Anfragen für eine Übertragung auf die in Abschnitt 1.1.1 definierte Zielstellung nur beschränkt anwendbar. Für die Erarbeitung einer Strategie, bei der Daten durch eine schrittweise Vorberechnungen bereits lokal und ohne Informationsverlust anonymisiert werden, muss gewährleistet sein, dass die Exaktheit und Vollständigkeit der Ergebnisse während der Verarbeitung erhalten bleibt.

Abgrenzung zu anderen Techniken: Die vorgestellten Ansätze umgehen die Schwierigkeiten des Query-Containment-Problems durch Einschränkung der Anfragesprache auf Operatoren, wie Wertevergleiche und ähnliche einfache Konstrukte, um bestimmte, abgegrenzte Problemklassen zu lösen. Auch komplexere Operatoren, wie Aggregationen, bzw. deren Komposition, werden meist aus der betrachteten Anfragesprache entfernt. Da Aggregationen (Summen, Minimum, Medianwerte) und deren Anwendung aber ein wesentlicher Bestandteil von Algorithmen in komplexen Analysefunktionen sind, müssen diese Konstrukte in der Anfragesprache enthalten bleiben.

Die nachfolgende Tabelle 5.1 gibt einen Überblick über die vorgestellten Verfahren und deren Einsatz im Anfrageprozessor von PArADISE. Bei den umgesetzten bzw. geeigneten Verfahren wurde nicht die eigentliche Umsetzung des Verfahrens betrachtet, sondern vielmehr inwieweit sich die Ergebnisse der Arbeiten für auf das eigene Problem zur Transformation von Anfragen übertragen lassen. Die so gewonnenen Transformationsregeln sind in Anhang C aufgeführt.

Answering Queries using Operators: Im nachfolgenden Kapitel 6 wird gezeigt, wie die Transformationsregeln des Query Containments, insbesondere für Aggregatanfragen nach [Tür99] sowie den klassischen Optimierungsregeln nach [GM93] bzw. [HFLP89], mit Techniken der Kontrakt-basierten Programmierung kombiniert werden können. Durch die Kombination beider Teilthemen lassen sich neue Regeln zur Anfragetransformation und zur vertikalen Verteilung von Anfragen in kapazitätsbeschränkten Umgebungen bilden. Der vorgestellte Ansatz erzeugt die vom Bundesdatenschutzgesetz gestellte Forderung nach Datensparsamkeit – dessen konzeptionelle und technische Realisierung ist die zentrale Zielsetzung dieser Arbeit.

Klasse	**umgesetzt**	**geeignet, aber nicht umgesetzt**	**ungeeignet**
Äquivalente Anfragen	[GM93] bzw. [HFLP89] [a]		
Anfragen mit Rekursion			[ZYI+17] [b] [DG97] [c]
Anfragen mit Aggregaten	[DZT13] (teilweise) [TNP14] [HT15] [Tür99]	[Müh02]	
Anfragen mit Abhängigkeiten	[Tür99] und [Gry99] [d]	[CCS18] [DLN05] [DPT06] [e]	[GOP14a] [f] [MHF03] [g]
Spezielle Klassen von Anfragen	[CWCS16] (teilweise) Big SQL (teilweise)	[DCLL15] [ZSA09] [KSK05]	
Ressourcenbeschränkte Umgebungen	[LRU99] [VP00]	[PGH98] [GLY99] [LRO96b]	
Vorverarbeitung von Informationen	[AFB07] [SW74]	[WMYR16] [FMLJ16] [YLX14]	
Design by Contract	[Mey92]		

[a] teils in der Anfragetransformation realisiert, teils optimiert mittels Apache Calcite
[b] zu starke Einschränkungen
[c] Containment-Eigenschaft in falscher Richtung
[d] im Rahmen einer studentischen Arbeit [Kas21] umgesetzt
[e] siehe Ausblick in Unterabschnitt 7.2.1
[f] aufgrund der fehlenden Terminierung
[g] Vollständigkeit der Ergebnisse nicht sichergestellt

Tabelle 5.1: Übersicht über die in diesem Kapitel vorgestellten Verfahren und deren Umsetzung bzw. Eignung für das AQuO-Konzept.

Kapitel 6

Datenschutzfreundliche Verarbeitung und Verteilung von Anfragen

In diesem Kapitel wird das Herzstück des Anfrageprozessors vorgestellt – die Anfragetransformation einer komplexen Anfrage in eine Folge von Anfragen, welche unter Berücksichtigung der lokalen, begrenzten Anfragesprache den Aspekt der Datensparsamkeit realisiert. Die Einordnung des entwickelten Verfahrens in das Gesamtkonzept des Anfrageprozessors ist in Abbildung 6.1 zu sehen.

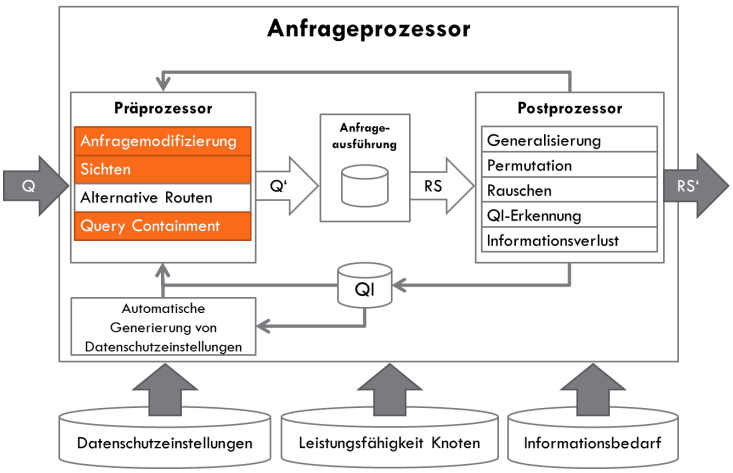


Abbildung 6.1: Einordnung der Anfragetransformation in den Anfrageprozessor

---

Bisherige Forschungsansätze zur Transformation von Anfragen (siehe Kapitel 5) fokussierten sich auf die Suche nach einer maximal enthaltenden Menge von Anfragetransformationen, die eine partielle, aber möglichst maximale Teilmenge der Ergebnisse einer Anfrage Q zurückliefern. In ressourcen-beschränkten Umgebungen, wie beispielsweise Sensornetzwerken (siehe Abbildung 6.2), kann nicht immer sichergestellt werden, dass ein einzelner Verarbeitungsknoten jede Art von Anfrage abarbeiten kann. Die Ursache liegt dabei in der beschränkten Leistungsfähigkeit einzelner Knoten.

Die unterschiedlichen Geräte bilden eine Verarbeitungskette zur Beantwortung komplexer Anfragen, die beispielsweise von webbasierten Diensten gestellt werden. Werden alle Daten zur Verarbeitung in das Rechenzentrum des Dienstanbieters geschickt, entstehen augenblicklich Datenschutzbedenken aufgrund der unbeschränkten Analysemöglichkeiten auf den Rohdaten. Diese Bedenken können stark vermindert werden, wenn durch eine schritt-

weise Vorverarbeitung der Daten auf jedem Knoten nur geringe Teile der erzeugten Daten an den Dienstanbieter übertragen werden.

Werden die Eigenschaften beginnend bei der Cloud-Ebene hin zur Sensorebene betrachtet, so werden die Beschränkungen hinsichtlich der zur Verfügung stehenden Ressourcen stets größer, während die verfügbaren Datenbankfunktionalitäten und -operatoren abnehmen. Andererseits hat ein Nutzer größere Kontrolle über die Geräte, welche außerhalb der Cloud angesiedelt sind, da sie in seinem persönlichen Besitz sind. Dadurch wird es dem Nutzer ermöglicht, zu steuern, welche Daten letztendlich in die Cloud gelangen. Durch dieses Vorgehen wird zudem eine optimierte Anfrageverarbeitung hinsichtlich der Ressourcenbeschränkung erreicht, da aufgrund der Verringerung des Datenvolumens nicht nur das Datenschutzniveau erhöht wird, sondern auch die Effizienz des Systems.

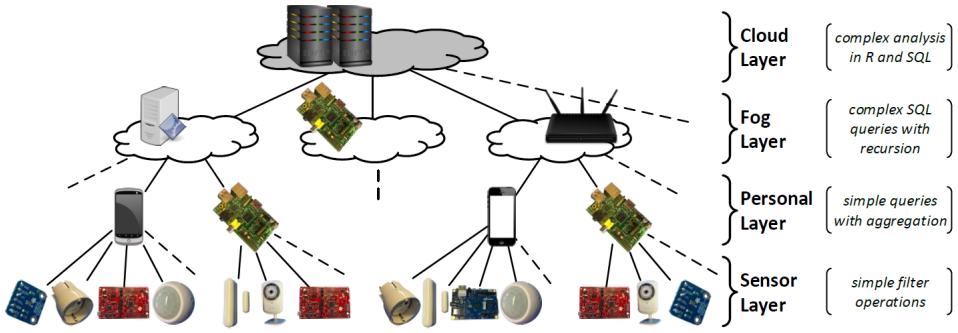


Abbildung 6.2: Diese Abbildung zeigt die Ebenen bei der Anfrageverarbeitung im Cloud / Fog-Umfeld. Die Architektur besteht aus vier logisch getrennten Ebenen, wobei eine Ebene aus mehreren, aufeinander aufbauenden Geräten und Sensoren bestehen kann. In der Sensorebene (Sensor Layer) werden Sensoren (z. B. Temperatur-, Licht- und Bewegungssensoren) eingesetzt, die hinsichtlich ihrer Rechenleistung, dem verfügbaren Speicher und der zur Verfügung stehenden Energiezufuhr stark beschränkt sind. Die persönliche Ebene (Personal Layer) umfasst Mobilgeräte, wie Mobiltelefone, Tablets und Handhelds, sowie eingebettete Systeme, wie sie beispielsweise in Autos oder in kleineren Netzwerkgeräten vorkommen. Auf der Fog-Ebene (Fog Layer) werden Geräte wie Router, Home Media Center (Spielekonsolen, Boxen zum Video-Streaming, ...) oder lokale Server eingesetzt. Die Cloud-Ebene (Cloud Layer) besteht aus umfassenden Servernetzwerken und Datenzentren, wie sie bei Amazon oder IBM vorkommen. Diese Ebene umfasst Geräte mit fast uneingeschränktem Zugang zu Ressourcen, um umfangreiche Berechnungen auf großen Datenmengen durchzuführen. (Initial erschien diese Abbildung in [GKB+16].)

In diesem Kapitel wird ein neues Verfahren vorgestellt (siehe Abschnitte 6.1 bis 6.4), welches für eine gegebene Anfrage Q eine Folge von Anfragen Q_i berechnet. Jedes Q_i wird dabei auf einem Knoten i ausgeführt, welcher über eine eingeschränkte Menge an Operatoren (im Sinne der relationalen Algebra) verfügt und eine minimale Obermenge an Daten zum nächsten Knoten i+1 innerhalb der Verarbeitungskette weiterleitet. Der restliche Anteil der Anfrage, Q_n, übernimmt die abschließenden Operatoren, wie verbleibende Selektionsprädikate und Aggregationen, auf dem höchsten Knoten n:

$$Q(D) := Q_n(Q_{n-1}(\dots Q_1(D))) \tag{6.1}$$

Das entwickelte Query-Rewriting-Verfahren ist als Komponente des Postprozessors im vorgestellten Anfrageprozessor integriert (siehe Abschnitt 6.5). In Abschnitt 6.6 wird das entwickelte Verfahren hinsichtlich der Laufzeitunterschiede zwischen Original- und transformierter Anfrage anhand mehrerer Beispielszenarien evaluiert. Die Zusammenfassung in Abschnitt 6.7 gibt einen Überblick über die erzielten Erkenntnisse und diskutiert die Vor- und Nachteile des entwickelten Verfahrens.

Beispielanfrage: Als laufendes Beispiel für dieses Kapitel soll die folgende Beispielanfrage Q dienen:

Beispiel: Laufendes Beispiel für dieses Kapitel

Gesucht ist die Gesamtlaufzeit der Musikstücke (`length`) pro Genre (`gid`). Dabei sollen nur diejenigen Musikstücke betrachtet werden, welche die folgenden Eigenschaften erfüllen:

- Die IDs der einzelnen Musikstücke, `tid`, muss größer oder gleich `42` sein.
- Die Laufzeit einzelner Musikstücke muss zwischen `200` und `500` Sekunden liegen.
- Zwecks Konsistenzüberprüfung muss pro Genre das durchschnittliche Anlegedatum (`createdate`) pro Genre vor dem durchschnittlichen Änderungsdatum (`modifydate`) liegen.
- Der Anstieg der Regressionsgeraden von Dateigröße (`filesize`) und der Länge der Musikstücke pro Genre muss größer als `25` sein.

Als SQL-Anfrage wird dieser Sachverhalt wie folgt realisiert:

```
1 SELECT gid, sum(length)
2 FROM tracks
3 WHERE tid >= 42
4 AND length BETWEEN 200 AND 500
5 GROUP BY gid
6 HAVING AVG(createdate) < AVG(modifydate)
7 AND regr_slope(filesize, length) > 25
```

Als Anwendungsszenario können solche Arten von Anfragen zur Steuerung von Musikanlagen genutzt werden, um gezielt bestimmte Musiktitel abzuspielen. Während die Daten aus der Musikdatenbank einerseits für die Beantwortung dieser Anfrage genutzt werden, kann aber auch eine zweckentfremdete Nutzung der Daten, beispielsweise zur Erkennung von Medien-Piraterie [EGKP11] erfolgen.

Dies ist sicher nicht im Interesse des Nutzers, da die Zweckbindung nicht eingehalten wird. Im Laufe dieses Kapitels werden wir sehen, wie die Anfrage derart modifiziert wird, dass keine „bösartigen" Auswertungen erfolgen können, während die obige Anfrage weiterhin ohne Informationsverlust ausgeführt werden kann.

Beitrag: In diesem Kapitel wird untersucht, wie Algorithmen und Regeln von bisherigen Query-Containment-Ansätzen adaptiert werden können, um eine gezielte Anfragetransformation unter Wahrung von Datenschutzaspekten zu erzielen. Um dies zu erreichen, werden bestehende Query-Containment-Techniken mit dem *Design-by-Contract*-Konzept aus der Programmiersprache Eiffel zu einem eigenen Ansatz, *Query Rewriting by Contract*, kombiniert. Als Ergebnis der Untersuchung entstanden ein Algorithmus und ein Regelverzeichnis zur Anfragetransformation, welches im Anhang C aufgelistet ist.

Literatur: Dieses Kapitel basiert auf einer Reihe früherer Publikationen [GH16b, GKB+16, GH17, GH18], in denen das entwickelte Verfahren und dessen Erweiterungen vorgestellt wurden. In Abgrenzung zu den bisherigen Veröffentlichungen wird in diesem Kapitel zusätzlich das Zusammenspiel der einzelnen Teilkomponenten betrachtet. Zudem wurde die Beschreibung des Konzepts zur Anfragetransformation um komplexere Anfragen ergänzt. Im Abschnitt zur Implementierung des Verfahrens wird ausführlich auf die entwickelten Algorithmen und die dabei verwendeten Techniken und Werkzeuge eingegangen. Weiterhin erfolgt eine umfangreichere Evaluation des Verfahrens anhand der drei Testszenarien, die bereits aus Abschnitt 4.5 bekannt sind.

6.1 Konzept

Der vorgestellte Ansatz erfüllt die vom Bundesdatenschutzgesetz gestellte Forderung nach Datensparsamkeit, indem eine komplexe Anfrage zunächst vertikal in Anfragefragmente aufgeteilt wird. Das Ergebnis jedes Anfragefragmentes enthält eine Obermenge dessen, was als Resultat der Originalanfrage zurückgeliefert werden würde. Jedes dieser Fragmente wird so erzeugt, dass es auf dem jeweils niedrigsten Knoten mit ausreichender Anfragekapazität berechnet werden kann.

Die Regeln bewirken ein erhöhtes Datenschutzniveau durch zusätzliche Vorselektionen und -aggregationen. Dies schließt komplexe Aggregatfunktionen, wie Regression- und Korrelationsanalysen, ebenso wie die Erweiterung der Anfragen um Gruppierungsklauseln mit ein. Durch die Integration in den entwickelten Anfrageprozessor lassen sich somit alle Arten von OLAP-Anfragen sowie die Basisoperatoren für maschinelle Lernverfahren beantworten.

Der Anfrageprozessor wird in einem Informationsnetzwerk auf jeden Datenknoten, vom Sensor bis zum Cloud-Server, aufgespielt. Wird die beschriebene Anfragetransformation rekursiv auf jedem Knoten ausgeführt, entsteht ein automatisiertes Edge-Computing-Netzwerk, welches beliebige Anfragen in einzelne Anfragefragmente zerlegt und deren Ergebnisse ohne Informationsverlust wieder zusammenführt.

6.1.1 Allgemeiner Ansatz und Basisdefinitionen

Gegeben sei eine Anfrage Q und eine Menge von Ebenen L. Durch die gewünschte Transformation von Q soll eine partielle Anfrage Q_1 und eine Restanfrage Q_δ entstehen. Q_1 wird so gewählt, dass es durch die lokal zur Verfügung stehenden Anfragekapazitäten auf der Ebene L_1 ausgeführt werden kann, während Q_δ auf die nächsthöhere Ebene L_2 verschoben wird. Das Ergebnis von Q_1 auf der Datenbank D wird mit $D_1 := Q_1(D)$ bezeichnet. Verfügt L_2 über alle für Q_δ notwendigen Operatoren, so wird Q_δ auf L_2 ausgeführt und das Endergebnis $D_\delta := Q_\delta(D_1)$ ausgegeben.

Falls die Anfrage Q_δ nicht vollständig auf L_2 unterstützt werden kann, so wird Q_δ erneut in eine partielle Anfrage Q_2 und eine Restanfrage $Q_{\delta'}$ aufgeteilt. Dies wird solange wiederholt, bis die transformierte Anfrage vollkommen unterstützt wird bzw. die Cloud-Ebene L_n, welche o. B. d. A. alle Operatoren unterstützt, erreicht ist. Durch die wiederholte Aufteilung der Anfrage wird eine Anfragekette (siehe Gleichung 6.1) erzeugt.

Zur Bestimmung der Teilanfragen Q_i werden die in Kapitel 5 eingeführten Definitionen und Techniken angewandt. Ausgehend von dem Query-Containment-Problem (siehe Definition in 5.1) definieren wir das *Answering-Queries-using-Operators-Problem* (AQuO) wie folgt:

Definition: Answering Queries using Operators

Gegeben sei ein Datenbankschema D, eine Anfrage Q und eine Menge von Ebenen L, wobei jede Ebene $L_i \in L$ eine Menge an Operatoren O_i unterstützt. Durch AQuO wird eine Transformation $transform$ mit $transform(Q) = Q_i$ gesucht, sodass

$$\begin{aligned} Q_i(D) \sqsupseteq Q(D) \;\Leftrightarrow \\ \forall d(D) : Q_i(d) \supseteq Q(d) \end{aligned} \tag{6.2}$$

gilt und die Anfrage Q_i nur Operatoren aus O_i verwendet.

Ähnlich zum *Maximally Contained Rewriting* (siehe Definition in Abschnitt 5.1.1) wird beim AQuO-Problem nach einer Anfrage gesucht, welche die *minimale Teilmenge an zusätzlichen Tupeln* enthält. Die Anfrage bezeichnen wir im Folgenden als *Rewriting Supremum*:

Definition: Rewriting Supremum

Eine transformierte Anfrage von Q für ein Datenbankschema D nennt man genau dann ein Rewriting Supremum Q_i, wenn

- keine Anfrage Q_i' existiert, für die gilt, dass diese weniger Tupel zurückliefert als Q_i, jedoch weiterhin eine Obermenge von Q ist:

$$\nexists Q_i' : Q_i(D) \sqsupset Q_i'(D) \sqsupseteq Q(D) \text{ und} \tag{6.3}$$

- Q_i und Q_i' nur Operatoren aus O_i nutzen.

Im besten Fall gilt, parallel zu AQuV, dass die Ergebnismengen von Q_i und Q äquivalent sind ($Q_i(D) \equiv Q(D)$).

Das *Rewriting Supremum* unterscheidet sich vom *maximally contained set of rewritings* in zwei Punkten: Zum einen wird statt einer Menge von verfügbaren Sichten eine Menge von erlaubten Operatoren vorgegeben. Diese Operatoren können dazu genutzt werden, um Templates für eine beliebig große Menge von Sichten zu generieren. Zudem wird die Richtung der Vergleichsoperatoren umgekehrt, um die Näherung an die Originalanfrage aus der einschränkenden Sichtweise zu beschreiben. Zur Veranschaulichung betrachten wir das folgende, einfache Beispiel:

Beispiel: Bestimmung des Rewriting Supremums für einen einfachen Fall

Durch eine Anfrage Q sollen alle Musiktitel mit einer maximalen Laufzeit von weniger als 120 Sekunden bestimmt werden. Diese Anfrage lässt sich in der Relationenalgebra wie folgt realisieren:

$$\pi_{Title}(\sigma_{length<120}(tracks)) \tag{6.4}$$

Stehen auf der Ebene L_1 nur die Konstantenselektionen $\sigma_{\neq_c}$ und $\sigma_{\leq_c}$ zur Verfügung ($O_1 = \{\sigma_{\neq_c}, \sigma_{\leq_c}\}$ bzw. in Kurzform $O_1 = \{\neq_c, \leq_c\}$), so lassen sich folgende Anfragen Q_1 und Q_1' auf der Musikdatenbank formulieren:

$$Q_1 = \pi_{Title}(\sigma_{length\leq 120}(tracks)) \tag{6.5}$$

bzw.

$$Q_1' = \pi_{Title}(\sigma_{length\neq 120}(tracks)) \tag{6.6}$$

Beide Anfragen liefern eine Obermenge der gewünschten Ergebnisrelation. Offensichtlich ist die erste Variante restriktiver, da sie zusätzlich nur die Musikstücke enthält, die genau 120 Sekunden lang sind. Q_1 bildet somit ein mögliches Rewriting Supremum. Da jedoch auch eine überlappende Containment-Beziehung zwischen den Vergleichsoperatoren $\leq_c$ und $\neq_c$ besteht, ist Q_1' ebenfalls ein Rewriting Supremum.

Steht zusätzlich zu $\neq_c$ und $\leq_c$ das logische UND ($\wedge$) als Operator zur Verfügung ($O_1' = \{\neq_c, \leq_c, \wedge\}$), lassen sich beide Selektionsprädikate miteinander verknüpfen:

$$Q \equiv Q' = \pi_{Title}(\sigma_{length\leq 120 \wedge length\neq 120}(tracks)) \tag{6.7}$$

Diese Anfrage ist – zumindest für Integer-Werte als Wertebereich des Attributes `length` – äquivalent zur Originalanfrage und somit restriktiver als Q_1 und Q_1'. Dadurch wird Q' zum alleinigen Rewriting Supremum. Wie das Rewriting Supremum formal bestimmt werden kann, wird in den nachfolgenden Abschnitten genauer erklärt.

6.1.2 Rewriting Supremum

Im Gegensatz zum *maximally contained set of rewritings* aus dem AQuV-Problemfeld wird für die hier vorgeschlagene Anfragetransformation eine Annäherung an die Originalanfrage aus einer anderen Blickrichtung ge-

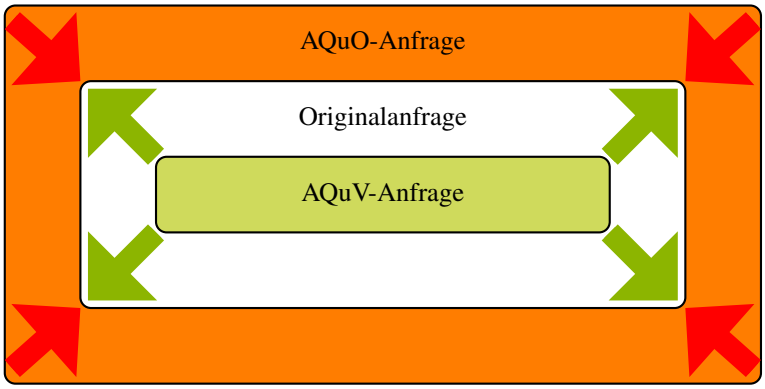


Abbildung 6.3: Visualisierung der *maximal enthaltenden Anfrage* (grün) und des *Rewriting Supremums* (rot)

sucht: Die transformierte Anfrage muss mindestens alle Informationen enthalten, die auch durch die Originalanfrage zurückgegeben werden, jedoch soll der Überschuss an Informationen möglichst gering gehalten werden. Im besten Fall sind beide Anfragen äquivalent, d. h., sie liefern das exakt gleiche Ergebnis zurück. Abbildung 6.3 zeigt das Verhältnis zwischen AQuV und AQuO. Die grünen Pfeile symbolisieren dabei das *maximally contained set of rewritings*, die roten Pfeile hingegen das *Rewriting Supremum*.

Dieses Vorgehen erscheint zunächst widersprüchlich zu der Forderung der Datensparsamkeit zu sein: Statt die minimal notwendige Datenmenge zur Erfüllung der Anfrage zurückzuliefern, werden mehr Daten übertragen. Wird jedoch bedacht, dass ohne die Transformation im Extremfall alle von den Sensoren erzeugten Daten ohne Vorfilterungen und Aggregationen in die Cloud übertragen werden, so wird die Datenmenge im direkten Vergleich stark reduziert. Zur Erzeugung eines äquivalenten Ergebnisses müssen auf den Rechenknoten der höheren Ebenen zusätzliche Operatoren ausgeführt werden.

Im mathematischen Sinne besitzt jede von oben beschränkte, partiell geordnete Menge ein Supremum als kleinste aller oberer Schranken [BSMM06]. Bezüglich des relationalen Modells stellt eine obere Schranke die maximal mögliche Anzahl an Tupeln dar, die unter festgelegten Bedingungen erreicht werden. Für das AQuO-Szenario werden diese Bedingungen durch die zur Verfügung stehenden Operatoren festgelegt. Als *Rewriting Supremum* wird demnach diejenige Transformation der Anfrage definiert, welche unter den einschränkenden Bedingungen die Ergebnismenge am geringsten erweitert. Durch die Rückgabe einer Obermenge der Ergebnisse können weiterhin alle Anfragen beantwortet werden; dies wäre bei Teilmengen nicht der Fall.

Anforderungen an das Rewriting Supremum: Für verteilte Datenbanken existieren mehrere Anforderungen [Nia78] an die Fragmentierung der Daten bei der Erstellung des Verteilungsentwurfes. Diese lassen sich auch teilweise auf die Anforderungen für eine verteilte Anfrageverarbeitung übertragen:

A1 **Rekonstruierbarkeit:** Fragmentierte Relationen lassen sich ohne Informationsverlust aus den einzelnen Fragmenten wiederherstellen. Gleiches gilt auch für die fragmentierte Anfragen: Wird eine Anfrage auf mehrere Knoten verteilt und ggf. in ihrer Ausführungsreihenfolge verändert, so muss sich die originale Anfrage aus den Anfragefragmenten wieder rekonstruieren lassen. Dadurch lässt sich die Äquivalenz der originalen Anfrage und der fragmentierten Anfrage sicherstellen.

A2 **Vollständigkeit:** Jedes Tupel bzw. Attribut ist mindestens einem Fragment zugeordnet. Um die vollständige Rekonstruierbarkeit der Anfrage zu gewährleisten reicht es nicht nur aus, dass die ursprüngliche und transformierte Anfrage semantisch äquivalent sind, sondern dass auch alle Anfragefragmente die jeweils richtigen Datenquellen adressieren.

A3 **Disjunktheit:** Jedes Tupel bzw. jeder Attributwert, mit Ausnahme von Schlüsselattributen, ist nicht mehreren Fragmenten zugeordnet. Aus Sicht der Anfragezerlegung wird es, wie später zu sehen ist, vorkommen,

dass mehrere Anfragefragmente gleich strukturiert sind und auf den gleichen Daten operieren. Um sicherzustellen, dass die Anfragefragmente nicht mehrfach berechnet werden, müssen äquivalente Teilbäume erkannt und miteinander verknüpft werden. Durch eine Referenzierung auf ein einmalig berechnetes Zwischenergebnis wird eine mehrfache Berechnung verhindert. Zudem wird ein mehrfaches Übertragen des gleichen Datensatzes an eine höhere Ebene verhindert, wodurch die Kommunikationskosten gesenkt werden können. Im Gegensatz zu verteilten Datenbanken kann ein berechnetes Zwischenergebnis an mehrere Knoten verteilt werden, sofern es für den berechneten Anfrageplan notwendig ist.

Das Rewriting Supremum muss für das in Abbildung 6.2 aufgespannte Edge-Netzwerk für jeden an der Anfrageverarbeitung beteiligten Knoten bestimmt werden. Dabei kann, insbesondere auf den leistungsschwächeren Knoten, die Äquivalenz der Anfragefragmente zur Originalanfrage nicht sichergestellt werden. Dadurch werden von diesen Knoten mehr Daten zurückgegeben als für das Gesamtergebnis benötigt werden. Das *mehr* an Daten spiegelt sich in mindestens einer der folgen Ausprägungen wider:

- In der Projektionsliste sind zusätzliche Attribute enthalten.
- Die Selektionsbedingungen sind weniger restriktiv und liefern mehr Tupel zurück.
- Aggregatfunktionen werden entweder gar nicht (Rohdaten) oder nur teilweise (voraggregierte Werte) ausgeführt.
- Durch fehlende Verbundbedingungen werden sogenannte *dangling tuples*, also Tupel ohne Verbundpartner, noch nicht erkannt.

Der in diesem Kapitel entwickelte Ansatz zur Bestimmung des Rewriting Supremums verbindet eine heuristische Suchstrategie mit einem Divide-and-Conquer-Ansatz: Die gegebene komplexe Anfrage Q wird in mehrere kleine Anfragefragmente Q_i zerlegt, die lokal ausgeführt werden, um anschließend das Anfrageergebnis vom nächsten Teilfragment verarbeiten zu lassen. Die Zerlegung erfolgt dabei einerseits durch bekannte Verfahren der Anfrageoptimierung (siehe Kapitel 5), andererseits durch zusätzliche Einschränkungen, die sich aus dem Funktionsumfang des jeweiligen Datenbanksystems ergeben. In den nachfolgenden Abschnitten wird beschrieben, wie diese Zerlegung effizient bestimmt werden kann.

6.1.3 Meta-Algebra

Nach Einführung der Ebenenarchitektur und des groben Konzepts zur Verteilung von Anfragen in Anfragefragmente wird zunächst die zugrunde liegende formale Algebra eingeführt:

Definition: Meta-Algebra

Die Meta-Algebra L ist ein Tupel der Form

$$L := (O, D), \tag{6.8}$$

wobei O die Menge der unterstützten Operatoren ist und D eine Multimenge von getypten Daten darstellt.

Die unterstützten Operatoren können beispielsweise eine Teilmenge der relationalen Algebra oder die Aggregatfunktionen eines eingeschränkten SQL-Dialektes sein. Analog dazu sind die Daten z. B. eine Menge von Relationen oder Sichten. Zur Veranschaulichung der Algebra fangen wir mit einem einfachen Beispiel an:

Beispiel: Eine einfache Meta-Algebra

Als D wählen wir den Wertebereich der ganzen Zahlen, $\mathbb{Z}$, und für O die Operatoren „+“ and „*“. Entsprechend wird L wie folgt definiert:

$$L := (\{+, *\}, \mathbb{Z}). \tag{6.9}$$

Daten und Operatoren können als Knoten in einem gerichteten Wald aufgefasst werden. Ein Spezialfall stellt der vollständige Anfragegraph dar, welcher als einzelner, gerichteter Baum das Endergebnis in der Wurzel enthält. Die Datenknoten enthalten die Rohdaten und sind die jeweiligen Blattknoten innerhalb der einzelnen Bäume. Diese werden über eine ausgehende Kante mit einem Operatorknoten verknüpft.

Definition: Operatorknoten

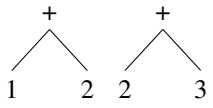
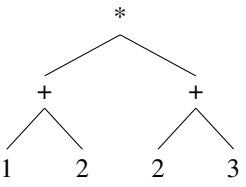
Ein Operatorknoten hat eine oder mehrere eingehende Kanten von Daten- bzw. weiteren Operatorknoten sowie eine ausgehende Kante, welche die getypte Ausgabe enthält. Operatorknoten können somit keine Blattknoten darstellen. Die eingehenden Kanten stellen die Argumente für den jeweiligen Operator dar. Diese besitzen wie die ausgehende Kante einen Typ.

Für den Operator müssen entsprechend passende Eingabetypen vorliegen. Beispielsweise erlaubt die Multiplikation nur die Verarbeitung von numerischen Werten und nicht die Verarbeitung von gesamten Datenbankrelationen.

Zur Vereinfachung können Datenknoten und einzelne Teilbäume im Graphen mehrfach vorkommen, sofern sie an verschiedenen Stellen der Berechnung benötigt werden. Dies erlaubt für die späteren Anfragetransformationen eine einfachere Handhabung sowie eine feingranularere Optimierung.

Beispiel: Schrittweise Verarbeitung auf Ebene der Meta-Algebra

Die nachfolgende Tabelle zeigt die schrittweise Verarbeitung der Berechnung $(1+2)*(2+3)$ als Wald- bzw. Baumstruktur. Die Verarbeitung erfolgt schrittweise von den Rohdaten L_0 über L_1 bis hin zum Endergebnis der Berechnung L_2. In der linken Spalte sind die Algebren für die einzelnen Schritte dargestellt, rechts die grafische Repräsentation des abstrakten Syntaxbaumes.

$L_0 := (\emptyset, \{1, 2, 3\})$	1 2 3
$L_1 := (\{+\}, \{(1+2), (2+3)\})$	
$L_2 := (\{+, *\}, \{(1+2)*(2+3)\})$	

In der Datenbankforschung ist diese Art der Darstellung für die Analyse und Optimierung von Anfragen nicht unbekannt. Der hier vorgestellte eigene Ansatz erweitert die Anfragebäume um beliebige Arten von Operatoren, insbesondere der Erweiterung relationaler Anfragen um Arithmetik, Aggregate und komplexe Analysefunktionen. Zusätzlich werden in die Anfragebäume Beschränkungen hinsichtlich der erlaubten Menge an Operatoren eingearbeitet. Dies dient als Vorbereitung zur Aufspaltung der Anfrage über mehrere Rechenknoten, welche über unterschiedliche Kapazitäten zur Beantwortung der Anfrage verfügen.

6.1.4 Transformation komplexer Aggregatfunktionen

In smarten Umgebungen werden häufig komplexe Analysen, die aus Hidden-Markov-Modellen, Regressionen und Korrelationen sowie Matrixoperationen bestehen, ausgeführt, um Aktivitäts- und Intentionserkennungen durchzuführen. Bei der Verwendung dieser komplexen Aggregatfunktionen wird es zunehmend schwerer, das Verhältnis zwischen zwei Anfragen zu bestimmen. Um den Suchraum für die Anfragetransformationen zu reduzieren, werden an dieser Stelle die Konzepte der Operatoräquivalenz bzw. des Enthaltenseins von Operatoren eingeführt:

Definition: Äquivalenz und Enthaltensein von Operatoren

Eine Menge von Operatoren O_1 enthält ($\sqsupseteq_O$) eine zweite Operatormenge O_2, wenn mindestens alle Operatoren aus O_2 durch O_1 gleichwertig darstellbar sind:

$$O_1 \sqsupseteq_O O_2 \tag{6.10}$$

Zwei Mengen von Operatoren sind äquivalent ($\equiv_O$), wenn sie sich gegenseitig enthalten:

$$O_1 \equiv_O O_2 \Leftrightarrow O_1 \sqsupseteq_O O_2 \wedge O_2 \sqsupseteq_O O_1 \tag{6.11}$$

Folgendes Beispiel illustriert das Verhältnis zwischen verschiedenen Operatoren:

Beispiel: Enthaltensein von Operatoren

Die Operatormenge, welche nur aus dem Operator Summe (`sum`) besteht, ist Teilmenge jeder Operatormenge, welche den Operator Addition (`add`) enthält:

$$\{sum\} \sqsubset_O \{add\}. \tag{6.12}$$

Während der `sum`-Operator nur die genaue Summe über die Menge von Werten eines Attributes A berechnen kann, ist der `add`-Operator in der Lage, beliebige Summen von je zwei Werten aus A zu berechnen. Durch die mehrfache Anwendung des `add`-Operators lässt sich somit auch die Summe berechnen. Ein weiteres, bekanntes Beispiel ist die Berechnung des Durchschnittes (`avg`), welche auch durch die Operatoren Summe (`sum`) , Anzahl (`count`) und Division (`div`) darstellbar ist:

$$\{avg\} \sqsubset_O \{sum, count, div\}. \tag{6.13}$$

Die gleichen Anforderungen gelten an die datenhaltenden Knoten:

Definition: Äquivalenz und Enthaltensein von Daten

Eine Menge von Daten D_1 ist in einer zweiten Datenmenge D_2 enthalten ($D_1 \sqsubseteq_D D_2$), wenn jeder Baum $t \in D_1$ entweder als Baum in D_2 vorhanden ist oder durch die Anwendung von Operatoren aus O_1 in einen solchen transformiert werden kann. Die Anfragetransformation kann durch verschiedene Techniken, wie die Assoziativ- und Distributivgesetze, durch weitere Optimierungstechniken aus relationalen Datenbanksystemen oder durch den im nächsten Abschnitt vorgestellten Mechanismus erfolgen. Zwei Datenmengen D_1 und D_2 sind äquivalent ($D_1 \equiv D_2$), wenn sie sich gegenseitig enthalten:

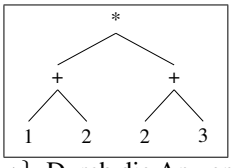
$$D_1 \equiv_D D_2 \Leftrightarrow D_1 \sqsubseteq_D D_2 \wedge D_2 \sqsubseteq_D D_1. \tag{6.14}$$

Folgendes Beispiel veranschaulicht das Verhältnis zwischen verschiedenen Datenmengen:

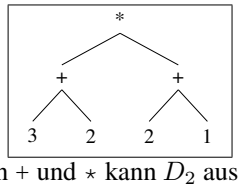
Beispiel: Enthaltensein und Äquivalenz von Daten

Betrachten wir folgende drei Mengen von Daten

- $D_1 := \{1, 2, 3\}$,
- $D_2 := \{(1+2)*(2+3)\}$, $\boxed{1, 2, 3}$ $\sqsupseteq_D$
- $D_3 := \{(3+2)*(2+1)\}$



$\equiv_D$



über der gemeinsamen Menge an Operatoren $O := \{+, *\}$. Durch die Anwendung von $+$ und $*$ kann D_2 aus D_1 erzeugt werden — neben vielen weiteren Berechnungen, wie z. B. $(2*2)+(3*3)$ oder $1+2+3$. Es gilt somit die Beziehung

$$D_2 \sqsubseteq_D D_1.$$

D_2 und D_3 sind hingegen äquivalent, da mittels Kommutativgesetz die Argumente miteinander vertauscht wurden:

$$D_2 \equiv_D D_3.$$

Folglich gilt auch, dass D_3 in D_1 enthalten ist:

$$D_3 \sqsubseteq_D D_1.$$

Die Zerlegung komplexer Funktionen und Aggregate in mehrere einfache Basisfunktionen bzw. -aggregate ergibt mehrere Vorteile: Primär müssen keine Beziehungen bzgl. des Enthaltenseins zwischen den einzelnen Funktionen bzw. Aggregaten aufgestellt werden – dies ist in vielen Fällen auch gar nicht möglich bzw. es ergeben sich überlappende Beziehungen, die keine Aussagekraft besitzen. Zudem wird durch die feingranulare Gliederung ermittelt, welche Teile der komplexen Funktionen und Aggregaten auf die gleichen Teilberechnungen und Datenknoten zurückgreifen. Dadurch werden potentielle Datenschutzverletzungen leichter erkannt, sofern ungewöhnliche bzw. unbekannte Anfragen oder Datenzugriffe auftreten.

Komplexere Beispiele zur Verwendung der Meta-Algebra und der damit verbundenen Verteilung von Anfragen sind in Anhang D aufgelistet: Abschnitt D.1 zeigt die Zerlegung der Aggregatfunktionen *Korrelation* und *Regression*. In Abschnitt D.2 wird näher auf die Verteilung relationaler Operatoren eingegangen.

Die nachfolgenden Abschnitte zeigen, wie auf Basis der definierten Kapazitäten (siehe Abschnitt 5.6) und einer vorgegebenen Anfrage Q die optimale Verteilung von Q auf mehrere Ebenen erfolgt. Abschnitt 6.2 geht dafür zunächst auf die Bestimmung des *Rewriting Supremums* für Selektionsbedingungen mit Aggregatfunktionen ein.

6.2 Selektionsbedingungen

Für die nachfolgenden Betrachtungen gehen wir von einer konjunktiven Aggregatanfrage Q aus, welche wie folgt aufgebaut ist:

Definition: Konjunktive Aggregatanfrage

Eine Anfrage Q ist eine konjunktive Aggregatanfrage der Form

$$Q(\alpha(X); Y) :- \bigwedge_i \bigvee_j (\neg) x_{ij}, \tag{6.15}$$

wobei α eine Aggregatfunktion über der Attributmenge X, gruppiert nach den Attributen Y, ist. Die einzelnen x_{ij} stellen (negierte) Selektionsprädikate auf einzelnen Tupeln oder Gruppen dar, welche ggf. aus mehreren `ODER`-verknüpften Teilprädikaten bestehen. Diese Prädikate können entweder einfache Attribut-Attribut- oder Attribut-Wert-Vergleiche sein bzw. Unteranfragen darstellen.

Die konjunktive Anfrage Q besteht aus mehreren Teilzielen. Diese werden wie folgt definiert:

Definition: Teilziele einer Anfrage

Wir bezeichnen jede Disjunktion der Prädikate aus Q als Teilziel g_i der Anfrage Q:

$$g_i = \bigvee_j (\neg) x_{ij}. \tag{6.16}$$

Zur Veranschaulichung bestimmen wir die Teilziele der laufenden Beispielanfrage:

Beispiel: Teilziele der Anfrage Q

Sei Q die Beispielanfrage, die zu Beginn dieses Kapitel als laufendes Beispiel eingeführt wurde. Q besteht aus den folgenden vier Teilzielen:

- $g_1 := tid \geq 42$
- $g_2 := length\, BETWEEN\, 200\, AND\, 500$
- $g_3 := MIN(createdate) < MAX(modifydate)$
- $g_4 := REGR\_SLOPE(filesize, length) > 25$

O. B. d. A. werden nachfolgend nur „einfache" Teilziele ohne `ODER`-Verknüpfungen betrachtet. Abweichend von den weiteren Anforderungen an die Anfragetransformation müssen für diese Arten von Teilzielen folgende Bedingungen gelten:

- Der Operator `ODER` muss durch die Anfragekapazitäten des aktuellen Knotens abgedeckt werden.
- Es müssen zudem *alle* Operatoren der Prädikate x_{ij} auf dem Knoten unterstützt werden.

Weiterhin betrachten wir nur eine zweistufige Anfragetransformation. Ansonsten müsste die Transformation solange für die Ebenen $L_3, \ldots L_n$ rekursiv ausgeführt werden, bis alle Teilziele ausführbar sind. Für das laufende Beispiel gehen wir von folgenden Anfragekapazitäten aus:

Beispiel: Anfragekapazitäten für das laufende Beispiel

Sei $L := \{L_1, L_2\}$ die Menge der zur Verfügung stehenden Knoten. L_1 beschränken wir auf folgende Operatormenge O_1:

$$O_1 := \{<, \leq, \geq, >, =, MIN, MAX\}. \tag{6.17}$$

Für L_2 nehmen wir an, dass alle Operatoren unterstützt werden.

6.2.1 Abbildung der Teilziele

Ausgehend von der konjunktiven Normalform von Q wird nun die Anfragetransformation r zur Ermittlung der transformierten Anfrage Q_1 bestimmt, welche den Anforderungen des AQuO-Problems genügt. Zur Bestimmung von r muss jedes Teilziel g_i aus Q auf ein äquivalentes Teilziel g_i' in Q_1 (bzw. einer Kombination aus mehreren Teilzielen) abgebildet werden. Ist dies nicht möglich, kann r auch derart formuliert werden, dass Teilziele enthalten sind, die eine Obermenge an Tupeln zurückliefern.

Ausgehend von einer Anfrage Q der Form

$$Q := g_1 \wedge g_2 \wedge \cdots \wedge g_x \cdots \wedge g_n \tag{6.18}$$

kann die Anfrage Q_1 so formuliert werden, dass sie folgende Struktur annimmt:

$$Q_1 := g_1 \wedge g_{2'} \wedge \cdots \wedge \top \cdots \wedge g_m, \tag{6.19}$$

wobei m und n die Anzahl der Teilziele in Q respektive Q_1 sind. Das Symbol $\top$ steht dabei für ein Teilziel, welches alle Tupel zurückgibt. Abbildung 6.4 zeigt eine mögliche Überführung der einzelnen Teilziele. Für jedes Teilziel tritt einer der folgenden Fälle ein:

Fall 1: Das Teilziel wird in der transformierten Anfrage vollständig unterstützt (Teilziel g_1).

Fall 2: Das transformierte Teilziel erzeugt eine Obermenge dessen, was durch das Teilziel der Originalanfrage zurückgegeben werden würde (Teilziel g_2 bzw. g_2').

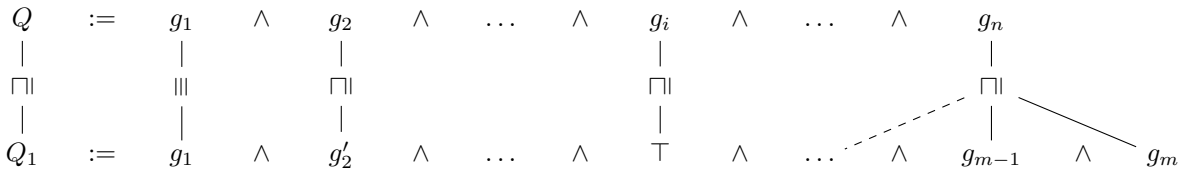


Abbildung 6.4: Bei der Anfragetransformation von Q nach Q_1 werden die einzelnen Teilziele g_i von Q auf Teilziele in Q_1 abgebildet.

Fall 3: Ein einzelnes Teilziel wird in mehrere Teilziele aufgeteilt, wenngleich diese weiterhin eine Obermenge zurückliefern (Teilziel g_n).

Fall 4: Ein Teilziel wird weder unterstützt noch kann es durch andere Operatoren (partiell) abgebildet werden (Teilziel g_i).

Im Folgenden bezeichnet F eine Klasse von Funktionen mit

$$F : G \subseteq Q \rightarrow G' \subseteq Q_1, \tag{6.20}$$

welche ein oder mehrere Teilziele G aus der Originalanfrage Q auf die Menge der Teilzielen G' der transformierten Anfrage Q_1 abbildet. Entsprechend der oben aufgeführten Fälle unterscheiden wir die folgenden Arten von Abbildungen einzelner Teilziele:

Definition: Äquivalente und operator-erhaltende Abbildungen

Wir bezeichnen mit $g_i \equiv f(g_i)$ eine *äquivalente Abbildung* des Teilziels g_i auf dem Datenbankschema D:

$$g_i \equiv f(g_i) \Leftrightarrow \forall d(D) : g_i(d) \equiv f(g_i)(d). \tag{6.21}$$

Gilt zudem folgende Bedingung:

$$ops(g_i) \equiv ops(f(g_i)), \tag{6.22}$$

d. h., in g_i und $f(g_i)$ wird die gleiche Menge an Operatoren (ops) verwendet, so bezeichnen wir die Abbildung als *operator-erhaltend*.

Definition: Partielle Abbildung

Wird durch eine Abbildung der Form

$$g_i \sqsubset f(g_i) \Leftrightarrow \forall d(D) : g_i(d) \subseteq f(g_i)(d) \tag{6.23}$$

das Teilziel g_i nicht vollständig umgesetzt, bezeichnen wir dies als *partielle Abbildung* – es werden durch die transformierte Anfrage mehr Daten zurückgeliefert als durch die ursprüngliche Anfrage.

Definition: Fragmentierte Abbildung

Durch die Gleichung

$$g_i \sqsubseteq f(g_i) = (g_a \wedge g_b \wedge \cdots \wedge g_x) \tag{6.24}$$

wird eine *fragmentierte Abbildung* des Teilziels g_i beschrieben. Das Teilziel g_i wird dabei in x Teilziele mit verschiedenen Operatoren aufgespalten. Die Abbildung kann vollständig oder partiell sein.

Definition: Fehlende Abbildung

Als *fehlende Abbildung* der Form

$$g_i \sqsubseteq f(g_i) = \top \Leftrightarrow \forall d(D) : f(g_i)(d) \equiv d \tag{6.25}$$

bezeichnen wir eine nicht erfolgte Abbildung des Teilziels g_i. Diese enthält mindestens einen Operator, welcher auf der aktuellen Ebene nicht unterstützt wird bzw. nicht vollständig oder partiell durch bekannte Regeln transformiert werden kann.

Die Anzahl der Teilziele kann sich bei der Transformation von Q nach Q_1 verändern, wenn einer der beiden folgenden Fälle auftritt:

1. Durch eine vollständige oder partielle, fragmentierte Abbildung mindestens eines Teilzieles entstehen zwei oder mehrere neue Teilziele.
2. Mindestens ein transformiertes Teilziel g_i' in Q' deckt mehrere Teilziele aus Q ab. Dies ist insbesondere dann der Fall, wenn mehrere Teilziele nicht unterstützt werden und somit auf $\top$ abgebildet wurden.

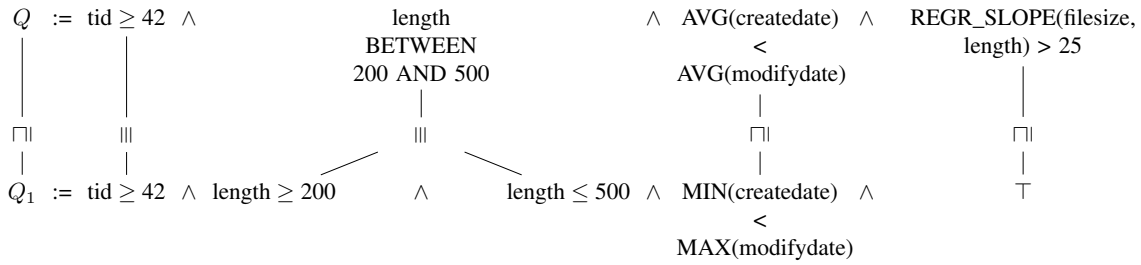
Im Folgenden werden die verschiedenen Arten von Abbildungen anhand des laufenden Beispiels verdeutlicht:

Beispiel: Arten von Abbildungen für die Anfrage Q

Das Teilziel g_1 kann direkt auf L_1 umgesetzt werden, da der Operator $\geq$ in der Anfragekapazität O_1 enthalten ist. Für g_2 existiert eine äquivalente Transformation, bei der das `BETWEEN`-Prädikat durch zwei Prädikate (mit $\leq$ und $\geq$) ersetzt wird, welche eine Teilmenge der Anfragekapazität O_1 darstellen.

Komplizierter wird es bei dem Teilziel g_3: Da die beiden `AVG`-Operatoren nicht durch die Anfragekapazitäten O_1 unterstützt werden, ist die Ausführung des Vergleiches nicht direkt möglich. Anstatt g_3 auf die Ebene L_2 zu verschieben, kann auf L_1 eine Vorselektion ausgeführt werden, indem eine partielle Abbildung der `AVG`-Operatoren auf einen `MIN`- und einen `MAX`-Operator auf Basis der Techniken von Can Türker (siehe Abschnitt 5.3) erfolgt.

Das Teilziel g_4 kann nicht auf der Ebene L_1 ausgeführt werden, da für die komplexe Aggregatfunktion `REGR_SLOPE` keine Transformationsregel bekannt ist sowie durch die zur Verfügung stehenden Operatoren in O_1 keine Teilaggregate vorberechnet werden können. Die folgende Abbildung verdeutlicht das Verhältnis zwischen den originalen und transformierten Teilzielen:



Zusammengefasst entstehen fünf Selektionsbedingungen als Teilziele der partiellen Anfrage Q_1, welche auf der Ebene L_1 vollständig ausführbar sind:

- $\sigma_a := tid \geq 42$
- $\sigma_b := length \geq 200$
- $\sigma_c := length \leq 500$
- $\sigma_d := AVG(createdate) < AVG(modifydate)$
- $\sigma_e := \top$

Ausgehend von den bisherigen Definitionen für das Answering-Queries-using-Operators-Problem und das Rewriting Supremum (siehe Formeln 6.2 bzw. 6.3) lassen sich diese verfeinern, indem die Containment-Eigenschaften auf die Teilziele der Anfragen verlagert werden:

Definition: Answering Queries using Operators (detailliertere Definition)

Gegeben sei ein Datenbankschema D, eine Anfrage Q und eine Menge von Ebenen L, wobei jede Ebene $L_i \in L$ die Menge an Operatoren O_i unterstützt. Durch AQuO wird eine Transformation r mit $r(Q) = Q_i$ gesucht, sodass

$$\begin{aligned} Q_i(D) \sqsupseteq Q(D) \Leftrightarrow \\ \forall d(D) : \forall \sigma_j \in Q : \exists f \in F : f(\sigma_j)(d) \supseteq \sigma_j(d) \end{aligned} \tag{6.26}$$

gilt und jede Selektionsbedingung $f(\sigma_j) \in Q_i$ nur Operatoren aus O_i verwendet.

Für jede Instanz d der Datenbank D und jedes Teilziel σ_j in der Originalanfrage Q existiert eine Abbildung f aus der Menge der möglichen Abbildungen FF , sodass die Auswertung von $f(\sigma_j)(d)$ mehr Ergebnistupel zurückliefert als $\sigma_j(d)$. Im schlimmsten Fall – wenn keine passende Transformationsregel existiert – werden durch das Teilziel alle Tupel zurückgegeben. Ausgehend von der Anforderung an die Abbildung einzelner Teilziele lässt sich das *Rewriting Supremum* detaillierter formulieren:

Definition: Rewriting Supremum (detaillierte Definition)

Q_1 ist ein *Rewriting Supremum* der Anfrage Q mit der Menge an Abbildungsfunktionen F, wenn für jedes Teilziel σ_i von Q keine Abbildung $f' \in F$ existiert, welches in Bezug auf $f \in F$ eine echte Teilmenge der Tupel zurückliefert, jedoch weiterhin eine Obermenge der Tupel zurückgibt, welche die Originalauswertung von σ_i liefert:

$$\begin{aligned} \nexists Q' : Q_1(D) \sqsupset Q'(D) \sqsupseteq Q(D) \Leftrightarrow \\ \forall d(D) : \forall \sigma_i \in Q : \nexists f' : f(\sigma_i)(d) \supset f'(\sigma_i)(d) \supseteq \sigma_i(d). \end{aligned} \tag{6.27}$$

Die Abbildungen der Selektionsprädikate stellt nur eine Anforderung an die Anfragetransformation dar. Insbesondere bei der Transformation von Selektionsbedingungen auf Gruppen ist darauf zu achten, dass die verwendeten Attributwerte weiterhin zur Verfügung stehen müssen. Auf die notwendigen Attribute in der Projektionsliste zur Beantwortung der verbleibenden Anfrage Q_δ wird in Abschnitt 6.3 näher eingegangen.

6.2.2 Verteilung der Anfrage

Auf Basis des bisherigen Transformationsprozesses wurde die gegebene Anfrage Q in eine partielle Anfrage Q_1 überführt, welche auf der untersten Ebene L_1 ausführbar ist. Zur vollständigen Ausführung der Anfrage wird weiterhin eine Restanfrage Q_δ benötigt, welche auf dem Zwischenergebnis $Q_1(D)$ die verbleibenden Selektionsbedingungen und Aggregationen realisiert. Die Anfrage Q_δ ist dabei so zu wählen, dass die Hintereinanderausführung von Q_1 und Q_δ äquivalent zur Originalanfrage Q ist:

A1 **Rekonstruierbarkeit:** $\forall d(D) : Q(d) \equiv Q_\delta(Q_1(d))$.

Die Menge der Selektionsprädikate Σ_Q der Originalanfrage Q kann als `UND`-Verknüpfung von drei Mengen ihrer Teilziele Σ aufgefasst werden:

$$\Sigma_Q := \Sigma_X \cup \Sigma_Y \cup \Sigma_Z, \tag{6.28}$$

wobei die Teilmengen wie folgt gebildet werden:

- Σ_X := Menge der durch F äquivalent abbildbaren Teilziele,
- Σ_Y := Menge der Teilziele, die durch Anwendung von F eine Obermenge erzeugen und
- Σ_Z := Menge der durch F nicht abbildbaren Teilziele.

Entsprechend müssen die folgenden Bedingungen gemäß den in Abschnitt 6.1.2 gestellten Anforderungen an das Rewriting Supremum für Σ_X, Σ_Y und Σ_Z gelten:

A2 **Vollständigkeit:** $\Sigma_X \cup \Sigma_Y \cup \Sigma_Z \equiv \Sigma_Q$
A3 **Disjunktheit:** $\Sigma_X \cap \Sigma_Y \equiv \Sigma_X \cap \Sigma_Z \equiv \Sigma_Y \cap \Sigma_Z \equiv \emptyset$

Beispiel: Klassen von Teilzielen im laufenden Beispiel

Das vorherige Beispiel auf Seite 145 erzeugte aus der Originalanfrage Q ein partielles Anfragefragment Q_1. Ausgehend von den angewendeten Abbildungsfunktionen $f \in F$, lassen sich die Teilziele von Q in die drei Klassen einordnen:

- $\Sigma_X := \{tid \geq 42, length \geq 200, length \leq 500\}$
- $\Sigma_Y := \{AVG(createdate) < AVG(modifydate)\}$
- $\Sigma_Z := \{REGR\_SLOPE(filesize, length) > 25\}$

Durch die Bestimmung von Σ_X, Σ_Y und Σ_Z lässt sich im Anschluss die Menge der Selektionsprädikate Σ_δ von der Restanfrage Q_δ wie folgt konstruieren:

$$\Sigma_\delta := \Sigma_Y \cup \Sigma_Z \tag{6.29}$$

Entsprechend dieser Formel besteht Q_δ aus allen Teilzielen, die durch F entweder nur teilweise oder überhaupt nicht in Q_1 ausgeführt wurden. Diejenigen Teilziele, die bereits vollständig in Q_1 auf der Ebene L_1 ausgeführt wurden, brauchen nicht erneut durch Q_δ auf einer höheren Ebene ausgeführt werden[1].

Beispiel: Konstruktion der δ-Anfrage

Durch die Kombination von Σ_Y und Σ_Z wird Q_δ wie folgt konstruiert:

$$\begin{aligned} Q_\delta := {} & AVG(createdate) < AVG(modifydate) \\ & \wedge REGR\_SLOPE(filesize, length) > 25 \end{aligned} \tag{6.30}$$

Die Idee zur Hintereinanderausführung mehrerer Anfrageprädikate bzw. weiterer relationaler Operatoren ist nicht neu – es ist ein Kernkonzept vieler Anfrageoptimierer in relationalen Datenbanken. Einige grundlegende Optimierungstechniken wurden in Abschnitt 5.8 bereits vorgestellt.

Im Gegensatz zu relationalen Datenbanken werden die Regeln in AQuO aber nicht zur Reduzierung der Beantwortungszeit von Anfragen eingesetzt. Durch die Erweiterung und Modifikation dieser Regeln wird vielmehr der Aspekt der Datensparsamkeit erzwungen. Durch die Verlagerung einzelner unterstützter Operatoren auf lokale Rechenknoten (bzw. zu den Basisrelationen) werden weniger Daten an den Vaterknoten weitergeleitet. Der Vaterknoten übernimmt ausschließlich die Operatoren, welche nicht durch die Kindknoten ausgeführt werden konnten.

Der Algorithmus 6 fasst die Transformation der Anfrage Q nach Q_1 und Q_δ zusammen. Im nachfolgenden Abschnitt wird die Korrektheit der Anfragetransformation bzw. des Rewriting Supremums bewiesen. Dies garantiert, dass die verbleibende Anforderung A1 an das Rewriting Supremum, die Rekonstruierbarkeit der Anfrage, umgesetzt wird.

6.2.3 Beweis für die Äquivalenz von Originalanfrage und transformierter Anfrage

In diesem Abschnitt wird die Richtigkeit der äquivalenten Anfragetransformation mittels AQuO bewiesen. Die Beweise für die Transformation einzelner Teilziele beruhen auf den Forschungsergebnissen der in Kapitel 5 vorgestellten wissenschaftlichen Arbeiten und werden an dieser Stelle nicht näher ausgeführt.

[1] Eine erneute Ausführung hat keinen Einfluss auf die Ergebnisrelation, da das Teilziel bereits durch Ausführung von Q_1 erfüllt wurde.

Algorithmus 6: Anfragetransformation von Q nach Q_i und Q_δ

Data: Eine Anfrage Q, die aktuelle Ebenen L_i sowie deren Menge von Abbildungen F_i
Result: Ein Tupel (Q_i, Q_δ) mit der partiell ausgeführte Anfrage Q_i und deren Restanfrage Q_δ
foreach $\sigma \in \Sigma_Q$ **do**
  if $\forall o \in \sigma : o \in O_i$ **then**
    $\Sigma_{Q_i} := \Sigma_{Q_i} \cup \sigma$;
  else
    if $\exists f \in F_i : \forall o' \in f(\sigma) : o' \in O_i \wedge \sigma(D) \equiv f(\sigma)(D)$ **then**
      $\Sigma_{Q_i} := \Sigma_{Q_i} \cup f(\sigma)$;
    else
      if $\exists f \in F_i : \forall o' \in f(\sigma) : o' \in O_i \wedge \sigma(D) \sqsubseteq f(\sigma)(D) \not\equiv \top$ **then**
        $\Sigma_Y := \Sigma_Y \cup \sigma$;
        $\Sigma_{Q_i} := \Sigma_{Q_i} \cup f(\sigma)$;
      else
        $\Sigma_Z := \Sigma_Z \cup \sigma$;
      end
    end
  end
end
$Q_i(D) := \bigwedge \Sigma_{Q_i}(D)$;
$Q_\delta(D) := \bigwedge \Sigma_Y \wedge \bigwedge \Sigma_Z(D)$;
return (Q_i, Q_δ);

Nach der Transformation der Originalanfrage Q wurde eine Anfragekette QC (*query chain*; siehe Gleichung 6.1) erzeugt. Im Folgenden wird gezeigt, dass Q äquivalent zu QC ist. Ohne Beschränkung der Allgemeinheit lässt sich der Beweis auf eine Anfragekette der Länge 2, bestehend aus einer Anfrage Q_1 auf den Basisdaten und einer Restanfrage Q_δ (siehe Gleichung 6.29), anwenden:

$$Q(D) \equiv Q_\delta(Q_1(D)) \tag{6.31}$$

Der Beweis für längere Anfrageketten der Länge n mit $n > 2$ lässt sich durch logische Induktion aus den jeweils kürzeren Ketten herleiten. Dies wird am Ende dieses Abschnittes näher erläutert.

Beispiel: Anfragekette $Q_\delta(Q_1(D))$

Durch die Hintereinanderausführung von Q_1 (siehe Beispiel) gefolgt von Q_δ (siehe Gleichung 6.30) ergibt sich folgende Anfragekette:

$$\begin{aligned} Q_\delta(Q_1(D)) := {} & AVG(createdate) < AVG(modifydate) \\ & \wedge REGR\_SLOPE(filesize, length) > 25 \; (\\ & \quad tid \geq 42 \\ & \quad \wedge length \geq 200 \\ & \quad \wedge length \leq 500 \\ & \quad \wedge MIN(createdate) < MAX(modifydate) \\ & \quad \wedge \top \\ &) \end{aligned} \tag{6.32}$$

Beweis. Im Folgenden wird bewiesen, dass die Anfragekette $Q_\delta(Q_1(D))$ äquivalent zur Originalanfrage Q ist. Zu zeigen ist, dass durch die Äquivalenzumformungen von $Q \Rightarrow Q_\delta(Q_1(D))$ sowie $Q_\delta(Q_1(D)) \Rightarrow Q$ beide Anfragen aufeinander abgebildet werden können.

Hinrichtung "⇒": Der Beweis für die Hinrichtung ergibt sich direkt aus der Konstruktion von der partiellen Anfrage Q_1 und der Restanfrage Q_δ aus der Originalanfrage Q, wie sie in Abschnitt 6.2 beschrieben wurde.

Rückrichtung "⇐": Aus den Grundlagen der logischen Optimierung von Datenbankanfragen (siehe Abschnitt 5.15) ist bekannt, dass das logische `UND` kommutativ für zwei Mengen von Selektionsprädikaten $P_1 := \bigwedge \Sigma_1$ und $P_\delta := \bigwedge \Sigma_\delta$ ist:

$$\sigma_{P_\delta}(\sigma_{P_1}(D)) \Leftrightarrow \sigma_{P_\delta \wedge P_1}(D) \Leftrightarrow \sigma_{P_1}(\sigma_{P_\delta}(D)) \tag{6.33}$$

Wir setzen nun für P_1 die Prädikate aus Q_1 bzw. für P_δ die Prädikate aus Q_δ ein. Da durch die Konstruktion von Q_1 und Q_δ

$$\forall \sigma_x \in P_\delta : \exists \sigma_{x'} \in P_1, \text{mit } \sigma_x(D) \subseteq \sigma_{x'}(D), \tag{6.34}$$

gilt, gibt $\sigma_{x'}$ im Verhältnis zu σ_x eine Obermenge der Tupel aus D zurück. Dadurch kann jedes σ_x aus P_1 im $(P_\delta \wedge P_1)$-Selektionsprädikat entfernt werden, wodurch nur die Prädikate aus P_δ sowie die nicht transformierbaren Selektionsbedingungen aus P_1 in der Anfrage Q_1 verbleiben. Für die transformierten Teilziele werden die äquivalenten, operator-erhaltenden Teilziele aus der Konstruktions-Phase anstatt der entsprechenden Teilziele aus Σ_X eingesetzt. Die auf diese Weise erzeugte Anfrage enthält die gleiche Menge an Prädikaten wie Q, wodurch die Annahme der Behauptung aus Gleichung 6.31 gilt. □

Beispiel: Herleitung der Äquivalenz von $Q(D)$ und $Q_\delta(Q_1(D))$

Nach der Anwendung von Gleichung 6.33 erhalten wir die folgende Anfrage Q':

$$\begin{aligned} Q'(D) := & AVG(createdate) < AVG(modifydate) \\ & \wedge REGR\_SLOPE(filesize, length) > 25 \\ & \wedge tid \geq 42 \\ & \wedge length \geq 200 \\ & \wedge length \leq 500 \\ & \wedge MIN(createdate) < MAX(modifydate) \\ & \wedge \top \end{aligned} \tag{6.35}$$

Anschließend werden nach Gleichung 6.34 diejenigen Selektionsbedingungen entfernt, die durch andere Bedingungen abgedeckt sind. Dies betrifft folgende Selektionen:

1. $length\, BETWEEN\, 200\, AND\, 500 \equiv length \geq 200 \wedge length \leq 500$
2. $AVG(createdate) < AVG(modfiydate) \sqsubseteq MIN(createdate) < MAX(modifydate)$
3. $REGR\_SLOPE(filesize, length) > 25 \sqsubseteq \top$

Die Selektionsbedingungen, welche auf der rechten Seite der obigen Abbildungen stehen, können entsprechend aus dem Prädikat $P_\delta \wedge P_1$ entfernt werden. Dadurch wird die Anfrage Q' wie folgt reduziert:

$$\begin{aligned} Q' := & tid \geq 42 \\ & \wedge length\, BETWEEN\, 200\, AND\, 500 \\ & \wedge AVG(createdate) < AVG(modifydate) \\ & \wedge REGR\_SLOPE(filesize, length) > 25. \end{aligned} \tag{6.36}$$

Diese Anfrage ist äquivalent zu den Teilzielen der laufenden Beispielanfrage Q.

Anfrageketten der Länge n > 2

Muss die entstehende Restanfrage Q_δ aufgrund fehlender Anfragekapazitäten auf der Ebene L_2 erneut transformiert werden, so entsteht nach mehrmaliger Anfragetransformation eine Anfragekette, wie sie in Gleichung 6.1 dargestellt ist. Die Äquivalenz von der Originalanfrage Q zur erzeugten Anfragekette beweisen wir über das Prinzip der *vollständigen Induktion*.

Beweis. Dafür müssen wir zeigen, dass für alle Anfrageketten der Länge n mit $n \in \mathbb{N}$ und $n \geq 2$ die Äquivalenz der Anfragen gilt.

Induktionsanfang: Für den Induktionsanfang muss gezeigt werden, dass für $n = 2$ die Gleichung 6.1 gilt. Dies haben wir zu Beginn dieses Unterabschnittes gezeigt.

Induktionsschritt: Sei $m \in \mathbb{N}$ und die Gleichung 6.1 für $n = m$ bereits bewiesen. Wir zeigen nun, dass die Äquivalenz auch für $n = m + 1$ gilt:

$$Q(D) \equiv Q_{\delta'}(Q_{m+1}(Q_m(\ldots Q_1(D)))) \equiv Q_\delta(Q_m(\ldots Q_1(D))). \tag{6.37}$$

Durch die in Abschnitt 6.2 beschriebene Transformation werden die verbleibenden Teilziele aus δ auf die Anfragefragmente Q_{m+1} sowie $Q_{\delta'}$ aufgeteilt. Wird die Teilanfragekette $Q_m(\ldots Q_1(D))$ zu dem Zwischenergebnis D' ausgewertet, erhalten wir

$$Q(D) \equiv Q_{\delta'}(Q_{m+1}(D')) \equiv Q_\delta(D'). \tag{6.38}$$

Diese verkürzte Anfragekette stellt wieder eine Transformation mit einem einzelnen Schritt dar (siehe Gleichung 6.31). Der Beweis für die Äquivalenz dieser Kette ist dem Anfang dieses Unterabschnittes zu entnehmen. □

6.2.4 Keine Unterstützung des logischen UNDs

Durch die Beschränkung der Anfragekapazitäten, insbesondere in Sensornetzwerken, kann es durchaus vorkommen, dass die logische `UND`-Verknüpfung von mehreren Teilzielen nicht unterstützt wird. Dadurch lässt sich auf dem betroffenen Knoten nur ein einzelnes Teilziel ausführen, wodurch keine komplexeren Auswertungen auf dieser Ebene möglich sind.

Um trotzdem eine möglichst datensparsame Verarbeitung auf dem Knoten zu realisieren, muss ein einzelnes Teilziel ausgewählt werden, welches dafür sorgt, dass möglichst viele Tupel herausgefiltert bzw. aggregiert werden. Zur Entscheidung, welches Teilziel ausgewählt wird, müssen zunächst die Teilziele σ_i der Anfrage Q nach ihren Selektivitäten, sofern verfügbar, in aufsteigender Reihenfolge sortiert werden. Falls für einzelne Teilziele keine Selektivitäten verfügbar sind, so werden diese an das Ende der Liste gehängt. Mit $sel(\sigma_i)$ wird im Folgenden die erwartete Tupelanzahl geteilt durch die Anzahl an Tupeln der gesamten Relation bezeichnet.

Beispiel: Selektivitäten der Teilziele von Q

Für die Anfrage Q aus dem laufenden Beispiel nehmen wir folgende Selektivitäten der einzelnen, bereits sortierten Teilziele an:

1. $sel(REGR\_SLOPE(filesize, length) > 25) = 0,1$
2. $sel(length\, BETWEEN\, 200\, AND\, 500) = 0,5$
3. $sel(AVG(createdate) < AVG(modifydate)) = 0,8$
4. $sel(tid \geq 42) = 0,9$

Nach erfolgter Sortierung der Teilziele nach deren Selektivitäten, wird dasjenige Teilziel ausgewählt, welches die höchste Selektivität besitzt, dabei aber auch durch die Operatoren aus L unterstützt wird (siehe Algorithmus 7).

Algorithmus 7: Ausführung von einzelnen Teilzielen

Data: Eine geordnete Anfrage in KNF Q_{KNF^o}, eine Datenbank *D*, eine Menge von n Ebenen *L*
Result: Eine partiell ausgeführte Anfrage Q'_{KNF^o}
for *x:=1;* L_x *unterstützt nicht alle Operatoren* $\wedge$ *x<n; x++* **do**
  $G_{high} := findHighSelectivitySubgoal(Q_{KNF^o})$;
  if $\exists G_{high}$ **then**
    $D_{x+1} := G_{high}(D_x)$;
    Entferne G_{high} aus Q_{KNF^o}
  end
end

Beispiel: Selektivitäten der Teilziele von Q

Im laufenden Beispiel ist dies das Teilziel $x < 5$ mit einer Selektivität von `0,5`. Die Regressionsanalyse, welche zwar die höchste Selektivität aufweist, wird nicht ausgewählt, da der Operator nicht auf der Ebene L_1 unterstützt wird. Eine äquivalente oder Obermenge-liefernde Anfragetransformation ist für die Regressionsanalyse nicht bekannt.

Der `BETWEEN`-Operator kann ebenfalls nicht ausgeführt werden, da einerseits der Operator nicht unterstützt wird, andererseits die äquivalente Anfrage $length \geq 200 \wedge length \leq 500$ den nicht unterstützten `UND`-Operator enthält. Für die einzelnen Bedingungen $length \geq 200$ und $length \leq 500$ nehmen wir an, dass diese jeweils eine niedrigere Selektivität als $tid \geq 42$ besitzen.

Der Vergleichsoperator auf den Durchschnittswerten von x und y kann ebenfalls nicht ausgeführt werden, da `AVG` nicht zu den unterstützen Operatoren zählt. Für die mögliche (und unterstützte) Anfragetransformation des Teilzieles nach $MIN(createdate) < MAX(modifydate)$ nehmen wir ebenfalls eine niedrigere Selektivität an. Entsprechend wird die Selektionsbedingung $tid \geq 42$ zur alleinigen Ausführung auf L_1 ausgewählt.

Die verbleibenden, ggf. transformierten Selektionsbedingungen müssen entsprechend auf den nachfolgenden Ebenen ausgeführt werden. Mit den bis hier vorgestellten Techniken lässt sich die Selektion als relationaler Operator auf mehrere Rechnerknoten verteilen. Im nachfolgenden Abschnitt wird speziell auf die Verteilung der Projektion und Aggregation eingegangen. Für arithmetische Rechenoperatoren sowie weitere Operatoren der Relationenalgebra sei auf die Anhänge C.1 und C.3 verwiesen.

6.3 Verbleibende Attribute in der Projektion

Neben der Selektion gehören Verbund, Projektion sowie Aggregation und Gruppierung zu den wichtigsten relationalen Operatoren. In diesem Abschnitt wird untersucht, welche (aggregierten und ggf. umbenannten) Attribute für die verteilte Berechnung der Anfrage auf den jeweiligen Ebenen benötigt werden. Auf Basis dieser Ergebnisse werden Projektionslisten erstellt, welche in den Transformationsprozess der Anfrage einfließen. Ausgehend von dem vorgestellten Konzept zur Anfragetransformation kann auch festgelegt werden, welche Attribute in der Anfragekette bei jedem Schritt weitergegeben werden müssen, um die verbleibende Anfrage ohne Informationsverlust zu beantworten.

Definitionen: Als notwendige Definitionen müssen zunächst zwei Funktionen, *var* und *attr*, eingeführt werden:

- $attr(Q_i)$ gibt den Kopf der Anfrage Q_i zurück. Dies sind diejenigen Attribute, die durch Projektion, Aggregation und Umbenennung nach Ausführung der Anfrage erhalten bleiben.
- $var(\Sigma_x)$ ermittelt alle Variablen, die innerhalb eines Teilzieles Σ_x einer Anfrage verwendet werden bzw. im Kopf einer Teilanfrage auftauchen.

6.3.1 Erstellung der Projektionslisten

Die notwendigen Attribute, die durch die äußere Restanfrage Q_δ zurückgegeben werden, müssen die gleichen Attribute sein, die durch die Originalanfrage Q zurückgegeben werden. Daher lässt sich der Kopf von Q_δ direkt über dem Kopf von Q definieren:

$$attr(Q_\delta) \equiv attr(Q) \tag{6.39}$$

Der Kopf der ersten Teilanfrage Q_1 muss entsprechend aus mindestens allen Attributen des Kopfes von Q_δ bestehen. Dies schließt umbenannte Attribute und aggregierte Attributwerte mit ein. Zusätzlich müssen alle Attribute, die weiterhin in Teilzielen von Q_δ verwendet werden, in Q_1 vorhanden sein:

$$attr(Q_1) := \ attr(Q_\delta) \bigcup var(\Sigma_x) \text{ mit } \Sigma_x \in Q_\delta \tag{6.40}$$

Wird die Originalanfrage Q über mehrere Ebenen verteilt, so ergibt sich $attr(Q_{i-1})$ (mit L_{i-1} als untere Ebene) analog aus den in Q_i vorkommenden Attributen der Selektionsbedingungen:

$$attr(Q_{i-1}) := \ attr(Q_i) \bigcup var(\Sigma_x) \text{ mit } \Sigma_x \in Q_i \tag{6.41}$$

Eine Ausnahme von den Gleichungen bilden Anfragen mit Aggregation und Umbenennungen. Diese werden im nachfolgenden Unterabschnitt erläutert.

6.3.2 Aggregation und Umbenennung

Kommen in einer der äußeren Anfragen Teilziele vor, welche auf unaggregierte Werte zugreifen, so kann die Aggregatfunktion aus der inneren Teilanfrage entfernt werden, sofern die Aggregation noch von einer der äußeren Anfragen aufgrund fehlender Kapazitäten realisiert werden muss. Eine ausführliche formale Beschreibung der Transformation von Aggregatanfragen wird zusammen mit vielen Beispielen im Abschnitt 6.4.3 und im Anhang C.2 angeführt. Das folgende Beispiel zeigt, welche Attribute in der laufenden Beispielanfrage durch die Teilanfrage Q_1 weitergegeben werden müssen:

Beispiel: Projektionslisten anhand des laufenden Beispiels

Durch die Originalanfrage Q werden die Attribute gid (ID des Genres) und die darauf gruppierten Gesamtlauflängen, $sum(length)$, zurückgegeben. Nach Gleichung 6.39 entspricht dies somit auch dem Kopf von Q_δ:

$$attr(Q_\delta) \equiv attr(Q) := \{gid, sum(length)\} \tag{6.42}$$

Die verbleibenden Teilziele in Q_δ verwenden weiterhin die Attribute $createdate$, $modifydate$ sowie $filesize$ und $length$. Da die Summe über $length$ erst in der Q_δ gebildet werden kann, wird $sum(length)$ aus der Projektionsliste von Q_1 gestrichen und durch $length$ ersetzt:

$$\begin{aligned} attr(Q_1) :=& \ attr(Q_\delta) \\ & \cup attr(\sigma_{REGR\_SLOPE(filesize,length)>25}) \\ & \cup attr(\sigma_{AVG(createdate)<AVG(modifydate)}) \\ & - \{sum(length)\} \\ :=& \ \{gid, sum(length)\} \\ & \cup \{filesize, length\} \\ & \cup \{createdate, modifydate\} \\ & - \{sum(length)\} \\ :=& \ \{gid, filesize, length, createdate, modifydate\}. \end{aligned} \tag{6.43}$$

> Das Attribut tid als Primärschlüssel ist für die äußere Anfrage uninteressant, da es nicht für die weitere Beantwortung der Anfrage benötigt wird und intern eine Multimengensemantik zum Einsatz kommt. Würde hingegen eine Mengensemantik verwendet, müsste der Primärschlüssel auch in der äußeren Anfrage vorkommen, da ansonsten Duplikate fälschlicherweise entfernt werden würden.

Bei der Umbenennung ist zusätzlich darauf zu achten, dass die umbenannten Attribute erst nach der Ausführung des Operators für die Projektionslisten genutzt werden. Bis dahin müssen entsprechend die bisherigen Attributnamen verwendet werden.

Ausgehend von den durch $attr$ definierten Attributmengen lassen sich initiale Projektionslisten für jede erzeugte Teilanfrage erstellen. Für die kapazitätsbasierte Anfragetransformation stellen diese Projektionslisten die minimal notwendigen Attribute zur Beantwortung der Originalanfrage Q dar.

6.3.3 Einschränkung der Projektion

Durch Quasi-Identifikatoren werden die Projektionsbedingungen von Anfragen verschärft, d. h., es kann vorkommen, dass weniger Attribute in der Projektionsliste auftauchen bzw. deren Attributwerte generalisiert wurden. Dies ist der Fall, wenn die Projektion mindestens einen Quasi-Identifikator vollständig enthält. Die Projektionsliste X wird reduziert, indem mindestens ein Attribut A_{QI} jedes Quasi-Identifikators QI entfernt wird. Im Falle einer Generalisierung ist darauf zu achten, dass die Anfrage ebenfalls transformiert werden muss. Dies ist der Fall, wenn das zu generalisierende Attribut Teil der verbleibenden Selektionsbedingungen ist.

Für eine minimale Teilmenge der zu entfernenden Attribute sei auf Abschnitt 4.6 im Kapitel zu den Quasi-Identifikatoren verwiesen. Die Möglichkeiten zum Vertauschen von relationalen Operatoren mit Anonymisierungstechniken werden in der Bachelorarbeit von Eric Klein [Kle20] näher skizziert.

Der folgende Abschnitt 6.4 stellt ein formales Regelwerk, *Query Rewriting by Contract*, vor, mit welchem die Anfragetransformation für konkrete Klassen von Anfragen realisiert wird. Als Basis dienen die in Kapitel 5 eingeführten Query-Containment-Techniken, welche mit den bis hierher vorgestellten Konzepten der ebenenweisen Anfragezerlegung rekombiniert werden.

6.4 Query Rewriting by Contract

Das *Query Rewriting by Contract*-Konzept bestimmt das Rewriting Supremum (siehe Gleichung 6.27) basierend auf einer großen Regelmenge. Die Regeln basieren auf den Erkenntnissen früher Forschungsarbeiten zur Anfrageoptimierung und Informationsintegration (vergleiche Kapitel 5), werden jedoch in vielen Fällen in „entgegengesetzter“ Richtung angewendet. Der Kern der Regeln basiert auf Ersetzungen von Teilen einer Anfrage bzw. eines Anfragefragments.

Jede Ersetzungsvorschrift besteht aus einer linken Seite, welche den unveränderten Teil der Anfrage enthält, und einer rechten Seite mit dem transformierten Anfrageteil. Zusätzlich werden für jede Regel *Invarianten* sowie *Vor-* und *Nachbedingungen* angegeben, welche eine ähnliche Bedeutung wie die Konstrukte in Eiffel (siehe Abschnitt 5.9) besitzen. Invarianten sind zusätzliche Bedingungen, die zu jedem Zeitpunkt während der gesamten Anfragetransformation wahr sein müssen. Beispielsweise muss für die Anwendung einer Regel gelten, dass eine Konstante in einem Attribut-Werte-Vergleich immer größer als 0 sein muss. Die Vor- und Nachbedingungen werden für die obere und untere Ebene der Anfrageverteilung getrennt voneinander erfasst.

Um eine Anfragetransformation auszulösen, müssen alle in der Regel erfassten Vorbedingungen und Invarianten erfüllt sein. In Hinsicht auf das AQuO-Problem sind dies nicht unterstützte Operatoren auf der niederen Ebene, z. B. eine fehlende Aggregatfunktion. Sofern das Anfragefragment die gegebenen Vorbedingungen sowie Invarianten erfüllt und ein Teil des Fragmentes mit der linken Seite der Regel übereinstimmt, wird die Regel ausgeführt. Die Transformationsregel ersetzt dabei den übereinstimmenden Teil des Anfragefragments mit dem rechten Teil der Regel.

Anschließend werden die Nachbedingungen überprüft. Ist jede Nachbedingung erfüllt, so ist die Transformation für das gegebene Anfragefragment abgeschlossen. Ist dies nicht der Fall, so wird jede nicht erfüllte Nachbedingung

in eine neue Vorbedingung umgewandelt. Anschließend müssen diese Vorbedingungen durch die Anwendung einer weiteren Transformationsregel erfüllt werden. Dieses Vorgehen wird solange wiederholt, bis alle Anfragefragmente transformiert wurden sowie jede Nachbedingung und Invariante erfüllt ist.

Um mögliche Zyklen zu verhindern, werden äquivalente Transformationen gesondert annotiert, sodass die mehrfache Ausführung der gleichen Transformationsregel auf demselben Anfrageteil verhindert wird. Durch die Abgeschlossenheit der Containment-Tests, der verhinderten Zyklen und der steigenden Anfragekapazitäten pro höherer Ebene wird die Anfragetransformation definitiv terminieren.

In den folgenden Abschnitten werden einige Transformationsregeln aus dem gesamten Regelwerk exemplarisch vorgestellt und erläutert. Für eine vollständige Auflistung aller Regeln, inklusive deren Invarianten sowie Vor- und Nachbedingungen, sei auf Anhang C verwiesen.

In den nachfolgenden Unterabschnitten werden aufeinander aufbauend die einzelnen Teilkonzepte von Query Rewriting by Contract näher betrachtet. Dabei wird zunächst in Unterabschnitt 6.4.1 das allgemeine Prinzip des Vertauschens von Operatoren eingeführt. Anschließend werden in Unterabschnitt 6.4.2 spezielle Klassen von Anfragen sowie die Rewriting-Supremum-Eigenschaft besprochen, bevor in Unterabschnitt 6.4.3 auf die Erzeugung von Regelketten eingegangen wird. Abschließend wird in Unterabschnitt 6.4.4 auf die Verteilung der Anfragen über mehr als zwei Ebenen eingegangen.

6.4.1 Operatoren zwischen Ebenen vertauschen

Die Verteilung einer Anfrage Q basiert auf der Zerlegung von Q in ein Anfragefragment Q_i und der restlichen Anfrage Q_δ, die auf der nächsthöheren Ebene L_{i+1} ausgeführt wird. Zur Realisierung dieser Zerlegung gehen wir zunächst davon aus, dass alle Operatoren auf der Ebene L_{i+1} ausgeführt werden können, d. h., L_{i+1} verfügt über ausreichend Anfragekapazität, um alle Anfragen direkt ausführen zu können. Wird festgestellt, dass ein einzelner Operator nicht auf L_i ausführbar ist, so muss dieser

a) auf L_{i+1} verschoben werden oder

b) durch die Anwendung einer Regel transformiert werden.

Die θ-$\square$-Regel dient zur Realisierung des ersten Falls, wobei θ einen beliebigen Operator der relationalen Algebra, mit $\{\theta\} \subseteq O_{i+1}$, darstellt. In der folgenden Definition wird die Regel formalisiert. Sie dient zudem als „Vorlage“ für alle weiteren Transformationsregeln.

Definition: θ-$\square$-Regel
Regel: $\theta(r) \sqsubseteq_K \square(r)$
Invarianten: —
Vorbedingungen:

- Untere Ebene L_i: $\{\theta\} \not\subseteq O_i$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: —
- Obere Ebene L_{i+1}: $\{\theta\} \subseteq O_{i+1}$

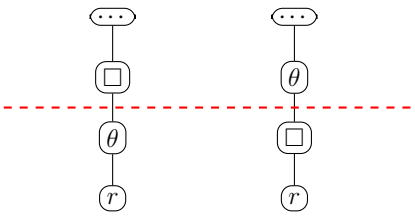


Abbildung 6.5: Visualisierung der θ-$\square$-Regel zur Verlagerung eines beliebigen relationalen Operators auf eine höhere Ebene.

Vertragsbasierte Transformationsregeln werden mit einem K am Vergleichssymbol ($\equiv_K$ bzw. $\sqsubseteq_K$) versehen, damit sie von normalen Query-Containment-Regeln (für welche die Anwendung der Regel nicht immer möglich wäre) unterscheidbar sind. Die Idee hinter der θ-$\square$-Regel ist das Verschieben eines nicht unterstützten Operators θ auf die nächsthöhere Ebene L_{i+1}. Die Transformationsregel ersetzt einen beliebigen relationalen Operator θ von der niederen Ebene L_i durch eine identische Abbildung, gekennzeichnet mit $\square$[2]. Falls θ ein binärer Operator

[2]Für den bzw. die Kindknoten r von θ muss entsprechend $\square(r) \equiv r$ gelten. $\square$ kann auch als imaginärer Operator, No-Op bzw. als reine Weiterleitung der zuvor berechneten (Zwischen-)Ergebnisse aufgefasst werden.

ist, wird für jeden Kindknoten von θ ein $\square$-Operator eingefügt. Auf der höheren Ebene wird entsprechend θ statt $\square$ ausgeführt. Dieser Sachverhalt spiegelt sich in den Vorbedingungen (θ wird nicht auf L_i unterstützt) und Nachbedingungen (θ wird auf L_{i+1} unterstützt) wider.

Die Visualisierung der θ-$\square$-Regel ist in Abbildung 6.5 als Operatorbaum der relationalen Algebra abgebildet. Auf der linken Seite ist die Anfrage vor der Transformation zu sehen. Die Projektion wird auf der niederen Ebene L_i (unterhalb der rot gestrichelten Linie) ausgeführt. Auf der oberen Ebene L_{i+1} erfolgt lediglich die Weiterleitung des auf L_i vorberechneten Zwischenergebnisses. Die rechte Seite der Abbildung stellt die transformierte Anfrage dar, in der θ auf der höheren Ebene L_{i+1} ausgeführt wird. Durch die Transformation erfolgte somit der beabsichtigte Tausch von $\square$ und θ zwischen den beiden Ebenen.

Mittels `...` wird angedeutet, dass oberhalb des betrachteten Anfragefragments weitere Operatoren folgen können. Gleiches gilt für `r` unterhalb des Fragments, jedoch können dort zusätzlich auch Basisrelationen anstatt weiterer Operatoren eingesetzt werden. Das nachfolgende Beispiel verdeutlicht die Auswirkungen der θ-$\square$-Regel anhand einer in SQL formulierten Beispielanfrage:

Beispiel: SQL-Anfrage vor und nach der Transformation mit der θ-$\square$-Regel am Beispiel der Projektion.

In diesem Beispiel wird die Projektion (d. h. θ) auf die Attribute `album` und `length` aus der unteren Ebene (verdeutlicht durch den benannten SFW-Block `X`) auf die obere Ebene verlagert. Der $\square$-Anteil der Regel wird durch eine einfache `SELECT-*`-Anfrage repräsentiert.

```
1 WITH X AS (
2   SELECT album, length
3   FROM tracks
4 )
5 SELECT *
6 FROM X
```

```
1 WITH X AS (
2   SELECT *
3   FROM tracks
4 )
5 SELECT album, length
6 FROM X
```

Basierend auf dem initial erzeugten Anfragebaum wird eine vorläufige Verteilung der Anfrage auf die einzelnen Ebenen vorgenommen. Die θ-$\square$-Regel dient dabei zur Zerlegung in die Anfragefragmente und wird nicht explizit aufgerufen. Nach der Initialverteilung werden weitere Regeln der vertragsbasierten Anfragetransformation ausgeführt. Im Rahmen der Entwicklung des Demonstrators lag der Fokus auf drei Klassen von Transformationen: die klassische Anfrageoptimierung, linear-arithmetische Bedingungen sowie Aggregatfunktionen. Diese drei Regelklassen und deren Anwendung für eine datenschutzkonforme Verteilung von Anfragen werden im nachfolgenden Abschnitt näher erläutert.

6.4.2 Regelklassen der Anfragetransformation

Um unnötig viele Transformationen zu vermeiden, werden zunächst die aus der Anfrageoptimierung bekannten Regeln (siehe Abschnitt 5.8) zur Verlagerung von Selektionen und Projektionen als Heuristik angewendet. Durch diese Heuristik werden bereits unnötig große Zwischenergebnisse für die späteren Verarbeitungsschritte der Anfrage vermieden. Dadurch, dass diese Regeln zur Anfrageoptimierung stets eine äquivalente Anfrage zurückliefern, ist innerhalb der Nähe dieser Knoten keine weitere Anfragetransformation mehr notwendig, sofern alle Operatoren unterstützt werden. Die Anwendung dieser Regelklasse wird im nächsten Textabschnitt näher betrachtet.

Für das verbleibende Anfragefragment, welches aus komplexeren Operatoren, wie Aggregationen und Gruppierungen, bestehen kann, ist durch die Anwendung der obigen Heuristik der Suchraum bereits stark eingeschränkt. Dadurch kann eine passende Transformation einzelner Anfragefragmente schneller gefunden werden, da die Anzahl der zu betrachteten Vorbedingungen sinkt bzw. spezieller (im Sinne der zu unterstützenden Operatoren) wird. Mit den linear-arithmetischen Bedingungen und Aggregatfunktionen werden zwei weitere Regelklassen genauer untersucht.

Klassische Anfrageoptimierung

Als wichtigste Optimierungsregeln zur Minimierung der Zwischenergebnisse werden die Regeln der klassischen Anfrageoptimierung verwendet. Die meisten Regeln der Anfrageoptimierung vertauschen die Operatoren *Selektion* und *Projektion* vorbei an den *Verbund-* und *Mengen*operatoren in Richtung der Blattknoten des betrachteten Operatorbaumes.

Dabei werden im ersten Schritt komplex zusammengesetzte Selektionen in einzelne Selektionsbedingungen aufgelöst. Anschließend werden alle, wenn auch nur teilweise, unterstützte Selektionen in Richtung der Blätter des aufgespannten Anfragebaumes geschoben. Dies betrifft insbesondere die Vertauschung der Selektion mit Verbund-, Projektions- und Mengenoperatoren (siehe u. a. Gleichung 5.13). Zudem werden unterstützte und nicht unterstützten Selektionsprädikate miteinander vertauscht (siehe Gleichung 5.15), damit erstere möglichst früh ausgeführt werden können. Abschließend werden, sofern möglich, die vorkommenden Projektionen ebenfalls in Richtung der Blätter des Anfragebaumes verschoben (siehe u. a. Gleichung 5.14) – bzw. dort zusätzlich eingefügt.

Regel: Kommutativität von Selektion und Projektion – V1: Die folgende Regel stellt die Vertauschung einer einfachen Selektion mit einer Projektion dar, welche Bestandteil des Anfrageoptimierers jedes Datenbanksystems ist. Durch die Kommutativität von Selektion und Projektion werden zwar äquivalente Anfragen erzeugt, durch die unterschiedlichen Vor- und Nachbedingungen werden jedoch zwei unterschiedliche vertragsbasierte Anfragetransformationen generiert. Die hier vorgestellte Variante zeigt, wie die Projektion π_X auf die Attributmenge X aus der unteren Ebene L_i mit der Selektion σ_Y mit den Prädikaten Y aus der oberen Ebene L_{i+1} getauscht wird. Intern wird diese Regel mit `K09`[3] bezeichnet.

Definition: Transformationsregel K09

Regel: $\sigma_Y(\pi_X(r)) \sqsubseteq_K \pi_X(\sigma_Y(r))$

Invarianten: —

Vorbedingungen:

- Untere Ebene L_i: $\{\pi_X\} \not\subseteq O_i$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: $\{\sigma_Y\} \subseteq O_i$
- Obere Ebene L_{i+1}: $\{\pi_X\} \subseteq O_{i+1}$

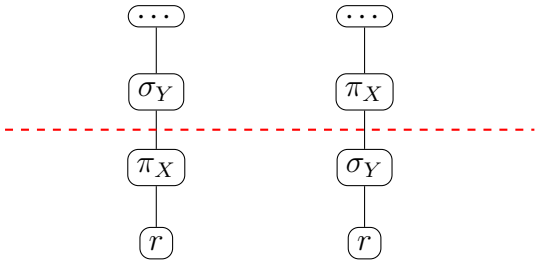


Abbildung 6.6: Visualisierung der Transformation K9

Das folgende Beispiel zeigt die Anwendung der Regel `K09`:

Beispiel: Transformation mit Regel K09

Auf der linken Seite ist die Anfrage vor Anwendung der Transformation zu sehen; die rechte Seite zeigt die Anfrage nach der Anwendung der Regel. Als Selektionsbedingung wurde das Prädikat `length > 250` über der Relation `tracks` gewählt; die Projektion erfolgt auf den Attributen `title` und `length`.

```
1 WITH X AS (
2   SELECT title, length
3   FROM tracks
4 )
5 SELECT *
6 FROM X
7 WHERE length > 250
```

```
1 WITH X AS (
2   SELECT *
3   FROM tracks
4   WHERE length > 250
5 )
6 SELECT title, length
7 FROM X
```

[3] Das *K* steht bei den durchnummerierten Regeln für die Klasse der Klassischen Optimierungsregeln.

Um eine unbeabsichtigte Optimierung seitens des Datenbanksystems zu vermeiden, werden *Common Table Expressions* (CTEs; SQL-Schlüsselwort *WITH*) innerhalb der SQL-Anfragen eingesetzt. Diese verhindern, dass Operatoren zwischen den definierten Teilanfragen vertauscht werden.

Diese Klasse von Optimierungsregeln für das Erreichen der Datensparsamkeit unterscheidet sich im Wesentlichen nicht von den bekannten Regeln der Anfrageoptimierung. Hinzu kommt jedoch eine Randbedingung: Durch die Einsortierung der Selektionsprädikate in die drei Klassen der unterstützten, der teilweise unterstützten sowie der nicht unterstützten Teilziele (siehe Abschnitt 6.2.1) werden die Sel*-Regeln nur bedingt angewendet. Bei der Vertauschung im Anfragebaum werden zunächst nur die vollständig unterstützten Teilziele soweit wie möglich nach unten gezogen. Die teilweise unterstützten Teilziele werden erst im Anschluss nach unten auf die aktuelle Ebene gezogen, verbleiben jedoch stets über den vollständig unterstützten Teilzielen. Bei den nicht unterstützten Teilzielen verhält es sich umgekehrt: Befindet sich ein solches Teilziel auf der aktuellen Ebene, so wird versucht, das Teilziel im Anfragebaum nach oben zu schieben und mit einem Operator zu vertauschen, der unterstützt wird. Wird dies nicht realisiert, so würde die unterste Schicht weniger Teile des Anfrageplans ausführen als sie durch ihre Anfragekapazität in der Lage wäre.

Diese Regelklasse eignet sich vorrangig für Sensoren und eingebettete Systeme, da dort meist einfache Selektionen und Projektionen unterstützt werden. Eine vollständige Übersicht zu allen 21 Regeln dieser Regelklasse ist in Anhang C.3 zu finden. Für den Rest der Anfrage, welche aus komplexeren Operatoren, wie Aggregationen und Gruppierungen, besteht, ist durch die Anwendung der obigen Heuristik der Suchraum bereits stark eingeschränkt. Dadurch kann eine passende Transformation einzelner Anfragefragmente schneller gefunden werden, da die Anzahl der zu betrachtenden Vorbedingungen sinkt bzw. spezieller (im Sinne der zu unterstützenden Operatoren) wird.

Linear-arithmetische Bedingungen

Als weitere Klasse betrachten wir linear-arithmetische Bedingungen. Diese können in Form von Vergleichsoperatoren innerhalb einfacher Attribut-Attribut- und Attribut-Wert-Vergleiche eingesetzt werden und bilden die Grundlage für den Vergleich von aggregierten Werten. Im Rahmen des entwickelten Regelsystems wurden je vier Regeln für Attribut-Attribut- sowie Attribut-Wert-Vergleiche entwickelt und implementiert. Diese Regeln basieren auf den Klassen *LAC1* und *LAC2*, welche in Abschnitt 5.3 eingeführt wurden und auf der Arbeit von Can Türker [Tür99] aufbauen. Die Transformationsregeln der Klasse L fügen, grob gesprochen, jeweils eine zusätzliche Selektionsbedingung mit einfachen arithmetischen Vergleichen in ein Anfragefragment ein.

Regel: Gleich → Größer-gleich (Attribut-Konstante): Wird auf der unteren Ebene L_i nur der Operator $\geq_c$ unterstützt, jedoch nicht der Operator $=_c$, so lassen sich die Daten zunächst mittels $\geq_c$ vorselektieren. Die abschließende Selektion der Daten durch den Operator $=_c$ lässt sich später auf der oberen Ebene L_{i+1} realisieren. Intern wird diese Regel mit `L03` bezeichnet.

Definition: Transformationsregel L03

Regel: $\sigma_{x=c}(r) \sqsubseteq_K \sigma_{x\geq c}(r)$

Invarianten: —

Vorbedingungen:

- Untere Ebene L_i: $\{=_c\} \not\subseteq O_i$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: $\{\geq_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: $\{=_c\} \subseteq O_{i+1}$

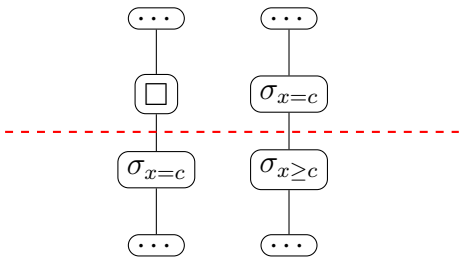


Abbildung 6.7: Visualisierung der Transformation L03

Das folgende Beispiel zeigt die Anwendung der Regel `L03`:

Beispiel: Transformation mit Regel L03

Auf der linken Seite ist erneut die Anfrage vor Anwendung der Transformation zu sehen; die rechte Seite zeigt die Anfrage nach der Anwendung der Regel. Als Selektionsbedingung wurde das Prädikat `length = 250` über der Relation `tracks` gewählt.

```
WITH X AS (
  SELECT aid, length
  FROM tracks
  WHERE length = 250
)
SELECT aid, length
FROM X
```

```
WITH X AS (
  SELECT aid, length
  FROM tracks
  WHERE length >= 250
)
SELECT aid, length
FROM X
WHERE length = 250
```

Regeln dieser Bauart erscheinen zunächst überflüssig, da durch Datenbanksysteme bzw. deren Anfragesprachen alle Vergleichsoperatoren unterstützt werden. Betrachten wir die unterste Schicht unserer Edge-Architektur (siehe Abbildung 6.2), so entdecken wir viele Sensoren mit geringem Leistungsumfang. Diese werden in vielen Fällen mit einfachen FPGA-Gattern programmiert, wodurch nur ein geringer Funktionsumfang bereitgestellt ist. Abbildung 5.3 zeigt einen exemplarischen Aufbau eines solchen Gatters. Durch die begrenzte Auswahl an zur Verfügung stehenden Vergleichs- und Aggregationsoperatoren wird deutlich, dass die Transformationsregeln der Klasse L (und wie wir später sehen werden auch die der Klasse A) durchaus ihre Daseinsberechtigung haben[4].

Eine vollständige Übersicht zu allen acht Regeln dieser Regelklasse ist in Anhang C.1 zu finden. Im nachfolgenden Unterabschnitt übertragen wir die Transformationen von einfachen Attributvergleichen auf Aggregatfunktionen, für die speziellere Anforderungen an die Vor- und Nachbedingungen sowie an die Invarianten gelten.

Aggregatfunktionen

Aggregatfunktionen spielen eine entscheidende Rolle für komplexere Auswertungen, wie beispielsweise der Linearen Regression oder der Ermittlung von Statistiken über die Worthäufigkeiten in großen Dokumentkollektionen. Da Aggregatfunktionen auf leistungsschwacher Hardware nicht immer direkt implementiert werden können (siehe hierfür wieder Abbildung 5.3), besteht auch hier ein großer Bedarf an diesen Transformationsregeln. In die Regelklasse für Aggregatfunktionen (Klasse A) fallen die folgenden drei Teilklassen:

1. Vergleich eines aggregierten Wertes mit einer Konstanten,
2. Vergleich zweier aggregierter Werte und
3. Ersetzung einer Aggregatfunktion durch einen Allquantor (und umgekehrt)

Jede dieser Teilklassen zielt darauf ab, eine nicht unterstützte Aggregatfunktion durch eine andere, unterstützte Aggregatfunktion bzw. eine allquantifizierte Unteranfrage zu ersetzen. Die Regeln in diesem Unterabschnitt basieren auf den in Abschnitt 5.3 eingeführten Ergebnissen von Can Türker [Tür99]. Betrachten wir zunächst anhand eines Beispiels die Transformation von Aggregatanfragen, die einen Vergleich mit einer Konstanten beinhalten.

Regel: Maximum-gleich → Durchschnitt-kleiner-gleich (Attribut-Konstante): Die folgende Regel ersetzt ein Vergleichsprädikat zwischen dem nicht unterstützten Operator Maximum (`max`) eines Attributes `X` und einer

[4]Dies ist auch einer der Gründe dafür, dass das obige Prädikat nicht als $\sigma_{x \geq c \wedge c \geq x}$ dargestellt werden kann und somit eine äquivalente Anfragetransformation auf der unteren Ebene vorliegt. Durch die beschränkte Anfragekapazität kann weder sichergestellt werden, dass das logische Und unterstützt wird, noch dass das Attribut und die Konstante vertauscht werden können.

Konstanten `c` auf der Ebene L_i durch den arithmetischen Durchschnitt (`avg`). Dabei kippt zusätzlich der Vergleichsoperator von $=_c$ zu $\leq_c$. Die eigentlich zu realisierende Selektionsbedingung `max(X)=c` wird abschließend auf der Ebene L_{i+1} ausgeführt, wo die Operatoren `max` und $=_c$ unterstützt werden.

Auf der unteren Ebene L_i wird durch die Selektion auf dem arithmetischen Durchschnitt (`avg(X)`) eine Obermenge der Tupel zurückgegeben, die bei Anwendung der Selektion auf dem Maximum ermittelt worden wären. Diese Eigenschaft beruht auf den Ergebnissen von Türker [Tür99]: der Durchschnitt über einem Attribut `X` ist stets kleiner als dessen Maximum. Intern wird diese Regel mit `A10` bezeichnet.

Definition: Transformationsregel A10

Regel: $\sigma_{max(X)=c}(r) \sqsubseteq_K \sigma_{avg(X)\leq c}(r)$

Invarianten: —

Vorbedingungen:

- Untere Ebene L_i: $\{max, =_c\} \not\subseteq O_i$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: $\{avg, \leq_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: $\{max, =_c\} \subseteq O_{i+1}$

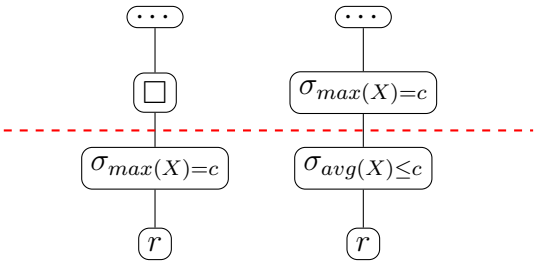


Abbildung 6.8: Visualisierung der Transformation A10

Das folgende Beispiel zeigt die Anwendung der Regel A10:

Beispiel: Transformation mit Regel A10

Auf der linken Seite ist die Anfrage vor Anwendung der Transformation zu sehen; die rechte Seite zeigt die Anfrage nach der Anwendung der Regel. Als Selektionsbedingung in der `HAVING`-Klausel wurde das Prädikat `max(length) = 250` gruppiert über die ID der Alben, `aid`, gewählt.

```
SELECT aid
FROM tracks
GROUP BY aid
HAVING max(length) = 250
```

```
WITH X AS (
  SELECT aid, length
  FROM tracks
  WHERE aid IN (
    SELECT aid
    FROM tracks
    GROUP BY album
    HAVING avg(length) <= 250
  )
)
SELECT aid
FROM X
GROUP BY album
HAVING max(length) = 250
```

Damit der Vergleich der Aggregatfunktion mit der Konstanten (`max(length) = 250`) auf L_{i+1} realisiert werden kann, ist ein zusätzlicher Schritt notwendig. Durch die Verwendung der Unteranfrage in L_i wird auf diejenige Alben-IDs zugegriffen, welche die zusätzlich eingefügte Selektionsbedingung erfüllen. Dadurch wird eine erneute Verwendung der Attributwerte von `length` ermöglicht, welche für die äußere Aggregation benötigt werden. Je nach Datenbanksystem stehen hierfür auch alternative Implementierungen der SQL-Anfrage zu Verfügung. Darauf wird in Abschnitt 6.5 kurz eingegangen.

Vergleich zweier aggregierter Werte: Für die Transformation von Anfragen, welche den Vergleich zweier aggregierter Werte enthalten, wird die Konstante `c` in den Regeln der Klasse A durch eine weitere Aggregatfunktion

ausgetauscht. Als weitere Bedingung muss nach Türker [Tür99] gelten, dass die verwendeten Attribute in beiden Aggregatfunktionen gleich sein müssen, da ansonsten nur überlappende Containment-Beziehungen ermittelt werden können.

Für die so entstandenen Regeln gelten die vereinigten Vor- und Nachbedingungen sowie Invarianten der beiden ursprünglichen Regeln. Das bedeutet im Detail, dass die zusammengesetzten Regeln immer dann ausgeführt werden können, wenn alle Invarianten und mindestens eine Vorbedingung (im Sinne eines nicht unterstützten Operators auf der Ebene L_i) erfüllt sind. Eine erfolgreiche Transformation wird erreicht, wenn alle Nachbedingungen erfüllt sind, d. h., alle Operatoren werden auf der Ebene L_{i+1} unterstützt.

Als letzte Teilklasse untersuchen wir die Ersetzung eines Allquantors durch eine Aggregatfunktion. Die entgegengesetzte Richtung kann dem Regelwerk aus Anhang C entnommen werden.

Regel: Alle-kleiner zu Maximum-kleiner (Attribut-Konstante): Als konkrete Regel betrachten wir, wie die Prüfung, ob alle Werte `kleiner als` eine Konstante sind, durch die die Aggregatfunktion `max` ersetzt wird. Dabei wird der Allquantor ($\forall$) auf der Ebene L_i nicht unterstützt, der Vergleich mittels $<_c$ hingegen schon. Die neu zu realisierende Selektionsbedingung `max(X) < c` wird statt dem Allquantor auf der Ebene L_i ausgeführt, wo der Operator `max` unterstützt wird.

Auf der oberen Ebene L_{i+1} werden sowohl vor als auch nach der Transformation keine relevanten relationalen Operatoren ausgeführt. Die Transformation dient vielmehr der Vorbereitung für weitere Transformationsschritte, wenn beispielsweise der Vergleichsoperator $<$ nicht auf L_i unterstützt wird und durch weitere Umformungen, wie aus der Klasse `L`, modifiziert werden muss.

Diese Unterklasse von Transformationen beruht wieder auf den Ergebnissen von Türker [Tür99]. In der Visualisierung der Anfragetransformation wird aus Gründen der Übersichtlichkeit die optionale Gruppierung nicht dargestellt, während sie im SQL-Beispiel auftritt.

Definition: Transformationsregel A27

Regel: $\sigma_{\forall x \in X : x < c}(r) \sqsubseteq_K \sigma_{max(X) < c}(r)$

Invarianten: —

Vorbedingungen:

- Untere Ebene L_i: $\{\forall\} \not\subseteq O_i$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: $\{max, <_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: —

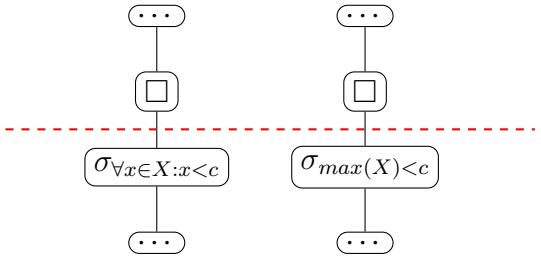


Abbildung 6.9: Visualisierung der Transformation A27

Das folgende Beispiel zeigt die Anwendung der Regel A27:

Beispiel: Transformation mit Regel A27

Auf der linken Seite ist die Anfrage vor Anwendung der Transformation zu sehen; die rechte Seite zeigt die Anfrage nach der Anwendung der Regel. Als Selektionsbedingung in der `HAVING`-Klausel wurde das Prädikat `max(length) < 250`, gruppiert über die ID der Alben (`aid`), gewählt.

```
SELECT DISTINCT aid
FROM tracks
WHERE aid NOT IN (
  SELECT aid
  FROM tracks
  WHERE NOT(length < 250)
)
```

```
SELECT DISTINCT aid
FROM tracks
GROUP BY aid
HAVING max(length) < 250
OR max(length) IS NULL
```

Zusätzlich wurde in der transformierten Anfrage das Prädikat `max(length) IS NULL` eingeführt, damit auch diejenigen Alben mit beachtet werden, deren Tracks keine Längenangaben haben. In dem für die Evaluierung verwendeten System PostgreSQL, welches sich sehr am SQL-Standard [Mel16] orientiert, werden Nullwerte standardmäßig mittels `NULLS FIRST` sortiert. Dies hat zur Folge, dass Nullwerte und, im Falle von Vergleichen auf Nullwerten, unbekannte Werte (engl.: *unknown*) stets kleiner als alle anderen Vergleichswerte sind, aber nicht als Werte in die Aggregation einfließen. Dadurch würde das Weglassen des Prädikates `max(length) IS NULL` in der `HAVING`-Klausel weniger Ergebnistupel zurückliefern als die Originalanfrage.

Die Anwendung einer einzelnen Regel hat für Geräte auf der Ebene L_i mit sehr beschränkter Anfragekapazität nur einen geringen Nutzen, da neben einzelnen Operatoren (z. B. die Aggregatfunktion *Maximum*) auch ganze Operatorklassen (wie z. B. Aggregate allgemein oder Verbundanweisungen) zu den nicht-unterstützten Operatoren zählen können. Daher sind nach Anwendung einer Transformationsregel nicht zwangsläufig alle Nachbedingungen erfüllt. Werden diese negiert und als neue Vorbedingungen für die Ebene L_i eingesetzt, so können auf dem gleichen Anfragefragment mehrere Regeln hintereinander ausgeführt werden. Im nachfolgenden Unterabschnitt 6.4.3 betrachten wir genauer, wie diese Regelketten erzeugt werden.

6.4.3 Erzeugung neuer Regeln

Für die Erzeugung neuer Anfragetransformationen werden zwei oder mehr Regeln aus dem bestehenden Regelverzeichnis (siehe Anhang C) benötigt. Wir gehen im Folgenden o. B. d. A. davon aus, dass wir je zwei Regeln betrachten. X, Y und Z stellen im Folgenden die auszuführenden Operatoren auf L_i bzw. L_{i+1} dar. Die dazugehörigen Transformationsregeln, $X \sqsubseteq_K Y$ und $Y \sqsubseteq_K Z$, werden nacheinander auf dem gleichen Anfragefragment ausgeführt, woraus die Transformationskette $X \sqsubseteq_K Y \sqsubseteq_K Z$ entsteht.

Dabei können jedoch zusätzliche Operatoren auf der unteren bzw. oberen Ebene entstehen, die nicht ausgeführt werden müssen. Um dieses Problem zu umgehen, müssen diese unnötigen Operatoren während der Anfragetransformation erkannt werden. Dies lässt sich durch eine einfache Überprüfung realisieren, bei der neue Regeln durch Inferenz abgeleitet werden.

Werden X und Y auf der oberen Ebene L_{i+1} ausgeführt und Z auf der unteren Ebene L_i, so kann Y übersprungen werden, da es kein *Rewriting Supremum* bezüglich L_{i+1} darstellt. Wird hingegen X auf L_{i+1} und Y und Z auf L_{i+1} ausgeführt, so lässt sich Z überspringen. Das folgende Beispiel und Abbildung 6.10 verdeutlichen diesen Sachverhalt für beide Fälle:

Beispiel: Erzeugung einer neuen Regel auf Basis von $A14$ und $A23$

Zur Veranschaulichung wird der zweite Fall an einem einfachen Beispiel illustriert. Als X verwenden wir eine Selektionsbedingung zwischen der Minimum-Funktion über dem numerischen Attribut A, welches größer als die Konstante c sein soll ($min(A) > c$). Y und Z vergleichen die Summe ($sum(A) > c$) bzw. den Durchschnitt ($avg(A) > 0$) über das gleiche Attribut, wobei der Durchschnitt nicht mit c, sondern mit 0 verglichen wird.

Zur Umsetzung dieser Transformationsschritte verwenden wir zwei Regeln aus dem Regelverzeichnis in Anhang C, welche zur Klasse der Aggregatanfragen gehören. Die erste Transformationsregel, $A14$, ist wie folgt definiert:

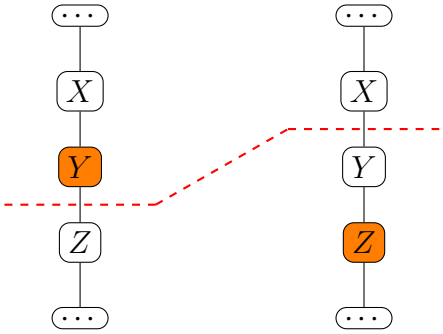


Abbildung 6.10: Veranschaulichung der überflüssigen Anwendung von Operatoren bei der Anfragetransformation. Bei der linken Anfragezerlegung ist der Operator Y überflüssig, bei der rechten Zerlegung hingegen Z, sofern die Beziehung $X \sqsubseteq_K Y \sqsubseteq_K Z$ gilt.

Regel: $\sigma_{min(X)\geq c}(r) \sqsubseteq_K \sigma_{sum(X)\geq c}(r)$
Invarianten: $c \geq 0$
Vorbedingungen:

- Untere Ebene L_i: $\{min\} \not\subseteq O_i$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: $\{sum, \geq_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: $\{min, \geq_c\} \subseteq O_{i+1}$

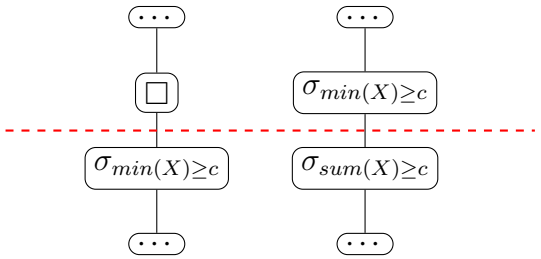


Abbildung 6.11: Visualisierung der Transformation A14

Die Regel setzt als Vorbedingung voraus, dass der Operator min nicht auf der unteren Ebene ausgeführt werden kann. Durch die Ersetzung von min durch sum gilt als Nachbedingung auf L_i, dass sum sowie $\geq_c$ in O_i enthalten sein müssen. Wird auch der sum-Operator auf der unteren Ebene nicht unterstützt, so muss eine weitere Transformationsregel, in diesem Beispiel $A23$, angewendet werden. Die Regel ist wie folgt definiert:

Regel: $\sigma_{sum(X)\geq c}(r) \sqsubseteq_K \sigma_{avg(X)\geq 0}(r)$
Invarianten: $c \geq 0$
Vorbedingungen:

- Untere Ebene L_i: $\{sum\} \not\subseteq O_i$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: $\{avg, \geq_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: $\{sum, \geq_c\} \subseteq O_{i+1}$

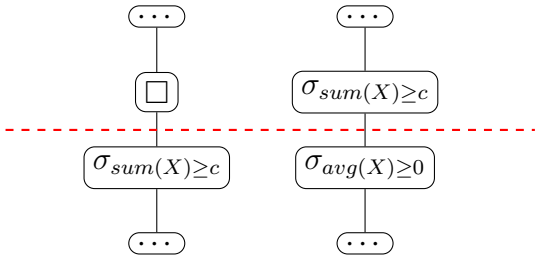


Abbildung 6.12: Visualisierung der Transformation A23

Wird davon ausgegangen, dass $min(A) \geq c$ und $sum(A) \geq c$ nicht unterstützt werden, so lässt sich zunächst Regel $A14$ anwenden. Dies ist in Abbildung 6.13 im Übergang vom ersten Anfragebaum zum zweiten Anfragebaum (von links gesehen) dargestellt. Da dies nicht zur Erfüllung der Nachbedingungen genügt, muss eine weitere Transformation erfolgen. Hierfür wird Regel $A23$ verwendet. In Abbildung 6.13 ist das Ergebnis der Transformation im dritten Anfragebaum zu sehen.

Wie leicht zu erkennen ist, kann die Selektionsbedingung $\sigma_{sum(A)\geq c}$ entfernt werden, da diese durch die ursprüngliche Selektionsbedingung $\sigma_{min(A)\geq c}$ überdeckt wird. Die reduzierte Anfrage ist in Abbildung 6.13 im rechten Anfragebaum dargestellt.

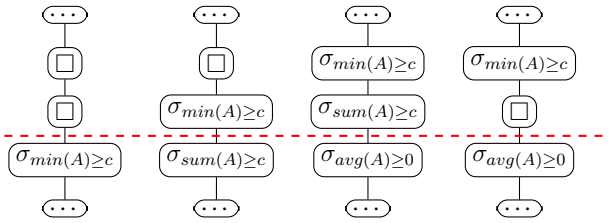


Abbildung 6.13: Kombination der Transformationsregeln $A14$ und $A23$ durch Inferenz.

Dies führt intern zu der folgenden neuen Transformationsregel:
Regel: $\sigma_{min(A)\geq c}(r) \sqsubseteq_K \sigma_{avg(A)\geq 0}(r)$
Invarianten: $c \geq 0$
Vorbedingungen:

- Untere Ebene L_i: $\{min\} \subseteq O_i$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: $\{avg\} \subseteq O_i$
- Obere Ebene L_{i+1}: $\{min, \geq\} \subseteq O_{i+1}$

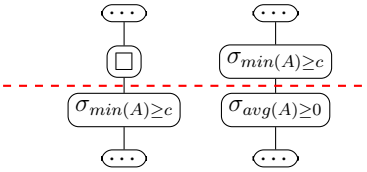


Abbildung 6.14: Die neu erzeugte Transformation

Die neu erzeugten Regeln werden nur intern verwendet, d. h., sie werden nur für die gegebene Anfrage genutzt und nicht im Regelwerk hinterlegt. Dies verhindert, dass für zukünftige Anfragen in anderen Systemumgebungen ein zu großer Suchraum entsteht und ggf. unpassende, wenngleich auch formal korrekte, Anfragetransformationen vorgeschlagen werden.

Als letztes Teilkonzept widmet sich der nächste Unterabschnitt der Verteilung von Anfragen über n Ebenen. Dafür wird ein rekursiver Algorithmus vorgestellt, der die Originalanfrage Q schrittweise in mehrere Teilanfragen $Q_1, \ldots, Q_n$ überführt.

6.4.4 Anfragetransformation als iterativer Algorithmus

Ausgehend von den bisherigen konzeptuellen Beschreibungen wurde ein Operatorbaum erstellt, welcher angibt, welche Teilausdrücke durch Operatoren miteinander verknüpft bzw. weiterverarbeitet werden. Die Blätter des Baumes stellen dabei die beteiligten Basisrelationen dar; die Wurzel ist der abschließende Operator, der das Endergebnis zurückliefert.

In den bisherigen Betrachtungen wurde davon ausgegangen, dass die Anfrage in zwei Teilanfragen transformiert wird. Dabei beantwortete die Teilanfrage Q_{i+1} auf der oberen Ebene L_{i+1} alle Teilziele, welche nicht durch die Teilanfrage Q_i auf der unteren Ebene L_i realisiert werden konnten. Falls auf der Ebene L_{i+1} keine unbeantworteten Teilziele mehr existieren, ist die Abarbeitung der Anfrage abgeschlossen. Das Ergebnis D_{i+1} ist ab diesem Punkt äquivalent zur Ausführung der Originalanfrage Q auf dem ursprünglichen Datenbestand D, wodurch D_{i+1} direkt an die oberste Ebene L_n weitergeleitet werden kann.

Ist dies nicht der Fall, muss die Prüfung der Anfragekapazitäten und die Anfragetransformation solange wiederholt werden, bis die oberste Ebene L_n erreicht ist. Für den Fall, dass auf der obersten Ebene immer noch Teilziele existieren, für welche nicht die notwendigen Anfragekapazitäten vorhanden sind, scheitert die Ausführung der transformierten Anfrage. Die bedeutet allerdings auch, dass die eingegebene Anfrage $Q(D)$ nicht direkt auf der

Ebene L_n mit dem Datenbestand D hätte ausgeführt werden können, da Q aus den gleichen Basisoperatoren aufgebaut ist wie die transformierte Anfragekette Q'. Dies sollte in realen Einsatzszenarien nicht der Fall sein.

Auf $L_0 \in L$ als unterste Ebene werden die Rohdaten verarbeitet, während auf $L_n \in L$ die letzten Berechnungen zur Bestimmung des Ergebnisses der Anfrage Q ausgeführt werden. Der folgende Algorithmus 8 übernimmt die beiden Eingaben Q und L und berechnet als Ausgabe eine Menge von maximalen Teilanfragen $Q_{i,j}$, welche auf den jeweiligen Ebenen L_i ausführbar sind. Durch den Index j werden Teilanfragen innerhalb einer Ebene L_i durchnummeriert, da ggf. durch fehlende Verbund- und Mengenoperatoren mehrere Teilbäume entstehen können.

Algorithmus 8: Anfrageverteilung auf mehrere Ebenen

Data: Originalanfrage Q als Operatorbaum, Menge von Ebenen $L_i \in L$ mit den jeweils unterstützten Operatoren O_i
Result: Eine Menge von Teilanfragen $Q_{i,j}$

```
// Initialisierung mit Basisrelationen
i := 0;
Q_{0,j} := blaetter(Q);
while i ≤ n do
    foreach Q_{i,j} do
        if v := vater(Q, Q_{i,j}) ∈ O_i then
            // Wenn Vaterknoten unterstützt wird, dann füge ihn hinzu
            Q_{i,j} := v(Q_{i,j});
        else
            // Sonst: Baum abschneiden
            Weise Q_{i,j} der Ebene L_i zu;
            Ersetze Q_{i,j} durch D_{i,j} in Q;
            i++;
        end
    end
    // Bei Verbund und Mengenoperatoren Teilbäume zusammenfassen
    Kombiniere Q_{i,j} mit dem gleichen Vaterknoten;
end
if L_n enthält wurzel(Q) then
    // Bei erfolgreicher Anfragetransformation alle Teilanfragen
       zurückgeben
    return ⋃ Q_{i,j};
else
    // Sonst: Leere Menge als Zeichen des Fehlschlags
    return ∅;
end
```

Der Algorithmus arbeitet in drei Teilschritten:

1. Zunächst werden die Teilanfragen mit den verwendeten Basisrelationen initialisiert: $Q_{0,j} := blaetter(Q)$.
2. Anschließend werden iterativ solange die Vaterknoten unter Verwendung möglicher Regeln der Anfragetransformation hinzugefügt ($Q_{i,j} := v(Q_{i,j})$), bis ein Operator nicht mehr unterstützt werden kann (Weise $Q_{i,j}$ der Ebene L_i zu).
3. Werden Verbund- ($\bowtie$) oder Mengenoperatoren ($\cup, \cap, \ldots$) auf einer Ebene unterstützt, werden die darunterliegenden Teilanfragen zusammengefasst.

Als Ergebnis einer erfolgreichen Anfragetransformation wird die Menge aller Teilanfragen zurückgegeben (**return** $\bigcup Q_{i,j}$). Im Falle eines Fehlschlages wird hingegen die leere Menge als Ergebnis ausgegeben, auch wenn einzelne Teilanfragen unterstützt werden.

Ausgehend von den Konzeptbeschreibungen in diesem Abschnitt widmet sich der nächste Abschnitt der Implementierung des Konzepts. Dabei wird in Abschnitt 6.5 nur auf die wesentlichsten Aspekte der Umsetzung eingegangen, während Anhang F weitere technische Details bereitstellt.

6.5 Implementierung

Dieser Abschnitt behandelt die Implementierung der AQuO-Anfragetransformation, wie sie in den vorherigen Abschnitten als Konzept eingeführt wurde. Dazu wird in Unterabschnitt 6.5.1 auf die notwendigen Teilschritte zur Umsetzung der Anfragetransformation eingegangen. Anschließend werden in Unterabschnitt 6.5.2 die verwendeten Werkzeuge und deren Rolle bei der Anfragetransformation eingeführt. Unterabschnitt 6.5.3 zeigt die konkrete Umsetzung der Anfragetransformation anhand einer einzelnen Beispielanfrage. In Unterabschnitt 6.5.4 wird abschließend die grafische Oberfläche des Standalone-Demonstrators vorgestellt. Weiterführende Details werden in den Anhängen C (Regelverzeichnis) und F (Demonstrator) dargelegt.

6.5.1 Schritte der Anfragetransformation

Die vollständige Implementierung des entwickelten Demonstrators zur vollständigen Anfrageverarbeitung gliedert sich in insgesamt elf Phasen. Die fett hervorgehoben Punkte entsprechen den bekannten Phasen aus dem Bereich der Informationsintegration in föderierten Systemen nach Leser und Naumann [LN07]:

1. Sichtauflösung (Apache Calcite)
2. Entschachtelung und Group-By-Erweiterungen auflösen (Apache Calcite)
3. Übersetzung in Relationenalgebra (Apache Calcite)
4. Konjunktive Normalform herstellen (Apache Calcite)
5. Voroptimierung und **Anfrageplanung** (Apache Calcite und DBMS)
6. Zerlegung komplexer Aggregate (Präprozessor PArADISE)
7. **Optimierung** bzgl. Datensparsamkeit (AQuO-Anfragetransformation; Präprozessor PArADISE)
8. **Anfrageübersetzung** für das konkrete System (Apache Calcite)
9. **Anfrageausführung** der maximal unterstützten Teilbäume (DBMS)
10. ggf. Anonymisierung der Daten (Postprozessor PArADISE)
11. **Ergebnisintegration** (Postprozessor PArADISE)

In Klammern werden jeweils die zuständigen Systeme bzw. Teilkomponenten des PArADISE-Anfrageprozessors aufgeführt. Die verwendeten externen Tools, Apache Calcite und PostgreSQL als DBMS, werden in Unterabschnitt 6.5.2 näher vorgestellt; der Postprozessor von PArADISE in Kapitel 3.

Die Phasen 7 bis 11 werden dabei für jeden an der Anfrageverarbeitung beteiligten Knoten wiederholt. Die Verarbeitung beginnt dabei auf der Sensorebene L_0 und endet auf der höchsten Ebene L_n. Algorithmus 9 verdeutlicht dieses Vorgehen.

Algorithmus 9: Algorithmus zur Anfragetransformation im Überblick

Data: Eine Anfrage Q, eine Menge von Ebenen $L := \{L_0, L_2, \ldots, L_n\}$, eine Datenbankinstanz d
Result: Die Ergebnisrelation RS der Anfrage $Q(d)$
$Q \leftarrow Sichtauflösung(Q)$;
$Q \leftarrow Entschachtelung(Q)$;
$Q \leftarrow SQL2Relationenalgebra(Q)$;
$Q \leftarrow KNF(Q)$;
$Q \leftarrow Voroptimierung(Q)$;
$Q \leftarrow AggregatGroupByZerlegung(Q)$;
$RS \leftarrow d$;
for $(i := 0; i \leq n; i++)$ **do**
  $Q_i \leftarrow Transformation(Q, L_i)$;
  $Q_i \leftarrow Anfrageübersetzung(Q_i)$;
  $RS_i \leftarrow Anfrageausführung(Q_i, RS)$;
  $RS_i \leftarrow Anonymisierung(RS_i)$;
  $RS \leftarrow Ergebnisintegration(RS_i)$;
end
return RS;

Schritt 1: Sichtauflösung

Im ersten Schritt werden mögliche Sichtdefinitionen in der Anfrage aufgelöst. Die Informationen über die Sichten werden dabei über die Metadaten der Datenbankverbindung ermittelt. Jede Schicht fragt dabei die Sichtdefinitionen der dazugehörigen unteren Schichten mit ab, um für die Anfrageverarbeitung alle nötigen Daten bereitzustellen.

Zur Sichtauflösung müssen zwei Änderungen an der originalen Anfrage Q vorgenommen werden: Zum einen müssen die Namen der Sichten in den `FROM`-Klauseln durch die Sichtdefinition ausgetauscht werden. Zudem müssen die in der `SELECT`- und `WHERE`-Klausel verwendet Attribute umbenannt werden, sofern sie in der Sichtdefinition ebenfalls umbenannt wurden. Durch die Sichtauflösung kommen evtl. zusätzliche Aggregatfunktionen, Gruppierungen und Selektionen in die Anfrage, wodurch zusätzliche Optimierungen bzgl. der Datensparsamkeit möglich sind, die ansonsten nicht erkannt werden würden.

Schritt 2: Entschachtelung und Auflösung von Group-By-Erweiterungen

Durch die Entschachtelung der Anfrage werden Unteranfragen entnestet und die Anfrage somit vereinfacht. Gleiches gilt für die Auflösung von `GROUP BY`-Erweiterungen für Data Warehouses, wie `GROUPING SETS`, `ROLLUP`, und `CUBE`, welche lediglich syntaktische Abkürzungen darstellen[5]. Wie bei der Transformation von Aggregatfunktionen werden die Teilbäume der vereinfachten Gruppierungen annotiert, um ggf. die Auflösung zurückzunehmen, sofern die Gruppierungen nicht auf die unteren Schichten verteilt werden konnten. Nach der Entschachtelung wird die transformierte SQL-Anfrage im nächsten Schritt in eine Anfrage der Relationenalgebra umgeformt.

Schritt 3: Übersetzung in Relationenalgebra

Durch die Übersetzung der SQL-Anfrage in die Relationenalgebra wird im Weiteren eine effizientere Verarbeitung und Optimierung der Anfrage ermöglicht. Zudem bietet ein relationenalgebraischer Ausdruck in Schritt 8 die Möglichkeit, Anfragen für beliebige (No)SQL-Systeme, beispielsweise in Form von XPath- und JsonPath-Ausdrücken, zu generieren. Dadurch wird ein breiteres Spektrum an Systemen unterstützt.

[5]Dieses Vorgehen erscheint zunächst ineffizient, durch das Verschieben einzelner Teiloperatoren auf die tieferen Ebenen kann, wie in Unterabschnitt 6.6.3 gezeigt, die Effizienz des Systems durch die parallele Ausführung einzelner Anfragefragmente gesteigert werden.

Intern wird der Ausdruck der Relationenalgebra als abstrakter Syntaxbaum auf Basis von XML gespeichert. Dies erlaubt eine einfache und schnelle Adressierung, Auswertung und Transformation der Anfrage mittels XML-Anfrage- und Transformationssprachen wie XPath und XSLT. Der Anfragebaum dient als Ausgangsbasis für die folgenden Schritte 4 bis 8.

Die Übersetzung von SQL in die Relationenalgebra wurde teilweise durch Apache Calcite realisiert. Calcite verwendet eine eigene, interne Darstellung für die Relationenalgebra in Form von Java-Objekten. Diese wurde auf eine eigens für PArADISE entwickelte vereinfachte Repräsentation übertragen, die in Form von XML-Dokumenten mittels des Simple-Frameworks (siehe Unterabschnitt 6.5.2) persistent gemacht wurden. Das folgende Beispiel zeigt diesen Übersetzungsprozess anhand einer einfachen Anfrage.

Beispiel: Überführung einer SQL-Anfrage in eine XML-Repräsentation

Für dieses Beispiel verwenden wir erneut die bekannte Musikdatenbank. Die folgende SQL-Anfrage beinhaltet zwei Selektionsbedingungen und zwei Basisrelationen, die durch einen natürlichen Verbund kombiniert werden.

```
SELECT title, length
FROM tracks NATURAL JOIN genres
WHERE name < 'Videospiel-OST'
AND length < 42000
```

Calcite bildet die einzelnen SQL-Klauseln auf Java-Objekte ab, die durch PArADISE in ein XML-Dokument überführt werden. In diesem Beispiel wird die `WHERE`-Klausel auf zwei `selection`- bzw. `predicate`-Tags abgebildet. Die Attributnamen werden mittels Calcite durch deren Positionen angegeben, daher tauchen im Quellcodeausschnitt Bezeichnungen wie `$12`, `$4` und `$7` statt der Attributnamen `tid`, `gid` und `length` auf.

```
<projection isDistinct="false">
  <naturaljoin>
    <projection isDistinct="false">
      <selection>
        <baseRelation name="tracks">
          <attributes/>
        </baseRelation>
        <predicate class="dev.ra.predicate.BinaryPredicate"
          leftSide="$12" not="false" op="&lt;" rightSide="42000"/>
      </selection>
      <attributes>
        <attribute>$12</attribute>
        <attribute>$4</attribute>
        <attribute>$7</attribute>
      </attributes>
    </projection>
    <projection isDistinct="false">
    <selection>
      <baseRelation name="genres">
        <attributes/>
      </baseRelation>
      <predicate class="dev.ra.predicate.BinaryPredicate"
        leftSide="$1" not="false" op="&lt;"
        rightSide="'Videospiel-OST'"/>
    </selection>
```

```
26      <attributes>
27        <attribute>$0</attribute>
28      </attributes>
29      </projection>
30    </naturaljoin>
31    <attributes>
32      <attribute>$1</attribute>
33      <attribute>$2</attribute>
34    </attributes>
35  </projection>
```

In der XML-Darstellung sind bereits die Schritte 4 und 5 eingearbeitet. Dies ist insbesondere daran zu erkennen, dass die Selektionsbedingungen auf die jeweiligen Basisrelationen angewendet werden und nicht auf das Zwischenergebnis nach dem Verbund. Für die vollständige Implementierung der Transformation sei auf das Begleitmaterial (siehe Anhang G) bzw. das GitLab zum Teilprojekt[a] verwiesen.

[a] GitLab des Bereiches Informatik an der Universität Rostock: `https://git.informatik.uni-rostock.de/hg/patch-queryrewriter`, zuletzt aufgerufen am 05.01.2022

Schritt 4: Konjunktive Normalform herstellen

Anfragen in konjunktiver Normalform sind die Grundlage vieler Optimierungsalgorithmen. Ein komplexes Selektionsprädikat P ist in konjunktiver Normalform (KNF), wenn es eine Konjunktion von Disjunktionen von Literalen der Form $L_{i,j}$ ist:

$$P := (L_{1,1} \vee \cdots \vee L_{1,m_1}) \wedge \cdots \wedge (L_{n,1} \vee \cdots \vee L_{n,m_n}) \tag{6.44}$$

Die Literale $L_{i,j}$ stellen dabei einfache Attribut-Wert- bzw. Attribut-Attribut-Vergleiche dar. Durch die konjunktive Normalform können die einzelnen `UND`-verknüpften Prädikate durch verschiedene Regeln, wie beispielsweise dem Vertauschen der Selektionsprädikate, optimiert werden. Diese Transformation der Anfrage wird in der Implementierung des Demonstrators ebenfalls mittels Apache Calcite realisiert.

Schritt 5: Voroptimierung und Anfrageplanung

Die Anfrage wird in einen oder mehrere Anfragepläne übersetzt. Basierend auf den Eigenschaften des DBMS und der konkreten Datenbank wird ein kostenoptimaler Anfrageplan ausgewählt, der am effizientesten umsetzbar ist.

Dieses Vorgehen eignet sich primär für Datenquellen, die auf einem zentralen Rechner gehalten werden. Nichtsdestotrotz eignen sich die verwendeten Heuristiken, wie das Verschieben von Selektionsprädikaten nahe an die Basisrelationen, um den initialen Anfragebaum für die Anfragetransformation mittels AQuO zu erstellen. Durch die Heuristiken werden einfache Operatoren nahe an die Blätter des Anfragebaumes geschoben, während komplexere Operatoren erst später ausgeführt werden.

Die Voroptimierung und die Anfrageplanung werden ebenfalls mittels Apache Calcite realisiert. Calcite erhält dabei die Zugangsdaten zu dem Datenbankmanagementsystem, welches auf der obersten Ebene L_n zum Einsatz kommt. Über die Metadaten des DBMS plant Calcite die Anfrage auf Basis voreingestellter Optimierungstechniken, selbst wenn diese nicht im verwendeten DBMS implementiert wurden. Lediglich der konkret verwendete SQL-Dialekt wird beibehalten, damit die transformierte Anfrage weiterhin syntaktisch und semantisch korrekt bleibt.

Calcite bietet eine umfangreiche Auswahl an Optimierungsregeln. Dies betrifft einerseits „klassische“ Regeln zum Verschieben von Selektions-, Projektions- und Verbundoperatoren, als auch weiterführende Regeln für Aggregate, Mengen und Unteranfragen sowie zum Pruning. Als initiale Einstellung im Demonstrator sind die Regeln zum Vertauschen von Projektion, Selektion und Verbund sowie zum Entfernen von Teilanfragen, die keine Zwischenergebnisse liefern, aktiviert. Ein Überblick zu allen Regeln ist in Abbildung F.4 im Anhang gegeben.

Schritt 6: Zerlegung komplexer Aggregate

Komplexe Aggregatfunktionen werden meist auf den unteren Schichten nicht unterstützt. Durch die Zerlegung in einfachere Aggregate können Teile der Anfrage ggf. auf den unteren Schichten verarbeitet werden. Die Zerlegung erfolgt intern durch eine Ersetzung der Knoten im Anfragebaum, welcher die jeweilige Aggregatfunktion repräsentiert.

Die Implementierung basiert auf dem in Unterabschnitt 6.1.4 vorgestellten Konzept. Ein umfassendes Beispiel für diesen Transformationsschritt wird in Anhang D.1 am Beispiel der linearen Regression vorgestellt. Auf die genaue Umsetzung der Anfragetransformation wird im nächsten Schritt bzw. im Unterabschnitt 6.5.3 eingegangen.

Schritt 7: Anfragetransformation

Als Kern des Verfahrens wird mittels der Anfragetransformation die Anfrage in mehrere Anfragefragmente aufgeteilt. Diese Fragmente werden derart bestimmt, dass die Daten möglichst nahe an den Datenquellen vorverarbeitet werden. Die dafür entwickelten Konzepte wurden in den Abschnitten 6.1 bis 6.4 bereits ausführlich vorgestellt.

Für die Umsetzung werden ausgehend von den Anfragekapazitäten einzelner Knoten mittels XPath diejenigen Teilbäume bestimmt, welche transformiert werden müssen. Die eigentliche Transformation geschieht durch die Manipulation des DOM[6], welches den Anfragebaum repräsentiert. Weiterführende Details werden in Unterabschnitt 6.5.3 besprochen.

Schritt 8: Anfrageübersetzung für das konkrete System

Jede Teilanfrage wird in einen für die adressierte Datenquelle ausführbaren Befehl übersetzt. Die Verteilung der Anfrage erfolgt auf eine Vielzahl von unterschiedlichen Systemen, die sich nicht nur in der Leistungsfähigkeit voneinander unterscheiden. Vielmehr existieren mehrere Derivate des SQL-Standards, die sich aus den jeweiligen Implementierungen des Datenbanksystems ergeben. Beispielsweise verfügt IBMs Db2-System nicht über die Möglichkeit, Relationen über den natürlichen Verbund zu verknüpfen, wodurch eine explizite Angabe der Verbundattribute erforderlich ist. Zudem nutzen viele der sensornahen Systeme, wie TinyDB, eigene Anfragesprachen, welche sich stark vom SQL-Standard unterscheiden. Um diesen Schritt zu automatisieren, wird der Ausdruck der Relationenalgebra zu einer (SQL-)Anfrage mittels Apache Calcite übersetzt.

Für Spezialsysteme, wie beispielsweise TinyDB, stehen im Anfrageprozessor angepasste Wrapper zur Verfügung. Im Zuge einer Projektarbeit [WH16] im Rahmen des PArADISE-Projekts wurde ein solcher Wrapper zur Abbildung von relationalen SQL-Anfragen auf die TinyDB-Anfragen entwickelt. Zudem können komplexere Programmkonstrukte, welche nicht durch die vorherige Anfragezerlegung optimiert bzw. verteilt werden konnten, als Gesamtblock betrachtet werden. Diese Blöcke können durch die auf dem jeweiligen Knoten vorhandenen Compiler bzw. Interpreter weiterverarbeitet und optimiert werden.

Schritt 9: Anfrageausführung der maximal unterstützten Teilbäume

Die für das jeweilige System erstellte Anfrage wird lokal ausgeführt und als Zwischenergebnis gespeichert. Durch die Anfrageausführung werden die Ergebnistupel aus der Datenbank geladen und in Schritt 11 als neue Basisrelation an die höher gelegenen Knoten weitergeleitet.

Schritt 10: Anonymisierung der Daten

Wie in Unterabschnitt 4.6.2 festgestellt wurde, ist die Anonymisierung von Daten meist mit einem erhöhten Informationsverlust behaftet. Durch die zuvor angewendeten Techniken zum Erreichen der Datensparsamkeit wird bereits ein hohes Datenschutzniveau erreicht, indem ein Großteil der sensiblen Informationen herausgefiltert oder aggregiert wird. Nichtsdestotrotz kann es zur Einhaltung von Datenschutzstandards notwendig sein, dass eine Anonymisierung bzw. Pseudonymisierung der Daten vor Übertragung an den nächsten Knoten erfolgen muss.

[6]Document Object Model

Verfahren zur Anonymisierung wurden im Rahmen von PArADISE durch studentische Projekte, wie den Bachelorarbeiten von Richard Dabels [Dab17] und Eric Klein [Kle20], konzipiert und implementiert.

Schritt 11: Ergebnisintegration

Nach der Auswertung und der ggf. erfolgten Anonymisierung der Teilanfragen der aktuellen Schicht werden die ausgewerteten Teilbäume der Anfrage aus dem Anfragebaum gelöscht und die Zwischenergebnisse durch neue Basisrelationen auf den höheren Knoten ersetzt. Dabei werden die Ergebnisse der Teilanfragen schrittweise zu einem Gesamtergebnis integriert.

Das Ergebnis der zusammengesetzten Teilanfragen ist durch die Transformation weiterhin äquivalent zur Originalanfrage, die auf einem integrierten Datenbestand auf einem globalen Schema hätte gestellt werden können.

Die einzelnen Schritte der Anfragetransformation wurden in Teilen durch bestehende Frameworks umgesetzt. Als Programmiersprache wurde Java genutzt, wobei hier auch auf bestehende Bibliotheken zurückgegriffen wurde. Diese Frameworks und Bibliotheken werden im nachfolgenden Unterabschnitt vorgestellt.

6.5.2 Verwendete Werkzeuge

In diesem Abschnitt wird kurz auf die verwendeten Technologien und Frameworks sowie deren Rolle im entwickelten Demonstrator eingegangen. Dabei wird neben den für die Anfragetransformation notwendigen Werkzeuge mit TreeViz eine Java-Bibliothek zur Visualisierung des Anfragebaumes vorgestellt.

PostgreSQL: Als primäres Testsystem zur Ausführung der Datenbankanfragen wurde *PostgreSQL*[7] in den Versionen 9.6 und 11.3 genutzt. Die Anfragesprache von PostgreSQL orientiert sich sehr am SQL-Standard [Mel16] und bietet darüber hinaus eine umfangreiche Funktionsbibliothek. Für die Kommunikation zwischen PostgreSQL und dem entwickelten Demonstrator werden die von PostgreSQL bereitgestellte JDBC-API sowie Apache Calcite genutzt. Wird für Vorführzwecke der Standalone-Demonstrator (siehe Anhang F) verwendet, wird statt PostgreSQL auf SQLite[8] zurückgegriffen, da dieses DBMS lokal lauffähig ist.

Apache Calcite: *Apache Calcite*[9] [BCH+18] ist ein Java-basiertes Tool zum Parsen, Validieren und Optimieren von SQL-Anfragen. Nach dem Parsen einer Anfrage steht diese als Ausdruck der relationalen Algebra bereit und kann durch Regeln der algebraischen Optimierung transformiert werden. Mittels des sogenannten `HepPrograms` kann eine Menge von heuristischen Regeln angegeben werden, welche zur Optimierung der Anfragen angewendet werden können. Die optimierte Anfrage kann anschließend zurück in SQL transformiert und auf der Datenbank ausgeführt werden.

Calcite dient somit auch als JDBC-Middleware zwischen einem Anwendungsprogramm und einem Datenbankmanagementsystem. Durch die vorgeschaltete Optimierung der Anfrage lassen sich somit Laufzeitverbesserungen erzielen, selbst wenn bestimmte Anfragetransformationen nicht vom konkreten DBMS unterstützt werden.

Neben einfachen Optimierungszielen, wie dem Vertauschen einer Projektion mit einer Selektion (`ProjectFilterTransposeRule`) bzw. einem Verbund (`ProjectJoinTransposeRule`), können verschiedene relationale Operatoren mit Aggregaten (z. B. mittels `AggregateFilterTransposeRule`) oder mit korrelierenden Unteranfragen (z. B. mittels `FilterCorrelateRule`) kommutiert werden. Ebenso ist es mit Apache Calcite möglich, Anfragen mit `IN`, `EXISTS` und skalaren Unteranfragen in Verbundanfragen zu transformieren. Eine Übersicht zu den möglichen Einstellungen ist im Anhang in Abbildung F.4 zu sehen.

Apache Calcite wird für die ersten fünf Phasen (Sichtauflösung bis Optimierung) sowie teilweise für die achte Phase (Anfrageübersetzung für konkretes System) eingesetzt. Diese Phasen sind diejenigen, die auch bei einer normalen Anfrageausführung durchlaufen werden.

[7] PostgreSQL: `https://www.postgresql.org/`, zuletzt aufgerufen am 05.01.2022
[8] SQLite: `https://www.sqlite.org/index.html`, zuletzt aufgerufen am 05.01.2022
[9] Apache Calcite: `https://calcite.apache.org/`, zuletzt aufgerufen am 05.01.2022

Simple: Bei *Simple*[10] handelt es sich um eine Java-Bibliothek zur Persistenzmachung von Java-Objekten. In der Implementierung des Algorithmus zur Anfragetransformation werden die Operatoren der relationalen Algebra als *Plain Old Java Objects* (POJOs) repräsentiert. Durch Simple werden diese anschließend als XML-Dateien gespeichert.

Auf Basis von XPath erfolgt in diesen Dateien die Selektion nach bestimmten Operatoren unter Beachtung der Vorbedingungen und Invarianten der Transformationsregeln. Durch Manipulation des Document Object Models erfolgt die eigentliche Anfragemanipulation. Dazu werden einzelne Operatoren gemäß den Transformationen in den XML-Dateien verschoben bzw. neu eingefügt. Nach der Transformation erfolgt die Rückübersetzung der XML-Dateien mittels Simple zunächst in die einzelnen POJOs, bevor durch Apache Calcite schlussendlich eine ausführbare SQL-Anfrage erzeugt wird.

TreeViz *TreeViz*[11] ist eine von Werner Randelshofer entwickelte Java-Bibliothek zur Visualisierung von großen Baumstrukturen. Als mögliche Visualisierungen stehen hyperbolische Bäume, kreisförmige und rechteckige Baumkarten sowie verschiedene strahlenförmige Visualisierungen zur Verfügung.

Für die eigene Visualisierung der SQL-Anfrage wurde der sogenannte *Iceray Tree* verwendet. In dieser Visualisierungsform wird der durch *Simple* erzeugte XML-Baum von links (Wurzelknoten) nach rechts (Blattknoten) aufgebaut. Während der Wurzelknoten eine Höhe von 100 % besitzt, werden alle Kindknoten, entsprechend ihrer Höhe im Baum, kleiner dargestellt. Der Name des Knotens, in diesem Fall der Name des jeweiligen relationalen Operators, wird direkt innerhalb der Visualisierung dargestellt. Weitere Informationen, wie die komplette Pfadangabe, die Anzahl der Kindknoten und die hinterlegten Parameter, können einem Knoten entnommen werden, indem der Mauszeiger über den jeweiligen Knoten bewegt wird.

Um eine übersichtlichere Darstellung zu erreichen, ist die Ansicht zweigeteilt: Nachdem ein Knoten im Baum durch einen Linksklick ausgewählt wurde, erscheinen dessen Kindknoten in einer separaten Ansicht. Innerhalb dieser Darstellung lässt sich anschließend tiefer in den Baum hinein navigieren, bis der gewünschte Knoten erreicht ist. Zusätzlich zur gesteigerten Übersichtlichkeit können durch diese Selektionsmöglichkeit auch einzelne Teilbäume der Anfrage ausgewählt werden, damit die Transformationsregeln nur auf diese angewendet werden. Auf die Implementierung dieser Regeln wird im nachfolgenden Unterabschnitt eingegangen.

6.5.3 Umsetzung der Anfragetransformation

Als erster Schritt in Vorbereitung der Anfragetransformation wird eine gegebene SQL-Anfrage `sql` in ein XML-Dokument überführt und dabei mittels des `HepProgramBuilders` voroptimiert. Die Funktionalitäten werden dabei in der Hilfsklasse `Sql2Ra` gekapselt. Der grobe Ablauf ist in Quellcodeausschnitt 6.1 skizziert.

```
1 RelNode r = Sql2Ra.trans(sql, builder);
2 Relation rel = Sql2Ra.analyzePlan(r);
3 Document docTest = Sql2RaTest.generateDomFromPojo(rel);
```

Quellcodeausschnitt 6.1: Java-Code zur Erzeugung der XML-Repräsentation einer SQL-Anfrage

Die Übersetzung einer SQL-Anfrage erfolgt zunächst in Calcite-Notation (Klasse `RelNode`; Zeile 1). Im `builder` werden dafür die in Unterabschnitt 6.5.2 aufgeführten Anfrageoptimierungen mittels Apache Calcite durchgeführt.

Anschließend erfolgt die Überführung in die eigene POJO-Notation (Klasse `Relation`; Zeile 2). Der Quellcodeausschnitt 6.2 zeigt ein Segment der Funktion `analyzePlan(r)`, durch die ein natürlicher Verbundoperator `r` in Calcite-Notation in die einfache Repräsentation der Klasse `NaturalJoin` überführt wird. Durch die rekursiven Aufrufe (Zeile 3 und 5) lässt sich die gesamte Anfrage, beginnend bei der Wurzel, schrittweise überführen.

[10] Simple XML Serialization: `http://simple.sourceforge.net/`, zuletzt aufgerufen am 05.01.2022
[11] TreeViz: `http://www.randelshofer.ch/treeviz/`, zuletzt aufgerufen am 05.01.2022

```
List<RelNode> childs = r.getInputs();
RelNode left = childs.get(0);
Relation r1 = analyzePlan(left);
RelNode right = childs.get(1);
Relation r2 = analyzePlan(right);
myOp = new NaturalJoin(r1, r2);
```

Quellcodeausschnitt 6.2: Java-Code zur Konvertierung von Calcite-Objekten zu einfachen POJOs am Beispiel eines natürlichen Verbunds.

Abschließend wird daraus die XML-Repräsentation (Klasse `Document`; Zeile 3) erzeugt. Dazu wird intern die Klasse `Serializer` aus dem Simple-Framework und der in Java verfügbare `DocumentBuilder` genutzt. Quellcodeausschnitt 6.3 zeigt die dafür zuständige Hilfsmethode `generateDomFromPojo`.

```
public Document generateDomFromPojo(Relation input) throws Exception {
  Serializer serializer = new Persister();
  File file = new File("xml_before.xml");
  DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
  DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
  Document document = dBuilder.parse(file);
  return document;
}
```

Quellcodeausschnitt 6.3: Java-Code zur Konvertierung der POJO-Repräsentation der Anfrage in die dazugehörige XML-Repräsentation.

Die konkrete Anfragetransformation wurde für jede der in Anhang C aufgeführten Transformationsregeln als eine Methode in Java programmiert. Quellcodeausschnitt 6.4 enthält dafür als Beispiel die Transformation für die Regel `L01`, durch welche ein zusätzliches Kleiner-gleich-Prädikat für Attribut-Konstanten-Vergleiche in die Anfrage eingefügt wird. Die nachfolgenden Zeilenangaben zur näheren Erklärung der allgemeinen Schritte beziehen sich auf dieses Beispiel.

```
/**
 * Anfragetransformation L01 [Kleiner zu Kleiner-gleich (Attribut-Konstante)]
 * @param input Die originale Anfrage bzw. das originale Anfragefragment
 * @return Die transformierte Anfrage bzw. das transformierte Anfragefragment
 * @throws Exception
 */
public Document rewriteL01(Document input, String prefix) throws Exception{
  String xPath = "//selection[predicate/@op=\"<\"]";
  if(!(prefix == null || prefix.equals("")))
    xPath = prefix + xPath;
  NodeList possibleNodes = findOperatorV2(input, xPath);
  for(int i=0; i<possibleNodes.getLength(); i++) {
    Node node = possibleNodes.item(i);
    // Selektionswerte kopieren
    String leftValue = "";
    String rightValue = "";
    String notValue = "";

    NodeList childs = node.getChildNodes();
    for(int j=0; j<childs.getLength(); j++) {
      Node node2 = childs.item(j);
```

```
22        if(node2.getNodeName().equals("predicate")) {
23          leftValue = node2.getAttributes().getNamedItem("leftSide").
            getTextContent();
24          rightValue = node2.getAttributes().getNamedItem("rightSide").
            getTextContent();
25          notValue = node2.getAttributes().getNamedItem("not").getTextContent();
26          node2.getAttributes().getNamedItem("op").setTextContent("<=");
27        }
28      }
29      //Neuen Knoten erzeugen
30      Element newSelection = input.createElement("selection");
31      Element newPredicate = input.createElement("predicate");
32      newPredicate.setAttribute("class", "dev.ra.predicate.BinaryPredicate");
33      newPredicate.setAttribute("leftSide", leftValue);
34      newPredicate.setAttribute("rightSide", rightValue);
35      newPredicate.setAttribute("op", "<");
36      newPredicate.setAttribute("not", notValue);
37      newSelection.appendChild(newPredicate);
38      node.getParentNode().appendChild(newSelection);
39
40      // Aktuellen Knoten aus alter Position entfernen und an neuer Position
        einfügen
41      node.getParentNode().removeChild(node);
42      newSelection.appendChild(node);
43    }
44    return cleanUp(input);
45  }
```

Quellcodeausschnitt 6.4: Kompletter Java-Code für die Anfragetransformation mittels Regel L01.

Als Eingabe erhält jede Methode die XML-Repräsentation der Anfrage in Form eines `Document`-Objektes (Zeile 7). Optional kann in Form eines XPath-Ausdruckes ein Präfix übergeben werden, welcher angibt, auf welchen Teilen des Dokumentes die Anfragetransformation ausgeführt werden soll. Wird dieser Präfix nicht angegeben oder enthält eine leere Zeichenkette (Zeile 9), so wird die Transformation auf die gesamte Anfrage angewendet.

Ausgehend von dem XML-Dokument bzw. -Fragment werden alle Teilbäume ermittelt (Zeile 11), welche durch die Vorbedingungen der Anfragetransformation betroffen sind. Diese werden ebenfalls als XPath-Ausdrücke (Zeile 8) angegeben.

Beispiel: Repräsentation der Vorbedingungen als XPath-Ausdruck in Regel L01

In Quellcodeausschnitt 6.4 wird die Vorbedingung als XPath-Ausdruck der Form

```
1 String xPath = "//selection[predicate/@op=\"<\"]";
```

angegeben. Der Ausdruck durchsucht die Anfrage nach allen (`//`) Selektionsbedingungen (`selection`), die als Vergleichsprädikat ein Kleiner-als verwenden (`[predicate/@op=\"<\"]`).

Anschließend erfolgt die Modifikation der bestehenden Knoten im XML-Dokument (Zeile 14 bis 28). Dabei werden die Werte einzelner Knoten ersetzt (Zeile 26) und ggf. weitere Werte für den nächsten Schritt der Anfragetransformation kopiert (Zeile 23 bis 25). Bei der Erzeugung neuer Knoten (Zeile 29 bis 38), sofern dies für die Anfragetransformation notwendig ist, werden die kopierten Werte eingesetzt. Das neu erzeugte XML-Fragment wird im Anschluss statt dem alten Fragment im Anfragebaum (Zeile 40 bis 42) eingesetzt. Dies wird für jedes betroffene Fragment wiederholt.

Das abschließende `cleanUp` des XML-Dokumentes dient lediglich zur Entfernung leerer Textknoten, die aus technischen Gründen unwillentlich in die interne Java-Repräsentation eingefügt werden. Dies hat zwar keine Aus-

wirkung auf die resultierende SQL-Anfrage, ohne das Entfernen der Knoten schlagen jedoch die intern verwendeten JUnit-Tests zum Überprüfen der Äquivalenz der Anfragen fehl.

Für jede Anfragetransformation wurde jeweils ein JUnit-Test entworfen, der anhand des laufenden Beispiels aus Unterabschnitt 2.2.1 die Äquivalenz der Transformationen überprüft. Die Testsuite befindet sich zusammen mit dem gesamten Quellcode für alle Regeln im Begleitmaterial (siehe Anhang G). Zudem sind die Testfälle über die grafische Oberfläche aufrufbar, die im nachfolgenden Abschnitt beschrieben wird.

6.5.4 Grafische Visualisierung der Anfragetransformation

Losgelöst vom eigentlichen Anfrageprozessor wurde ein Demonstrator entwickelt, welcher den Transformationsprozess der Anfrage nachvollziehbar visualisiert. Die grafische Oberfläche des Demonstrators ist in Abbildung 6.15 dargestellt.

Über die Menüleiste lassen sich SQL-Anfragen laden und speichern sowie die in Anhang C aufgeführten Beispielanfragen für die einzelnen Transformationsregeln laden. Der Demonstrator ist mit einer SQLite-Datenbank gekoppelt, welche das durchgängige Beispiel der Amarok-Datenbank aus Unterabschnitt 2.2.1 vollständig enthält. Die einzelnen Teilkomponenten werden in den folgenden Absätzen näher erläutert.

Eingabe: Im oberen linken Fenster kann eine SQL-Anfrage formuliert werden, welche mit einem Klick auf die Schaltfläche `[SQL ⇒ RA]` in einen Ausdruck der relationalen Algebra umgeformt wird. Über den Menüeintrag `Testfälle` können zudem bestehende, vorformulierte Anfragen für alle implementierten Transformationsregeln in dieses Eingabefenster geladen werden.

Originalanfrage in der Relationenalgebra: Das linke untere Fenster beinhaltet den zur SQL-Anfrage gehörenden Ausdruck der Relationenalgebra, welcher als XML-Dokument dargestellt wird. Zur besseren Lesbarkeit werden die einzelnen Elemente des XML-Dokumentes eingerückt dargestellt.

Transformationsregeln: Im oberen mittigen Fenster befinden sich Schaltflächen zur manuellen Ausführung der Transformationsregeln. Diese werden gruppiert nach den Regeln für die `Klassische Optimierung`, sowie Anfragen mit `Linearen Abhängigkeiten` und `Aggregatfunktionen`. Ein Klick auf eine Regel wendet die Transformation auf den ausgewählten Teilbaum (siehe unten) einmalig an.

Transformierte Anfrage in der Relationenalgebra: Das untere mittlere Fenster zeigt die Anfrage nach Ausführung der Transformationsregeln. Wurde bisher noch keine Regel angewandt, so ist die Darstellung als XML-Dokument äquivalent zur Originalanfrage.

Konfiguration der Ebenen: Im oberen rechten Fenster kann die Konfiguration der einzelnen Ebenen vorgenommen werden. Standardmäßig sind zwei Ebenen aktiv. Durch einen Klick auf die Schaltflächen `[Neue Ebene]` und `[X]` können neue Ebenen hinzugefügt bzw. bestehende Ebenen entfernt werden. Die Schaltflächen `[△]` und `[▽]` dienen zum Vertauschen der Ebenen.

Die Schaltfläche `[...]` dient zur Konfiguration der Anfragekapazität auf der jeweiligen Ebene. Abbildung 6.16 zeigt die verfügbaren und initial eingestellten relationalen Operatoren und Vergleichsprädikate. Diese unterteilen sich weiter in Selektionsprädikate, Gruppierungen, einfache und komplexe Aggregate sowie weitere Operatoren der relationalen Algebra. Zu den letzteren Operatoren zählen neben Projektion und Verbund auch Anfragen mit Rekursion und Fensterfunktionen, wenngleich diese noch nicht unterstützt werden.

Visualisierung des Anfrageplans: Das untere rechte Fenster enthält die Visualisierung der transformierten Anfrage als sogenannten `IcerayTree`. Diese Visualisierungsform ermöglicht die Darstellung größerer Baumstrukturen [Ran13]. Wird ein Element im Baum angeklickt, so wird der darunterliegende Teilbaum rechts daneben

Abbildung 6.15: Übersicht über die grafische Oberfläche des Optimierer-Werkzeuges

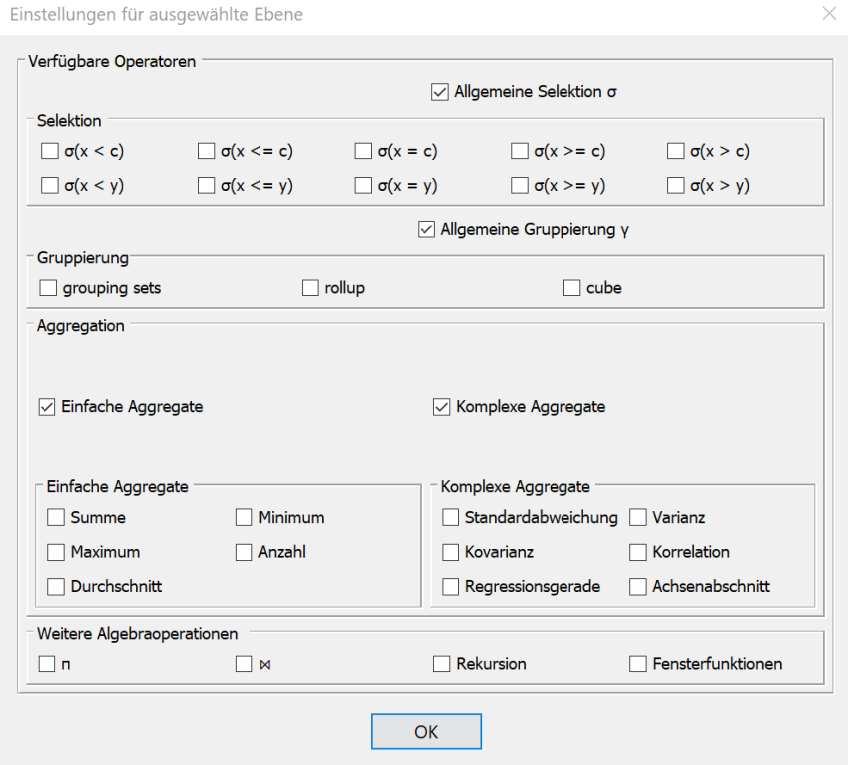


Abbildung 6.16: Einstellungen für die initial verfügbaren Operatoren

vergrößert dargestellt. Dies ermöglicht einerseits eine vereinfachte Exploration der visualisierten Anfrage, andererseits lassen sich die Transformationsregeln somit auf dem ausgewählten Teilbaum anstatt der gesamten Anfrage ausführen. Dadurch können gezielt Teile der Anfrage transformiert werden.

Dem Begleitmaterial (siehe Anhang G) wurde ein Video beigefügt. In diesem Video wird die Verwendung der grafischen Oberfläche anhand einer Anfragetransformation auf einer Beispielanfrage gezeigt.

6.6 Testszenario

In diesem Abschnitt werden die Testergebnisse für Laufzeitmessungen der Anfragetransformationen vorgestellt. Die Evaluation der entwickelten Anfragetransformationen erfolgte anhand der in Abschnitt 4.5.1 vorgestellten Testszenarien. In Unterabschnitt 6.6.1 wird auf die Methodik zum Messen der Laufzeitunterschiede eingegangen. Unterabschnitt 6.6.2 zeigt die Messergebnisse anhand einzelner ausgewählter Beispielanfragen. Anschließend wird in Unterabschnitt 6.6.3 der Einfluss verschiedener Parameter, wie die Rechenleistung einzelner Knoten und die Selektivität von Prädikaten, auf die Laufzeit untersucht. Abschließend werden in Unterabschnitt 6.6.4 Empfehlungen zur Anwendung des Regelsystems in konkreten Systemumgebungen gegeben.

6.6.1 Methodik

Für jedes Testszenario und jede Transformationsregel wurden jeweils zwei Anfragen entworfen: Eine Anfrage vor Anwendung der Transformation und eine Anfrage danach. Die Anfragen wurden als Minimalbeispiele gestal-

tet; sie enthalten neben den notwendigen Basisrelationen nur die benötigten relationalen Operatoren, welche durch die Transformationsregel vorgegeben sind.

Für Anfragen mit den Vergleichsoperatoren Kleiner-als, Kleiner-gleich, Größer-gleich und Größer-als für Attribut-Konstanten-Selektionen wurde zunächst der Medianwert für das zu testende Attribut bestimmt. Dieser wurde als Konstante für die jeweiligen Selektionen verwendet. Dadurch wurde sichergestellt, dass durch die Selektionsprädikate jeweils die Hälfte der Tupel ausgewählt wurden. Eine genauere Untersuchung des Einflusses der Selektivität wird in Unterabschnitt 6.6.3 beschrieben.

Jede Anfrage wurde zehnfach ausgeführt. Es wurde jeweils der Zeitstempel in Nanosekunden direkt vor und nach Ausführung der Anfrage ermittelt, die Differenz gebildet und auf Millisekunden zurückgerechnet. Von den erhobenen Messwerten wurde pro Anfrage der Mittelwert gebildet. Ausgehend von den mittleren Laufzeiten der unveränderten ($t_{unverändert}$) und der transformierten Anfrage ($t_{transformiert}$) wurde der prozentuale Laufzeitunterschied bestimmt:

$$Laufzeitunterschied = \frac{t_{transformiert}}{t_{unverändert}} - 1. \tag{6.45}$$

In einem ersten Versuch wurde die integrierte Zeitmessung von *pgAdmin*[12], der grafischen Oberfläche zu PostgreSQL, genutzt. Durch eine betriebssystembedingte Ungenauigkeit in der Java-API, auf die *pgAdmin* aufbaut, wurde jedoch nicht die Methode `System.currentTimeMillis` zur Zeitmessung genutzt. Der Fehler sorgte für zu ungenaue Messwerte, da nach mehrmaligen Aufrufen der Methode nur Werte zurückgegeben wurden, die ein Vielfaches von dem Wert `10` waren. Stattdessen wurde auf die Methode `System.nanoTime` ausgewichen, wobei die Laufzeit nach erfolgter Zeitmessung durch den Faktor 10^6 geteilt wurde.

Aufbau der SQL-Anfragen

Für die Evaluation werden pro Transformationsregel zwei SQL-Anfragen angegeben; jeweils eine Anfrage vor ($-$) und eine Anfrage nach Anwendung ($+$) der jeweiligen Regel. Zur Simulation einer IoT-Umgebung mit mehreren Knoten werden für die SQL-Anweisungen Common Table Expressions (Kurzform: CTEs) genutzt. Im SQL-Standard entsprechen die CTEs der `WITH`-Klausel. Durch den Einsatz von CTEs werden ungewollte Optimierungen zwischen den einzelnen Ebenen verhindert, während innerhalb einer CTE Optimierungen weiterhin möglich sind (siehe Abbildung 6.17). Die Ergebnisse bezüglich der Laufzeit werden im nächsten Unterabschnitt präsentiert.

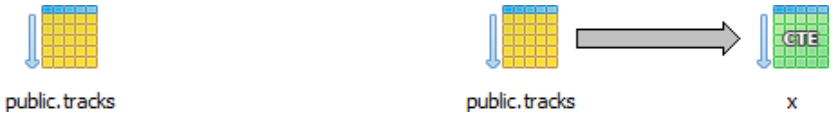


Abbildung 6.17: In der Originalanfrage (links) wird lediglich einmal die komplette Basisrelation für die Selektion durchgegangen. Die transformierte Anfrage (rechts) verwendet zusätzlich eine Selektion auf der CTE, welche als Ergebnis der ersten Selektion entsteht.

Aufbau der Testumgebung

Die Messungen wurden auf einer PostgreSQL-Datenbank in der Version 9.6 durchgeführt. Als Betriebssystem wurde CentOS in der Version 7.4 eingesetzt. Der Server wurde als virtuelle Maschine aufgesetzt und verfügt über eine Vier-Kern-CPU (Intel Xeon E312xx (Sandy Bridge)) mit einer Taktung von 2000 MHz und 4 MB Cache. Als Arbeitsspeicher stehen 8 GB RAM zur Verfügung.

[12] `https://www.pgadmin.org/`, zuletzt aufgerufen am 05.01.2022

Da der Server in Vergleich zu Sensoren und FPGAs eine sehr hohe Leistung hat, spiegeln die Messwerte nicht die Laufzeit in einer realen IoT-Umgebung wieder. Das Ziel dieser Messreihe ist es vielmehr, die zusätzliche bzw. eingesparte Laufzeit zu ermitteln, welche durch die Anwendung einzelner Regeln hervorgeht.

Für die Tests wurde zunächst das Datenbankmanagementsystem PostgreSQL in der Version 9.6 eingesetzt. Zudem wurden die Tests für alle Testszenarien mit der aktuellen Version von PostgreSQL, v11.3, wiederholt. Dabei traten Schwankungen der Messwerte in beide Richtungen auf. Diese lassen sich auf die Verbesserungen am Datenbanksystem zurückführen[13].

So wurden ab v10.0 signifikante Verbesserungen hinsichtlich der Performance des DBMS realisiert. Beispielsweise wurde in v10.0 der Anfrageoptimierer um Multi-Column-Statistiken erweitert, wodurch die Anzahl an verschiedenen Werten bzw. Tupeln besser berechnet werden kann. v10.8 verbesserte die Anfrageplanung bei Selektion auf Gleichheit und beim Überprüfen von Fremdschlüsselbedingungen. In Version 11.0 wurde die Prüfung mehrerer Selektionsbedingungen parallelisiert. Zudem wurde die geschätzte Selektivität für Größer-gleich- und Kleiner-gleich-Selektionen verbessert, indem diese nicht mehr identisch zu Größer-als bzw. Kleiner-als sind.

6.6.2 Beispielanfragen

Um die Vielfalt der Regelmenge zu zeigen, werden an dieser Stelle je eine Regel der klassischen Anfrageoptimierung (Klasse K), eine Regel mit linear-arithmetischen Bedingungen (Klasse L) und zwei unterschiedliche Transformationen mit Aggregatfunktionen (Klasse A) vorgestellt. Eine vollständige Auflistung aller Regeln sowie komplexere Beispiele sind in Anhang C bzw. Anhang D zu finden.

Alle Testergebnisse für das Anwendungsszenario *Amarok* sind in Anhang E aufgeführt. Die Ergebnisse der weiteren Testszenarien *TPC-H* und *MuSAMA* sowie die Ergebnisse für weitere Datenbankmanagementsysteme sind im Begleitmaterial (siehe Anhang G) zu finden. Da die Messwerte nicht signifikant von denen des Amarok-Szenarios abweichen, wird an dieser Stelle auf eine detaillierte Betrachtung verzichtet.

In dieser Arbeit wurden die zum Zeitpunkt des Schreibens verfügbaren Regeln evaluiert. Für nachträglich hinzugefügte Regeln und deren Tests sei auf die Webseite des Projektes[14] verwiesen. Die folgende Tabelle fasst die Ergebnisse der Evaluation für die Regeln `K09` (Kommutativität von Selektion und Projektion), `L03` (Gleich → Größer-gleich), `A10` (Maximum-gleich → Durchschnitt-kleiner-gleich) und `A27` (Alle-kleiner → Maximum-kleiner) zusammen.

Regel	**K09**		**L03**		**A10**		**A27**	
Messung #1 in ms	24	21	6	18	12	22	17	9
Messung #2 in ms	24	21	7	13	11	22	25	9
Messung #3 in ms	22	22	6	12	11	22	19	9
Messung #4 in ms	21	22	7	12	12	22	17	9
Messung #5 in ms	21	22	6	11	12	22	17	9
Messung #6 in ms	21	22	7	11	12	22	17	9
Messung #7 in ms	21	22	6	11	15	22	17	9
Messung #8 in ms	21	34	6	11	13	22	17	9
Messung #9 in ms	25	27	7	11	12	22	17	9
Messung #10 in ms	34	23	6	10	12	22	17	9
Durchschnitt in ms	23,4	23,6	6,4	12,0	12,2	22,0	18,0	9,0
Standardabweichung in ms	3,83	3,83	0,49	2,14	1,08	0,0	2,41	0,0
Laufzeitunterschied in %	0,85		87,50		80,33		-50,00	

Tabelle 6.1: Laufzeitunterschiede für die ausgewählten Transformationsregeln $K09$, $L03$, $A10$ und $A27$

[13] Eine detaillierte Liste der Änderungen ist unter `https://www.postgresql.org/docs/release/` zu finden. Die Seite wurde zuletzt am 05.01.2022 aufgerufen.

[14] `https://dbis.informatik.uni-rostock.de/ueberblick/`, zuletzt aufgerufen am 05.01.2022

Wie anhand des Laufzeitunterschieds zu erkennen ist, hat die Regel K09 keine wesentlichen Auswirkungen auf die Laufzeit – der geringe Unterschied von 0,85% hätte aufgrund der hohen Schwankungen bei der Messung auch zugunsten der transformierten Anfrage ausfallen können. Bei denjenigen Anfragetransformationen (L03 und A10), die zu einem doppelten Auslesen des auf der unteren Ebene berechneten Zwischenergebnisses führten, entstand ein sehr hoher – aber erwarteter – Laufzeitunterschied. Diese Transformationen sollten, sofern möglich, vermieden werden. Allerdings können diese Arten von Transformationen unter bestimmten Randbedingungen (siehe Unterabschnitt 6.6.3) zu besseren Laufzeiten führen.

Für die Anfragen mit Aggregatfunktionen (Klasse A) wurden CTEs zur Vermeidung unbeabsichtigter Optimierungen eingesetzt. Im Falle der Anfragen mit linear-arithmetischen Bedingungen (Klasse L) wurden die CTEs nur für die transformierten Anfragen genutzt, um die zusätzlichen Filterbedingungen zu realisieren. Durch die zusätzliche Selektion in jeder transformierten Anfrage steigt die Laufzeit meist leicht bzw. bis maximal 100% an. Eine Ausnahme bildet Regel L07, welche aus bisher ungeklärter bzw. unerklärlicher Ursache einen extremen Laufzeitanstieg verzeichnete. Dieser Fehler trat jedoch nur mit der Version 9.6 von PostgreSQL auf; andere Datenbankmanagementsysteme zeigten hier keine Laufzeitunterschiede im Vergleich zu den anderen Transformationen aus der Regelklasse L.

Die meisten Anfragen verzeichneten einen Anstieg von ca. 50% bis 80%. Dies entspricht der vermuteten Erwartung, da die Anfragen in den meisten Fällen ca. 20% bis 50% der Tupel herausselektierten, sodass mindestens die Hälfte der Daten erneut überprüft werden mussten. In Bezug auf IoT-Anwendungsfälle wird die Ausführungszeit des Systems nicht beeinflusst, da die Daten von der erzeugenden Quelle, i. d. R. einem Sensorknoten, beim Weiterleiten einmal komplett gelesen werden, unabhängig davon, ob ein Filter hinzugefügt wurde oder nicht.

Als Überraschung stellten sich die Transformationsregeln A27 bis A34 heraus: Wegen der Ersetzung des Allquantors durch eine bedingte Aggregatfunktion verringert sich die Laufzeit um einen nicht unbeachtlichen Anteil – im Falle von Regel A27 um 50%. Sofern Aggregatfunktionen auf den unteren Ebenen unterstützt werden, wie es bei FPGAs der Fall ist, helfen die Transformationsregeln dabei, die Laufzeit des Gesamtsystems drastisch zu reduzieren.

Im nächsten Unterabschnitt wird die Laufzeit der Transformationsregeln in einer simulierten Umgebung evaluiert. Während die Datensparsamkeit – unter Beibehaltung des exakten Ergebnisses – durch die Anwendungen der Regeln erreicht werden kann, bleibt die Frage offen, welche Auswirkungen dies auf die Laufzeit in IoT-Umgebungen hat.

6.6.3 Parametervergleiche

In diesem Unterabschnitt wird untersucht, welchen Einfluss die Anzahl der Knoten auf den unteren Ebenen, die Rechenleistung und die Selektivität auf die Laufzeiten der Original- sowie der transformierten Anfrage haben. Als Beispielanfrage betrachten wir im Folgenden eine Folge von relationalen Operatoren auf einer Menge von Sensorknoten: Jeder dieser Knoten "produziert" eine Million Tupel. Die Originalanfrage sammelt die Daten von den einzelnen Knoten n_i, mit $i \in 1 \dots m$ ein (mengentheoretische Vereinigung $\bigcup$), führt eine Selektion σ auf diesen Daten aus, bevor abschließend ein Selbstverbund $\bowtie$ ausgeführt wird. Bei der transformierten Anfrage wird zusätzlich eine Selektion σ^* auf den Sensorknoten ausgeführt, um die Menge der übertragenden Daten zu reduzieren. Abbildung 6.18 visualisiert die beiden Anfragen in der bekannten Baumstruktur.

Ausgehend von den relationalen Operatoren ergeben sich folgende Laufzeitkomplexitäten:

- Originalanfrage: Selbstverbund + Selektion = $O(n^2) + O(n)$,
- Transformierte Anfrage: Selbstverbund + Selektion = $O(n^2) + O(n)$ + Selektion auf unteren Knoten: $O(n)$,

wobei n die Anzahl der Tupel auf den jeweiligen Knoten ist. Die Kosten für die Vereinigung werden vernachlässigt, da diese in beiden Anfragen auf $O(1)$ betragen.

Die Gesamtlaufzeiten der Anfragen ergeben sich aus den aufsummierten Kosten für die Ausführung jedes Operators. Für die Einzelkosten wird die Anzahl der Tupel mit der Selektivität der Selektionsprädikate multipliziert und durch die Rechenleistung des jeweiligen Knotens, auf dem der Operator ausgeführt wird, dividiert. Für jede

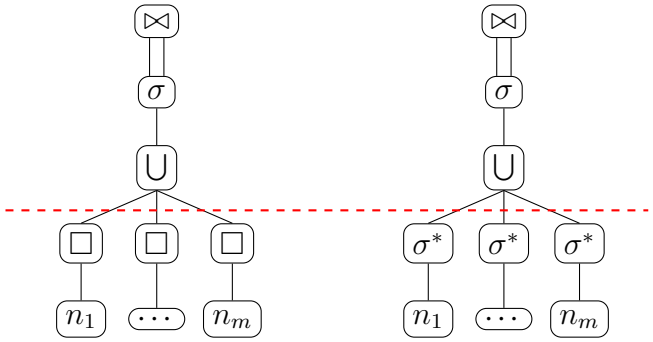


Abbildung 6.18: Visualisierung der Beispielanfragen in Relationenalgebra für die Untersuchung der Auswirkung von Knotenanzahl, Rechenleistung und Selektivität auf die Laufzeit der Anfragen. Links ist die Originalanfrage dargestellt, rechts die transformierte Anfrage.

Untersuchung wird im Folgenden jeweils eine Tabelle mit den jeweiligen Parameterwerten für Knotenanzahl, Rechenleistung und Selektivität angegeben sowie die daraus berechnete Laufzeit für beide Anfragen angeführt. Ohne Variation der Parameterwerte wird nachfolgend von folgenden Standardwerten ausgegangen:

Parameter		**Wert**
Knotenanzahl untere Ebene		5
Rechenleistung	untere Ebene	200, 500 bzw. 1000 MHz
	obere Ebene	2000 MHz
Selektivität	untere Ebene	0,1
	obere Ebene	0,01

Für die Betrachtungen wird, mit Ausnahme der Selektivität auf der oberen Ebene, jeder dieser Parameterwerte variiert, um dessen Auswirkung zu ermitteln. Die Laufzeitangaben in den Diagrammen und Tabellen sind jeweils in Sekunden angegeben.

Anzahl der Knoten auf der unteren Ebene

Mit der steigenden Anzahl an Knoten auf der unteren Ebene steigt auch die Datenmenge, die es zu verarbeiten gilt. Aus diesem Grund steigt die Laufzeit sowohl für die Original- als auch die transformierte Anfrage mit steigender Knotenzahl an. Da die Auswertung des Selektionsprädikates der transformierten Anfrage parallel auf den unteren Knoten ausgeführt werden kann, steigt die Laufzeit in diesem Fall langsamer an. Wird eine bestimmte Knotenanzahl (im Beispiel: 12 Knoten) überschritten, ist die transformierte Anfrage trotz des zusätzlichen Selektionsprädikates schneller als die Originalanfrage. Tabelle 6.2 bzw. Abbildung 6.19 zeigen die Auswirkung der variablen Knotenanzahl von einem bis 20 Knoten mit jeweils 200 MHz Rechenleistung für die unteren Knoten.

Die Steigerung der Rechenleistung der unteren Knoten hat bzgl. der Variation der Knotenzahl lediglich eine konstant bleibende Verringerung auf die Laufzeit der transformierten Anfrage. Wird, wie in Tabelle 6.3 bzw. Abbildung 6.19 zu sehen, die Rechenleistung auf 1000 Mhz verfünffacht, sinkt die Rechenzeit der Beispielanfrage konstant um vier Sekunden ab. Durch die höhere Rechenleistung wird jedoch die Anzahl der benötigten Knoten, um die transformierte Anfrage schneller auszuführen als die Originalanfrage, auf drei Knoten reduziert.

Rechenleistung

Im Folgenden wird untersucht, welchen Einfluss die Veränderung der Rechenleistung sowohl auf der unteren als auch der oberen Ebene auf die Laufzeit der Anfragen hat. Sowohl Selektivität als auch Knotenanzahl bleiben bei dieser Untersuchung konstant. Tabelle 6.4 und Abbildung 6.20 zeigen, dass durch eine lineare Erhöhung der

Anzahl	MHz		Selektivität		Laufzeit	
Knoten untere Ebene	untere Ebene	obere Ebene	untere Ebene	obere Ebene	Original-anfrage	transformierte Anfrage
1	200	2000	0,1	0,01	0,55	5,10
2	200	2000	0,1	0,01	1,20	5,30
3	200	2000	0,1	0,01	1,95	5,60
4	200	2000	0,1	0,01	2,80	6,00
5	200	2000	0,1	0,01	3,75	6,50
6	200	2000	0,1	0,01	4,80	7,10
7	200	2000	0,1	0,01	5,95	7,80
8	200	2000	0,1	0,01	7,20	8,60
9	200	2000	0,1	0,01	8,55	9,50
10	200	2000	0,1	0,01	10,00	10,50
11	200	2000	0,1	0,01	11,55	11,60
12	200	2000	0,1	0,01	13,20	12,80
13	200	2000	0,1	0,01	14,95	14,10
14	200	2000	0,1	0,01	16,80	15,50
15	200	2000	0,1	0,01	18,75	17,00
16	200	2000	0,1	0,01	20,80	18,60
17	200	2000	0,1	0,01	22,95	20,30
18	200	2000	0,1	0,01	25,20	22,10
19	200	2000	0,1	0,01	27,55	24,00
20	200	2000	0,1	0,01	30,00	26,00

Tabelle 6.2: Messwerte für den Einfluss der Knotenanzahl der unteren Ebene bei einer Rechenleistung von 200 MHz. Die orangefarbene Zeile markiert die Knotenanzahl, ab der die transformierte Anfrage schneller ausgeführt wird als die Originalanfrage.

Rechenleistung der Knoten auf der unteren Ebene die Laufzeit der transformierten Anfrage logarithmisch sinkt. Die Laufzeit der Originalanfrage bleibt hiervon unberührt, da auf der unteren Ebene keine Operatoren ausgeführt werden.

Tabelle 6.5 und Abbildung 6.21 zeigen den Einfluss der Rechenleistung des oberen Knotens auf die Laufzeit. Durch die Steigerung der Rechenleistung verringert sich die Laufzeit sowohl der Originalanfrage als auch die der transformierten Anfrage. Der Effekt ist für die Originalanfrage stärker ausgeprägt, sodass mit steigender Rechenleistung des Knotens der oberen Ebene die Anfrage schneller ausgeführt wird als die transformierte Anfrage. Im gezeigten Beispiel ist bei einer Rechenleistung ab 2200 MHz des oberen Knotens die Laufzeit der Originalanfrage kürzer als die der transformierten Anfrage.

Selektivität

Als weiteren Aspekt wird der Einfluss der Selektivität des Prädikates auf den unteren Ebenen auf die Laufzeit untersucht. Tabelle 6.6 und Abbildung 6.22 zeigen, dass eine höhere Selektivität[15] die Laufzeit der transformierten Anfrage reduziert, da weniger Daten an die obere Ebene gesendet und dort verarbeitet werden. Die Laufzeit der Originalanfrage bleibt hingegen konstant, da das eingeschobene Selektionsprädikat der unteren Ebene in der Anfrage nicht vorkommt.

Die gezeigten Vergleiche der Parameter beziehen sich alle auf die eingangs vorgestellte Beispielanfrage. Durch die verschiedenen Arten von Anfragen ergeben sich in anderen Anwendungsszenarien unterschiedliche Laufzeit-

[15] Bei einer höheren Selektivität werden weniger Tupel in die Ausgaberelation übernommen. Der Wert für die Selektivität ist dabei gering (z. B. von 0,01). Eine geringe Selektivität hat entsprechend einen hohen Wert (z. B. von 1,00).

Anzahl Knoten untere Ebene	MHz		Selektivität		Laufzeit	
	untere Ebene	obere Ebene	untere Ebene	obere Ebene	Original-anfrage	transformierte Anfrage
1	1000	2000	0,1	0,01	0,55	1,10
2	1000	2000	0,1	0,01	1,20	1,30
3	1000	2000	0,1	0,01	1,95	1,60
4	1000	2000	0,1	0,01	2,80	2,00
5	1000	2000	0,1	0,01	3,75	2,50
6	1000	2000	0,1	0,01	4,80	3,10
7	1000	2000	0,1	0,01	5,95	3,80
8	1000	2000	0,1	0,01	7,20	4,60
9	1000	2000	0,1	0,01	8,55	5,50
10	1000	2000	0,1	0,01	10,00	6,50
11	1000	2000	0,1	0,01	11,55	7,60
12	1000	2000	0,1	0,01	13,20	8,80
13	1000	2000	0,1	0,01	14,95	10,10
14	1000	2000	0,1	0,01	16,80	11,50
15	1000	2000	0,1	0,01	18,75	12,00
16	1000	2000	0,1	0,01	20,80	14,60
17	1000	2000	0,1	0,01	22,95	16,30
18	1000	2000	0,1	0,01	25,20	18,10
19	1000	2000	0,1	0,01	27,55	20,00
20	1000	2000	0,1	0,01	30,00	22,00

Tabelle 6.3: Messwerte für den Einfluss der Knotenanzahl der unteren Ebene bei einer Rechenleistung von 1000 MHz. Die orangefarbene Zeile markiert die Knotenanzahl, ab der die transformierte Anfrage schneller ausgeführt wird als die Originalanfrage.

komplexitäten. Tabelle 6.7 fast die Änderungen der Laufzeitkomplexität für die einzelnen Transformationsregeln zusammen. Die Komplexitäten ergeben sich aus der Betrachtung der Planoperatoren, die durch den Optimierer von PostgreSQL zur Beantwortung der Anfragen auf dem Amarok-Datensatz ausgewählt wurden. Für die anderen Anwendungsbeispiele ergeben sich leicht abweichende Komplexitätsangaben. Diese sind, zusammen mit ihrer grafischen Visualisierung, im Begleitmaterial (siehe Anhang G) sowie auf der Website zum Projekt[16] zu finden.

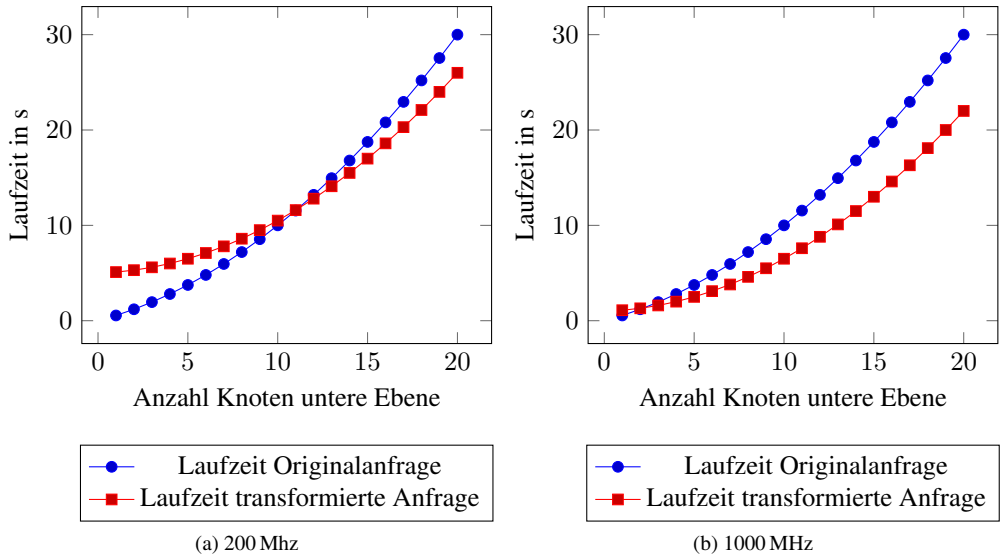
Wie wir in diesem Abschnitt gelernt haben, weisen die Anfragetransformationen nicht immer eine positive Auswirkung auf die Laufzeit des gesamten Informationssystems auf. Obwohl Datenschutz, insbesondere durch die Einhaltung des Prinzips der Datensparsamkeit, bei der Konzeption von Anwendungen einen hohen Stellenwert einnimmt, sollten andere Aspekte, wie eine geringe Latenz, nicht vernachlässigt werden. Dies trifft insbesondere auf Systeme zu, die hohe Anforderungen an die Echtzeitdatenverarbeitung stellen. Der nachfolgende Unterabschnitt geht näher auf diese Problemstellung ein.

6.6.4 Empfehlungen für den Einsatz in konkreten Systemumgebungen

Durch die Laufzeitmessungen und den sich daraus ergebenen Laufzeitunterschieden lassen sich die Regeln grob in drei Kategorien einordnen:

1. <0%: Diese Anfragetransformationen sollten unabhängig von den vorhanden Anfragekapazitäten angewendet werden, da sie neben datenschutzrelevanten Aspekten auch zu einer Verkürzung der Anfragelaufzeit führen.

[16] https://dbis.informatik.uni-rostock.de/vldb2018/, zuletzt aufgerufen am 05.01.2022



(a) 200 Mhz

(b) 1000 MHz

Abbildung 6.19: Laufzeiten der Original- und der transformierten Anfrage unter Variation der Knotenanzahl auf der unteren Ebene. Die Knoten der unteren Ebene verfügen über jeweils 200 MHz bzw. 1000 MHz Rechenleistung, der zentrale Knoten auf der oberen Ebene über 2000 MHz Rechenleistung. Die Selektivität auf der unteren Ebene beträgt 10 %, auf der oberen Ebene 1 %. Ab zwölf bzw. drei Knoten ist die Laufzeit der transformierten Anfrage kürzer als die der Originalanfrage.

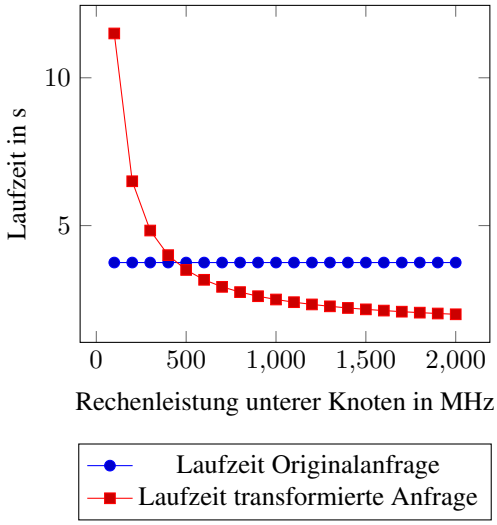


Abbildung 6.20: Laufzeiten der Original- und der transformierten Anfrage unter Variation der Rechenleistung der Knoten auf der unteren Ebene. Die fünf Knoten der unteren Ebene verfügen über jeweils 100 MHz bis 2000 MHz Rechenleistung, der zentrale Knoten auf der oberen Ebene über 2000 MHz Rechenleistung. Die Selektivität auf der unteren Ebene beträgt 10 %, auf der oberen Ebene 1 %. Ab 500 MHz ist die Laufzeit der transformierten Anfrage kürzer als die der Originalanfrage.

Anzahl Knoten untere Ebene	MHz untere Ebene	MHz obere Ebene	Selektivität untere Ebene	Selektivität obere Ebene	Laufzeit Original-anfrage	Laufzeit transformierte Anfrage
5	100	2000	0,1	0,01	3,75	11,50
5	200	2000	0,1	0,01	3,75	6,50
5	300	2000	0,1	0,01	3,75	4,83
5	400	2000	0,1	0,01	3,75	4,00
5	500	2000	0,1	0,01	3,75	3,50
5	600	2000	0,1	0,01	3,75	3,17
5	700	2000	0,1	0,01	3,75	2,93
5	800	2000	0,1	0,01	3,75	2,75
5	900	2000	0,1	0,01	3,75	2,61
5	1000	2000	0,1	0,01	3,75	2,50
5	1100	2000	0,1	0,01	3,75	2,41
5	1200	2000	0,1	0,01	3,75	2,33
5	1300	2000	0,1	0,01	3,75	2,27
5	1400	2000	0,1	0,01	3,75	2,21
5	1500	2000	0,1	0,01	3,75	2,17
5	1600	2000	0,1	0,01	3,75	2,13
5	1700	2000	0,1	0,01	3,75	2,09
5	1800	2000	0,1	0,01	3,75	2,06
5	1900	2000	0,1	0,01	3,75	2,03
5	2000	2000	0,1	0,01	3,75	2,00

Tabelle 6.4: Messwerte für den Einfluss der Rechenleistung der Knoten der unteren Ebene bei einer konstanten Knotenanzahl. Die orangefarbene Zeile markiert die Rechenleistung, ab der die transformierte Anfrage schneller ausgeführt wird als die Originalanfrage.

---

2. `0 - 100%`: Die Regeln in dieser Kategorie sollten trotz ihres negativen Einflusses auf die Laufzeit bevorzugt angewendet werden. Besteht zwischen zwei möglichen Anfragetransformationen keine bekannte Beziehung hinsichtlich des Query Containments, sollte die Transformation mit der geringeren Laufzeiterhöhung gewählt werden.

3. `> 100%`: Transformationen aus dieser Kategorie sollten nur dann angewendet werden, wenn ein sehr hohes Datenschutzniveau erforderlich ist. Die Nachteile der erhöhten Laufzeit sind mit den Vorteilen des besseren Datenschutzniveaus abzuwägen.

Der Grenzwert von `100%` wurde hier nur exemplarisch gewählt – je nach Anwendungsbereich und Echtzeit- sowie Datenschutzanforderungen an das System ist dieser entsprechend anzupassen. Soll das Informationssystem die spezifischen Anforderungen des Nutzers noch mehr beachten, so sollte dieser den Grenzwert selber festlegen bzw. die anzuwendenden Regeln selbst bestimmen – ein durchschnittlicher Nutzer ist wahrscheinlich damit überfordert. Möglich ist auch ein Feedback an das System, falls eine Anfrage zu lange gedauert hat. Dadurch könnten einzelne Regeln mit positiven bzw. negativen Gewichten versehen werden, wodurch sie häufiger bzw. seltener angewendet werden.

Je nach konkreter Systemumgebung mit mehr oder weniger leistungsfähigen Knoten und verschiedener Anfragekapazität können die Laufzeit unterschiedlich ausfallen. Die in Anhang C vorgestellten Beispielanfragen sind als Benchmark für die Systemumgebung gedacht: Sie geben Aufschluss darüber, inwieweit konkrete Transformationen Einfluss auf die Laufzeit einer Anfrage haben. Durch die ermittelten Laufzeitunterschiede kann eine Ordnung zwischen den Regeln festgelegt werden, damit die Ausführung der Anfrage möglichst effizient wird. Diese Art der

Anzahl Knoten untere Ebene	MHz untere Ebene	MHz obere Ebene	Selektivität untere Ebene	Selektivität obere Ebene	Laufzeit Original-anfrage	Laufzeit transformierte Anfrage
5	500	3000	0,1	0,01	2,50	3,00
5	500	2900	0,1	0,01	2,59	3,03
5	500	2800	0,1	0,01	2,68	3,07
5	500	2700	0,1	0,01	2,78	3,11
5	500	2600	0,1	0,01	2,88	3,15
5	500	2500	0,1	0,01	3,00	3,20
5	500	2400	0,1	0,01	3,13	3,25
5	500	2300	0,1	0,01	3,26	3,30
5	500	2200	0,1	0,01	3,41	3,36
5	500	2100	0,1	0,01	3,57	3,43
5	500	2000	0,1	0,01	3,75	3,50
5	500	1900	0,1	0,01	3,95	3,58
5	500	1800	0,1	0,01	4,17	3,67
5	500	1700	0,1	0,01	4,41	3,76
5	500	1600	0,1	0,01	4,69	3,88
5	500	1500	0,1	0,01	5,00	4,00
5	500	1400	0,1	0,01	5,36	4,14
5	500	1300	0,1	0,01	5,77	4,31
5	500	1200	0,1	0,01	6,25	4,50
5	500	1100	0,1	0,01	6,82	4,73

Tabelle 6.5: Messwerte für den Einfluss der Rechenleistung des Knotens der oberen Ebene. Die orangefarbene Zeile markiert die Rechenleistung, ab der die transformierte Anfrage schneller ausgeführt wird als die Originalanfrage.

Entscheidung für oder gegen Datenschutz betrifft nicht nur die Laufzeitunterschiede, sondern berührt auch viele weitere soziale und politische Bereiche:

> **Zitat:** Entwicklung von Kriterien zur Interessenabwägung [WR67]
>
> If privacy is to receive its proper weight on the scales in any process of balancing competing values, what is needed is a structured and rational weighting process, with definite criteria that public and private authorities can apply in comparing the claims for disclosure or surveillance through new devices with the claims to privacy. The following are the basic steps of such a process: measuring the seriousness of the need to conduct surveillance; deciding whether there are alternative methods to meet the need; deciding what degree of reliability will be required of the surveillance instrument; determining whether true consent to surveillance has been given; and measuring the capacity for limitation and control of the surveillance if it is allowed.

Der politische Standpunkt zum Thema Datenschutz wird, wie in Abschnitt 2.4 angerissen, sehr kontrovers diskutiert. Aus der Debatte ist für den konzeptionellen Aspekt zu übernehmen, dass Datenschutz einen hohen Stellenwert einnehmen muss, eine konsequente Umsetzung aber nicht zu Ungunsten weiterer Aspekte, wie der in diesem Abschnitt angerissenen Performanceeinbußen, realisiert werden darf.

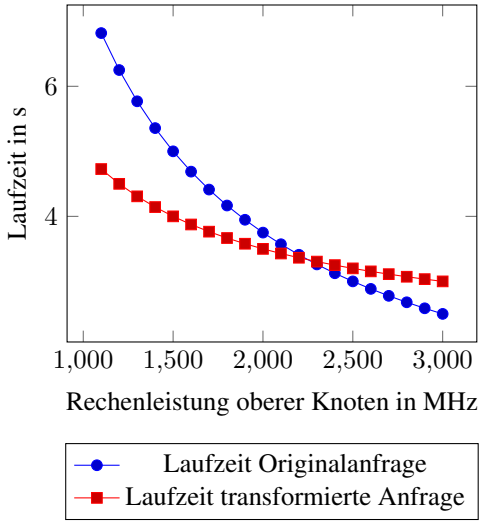


Abbildung 6.21: Laufzeiten der Original- und der transformierten Anfrage unter Variation der Rechenleistung des Knotens auf der oberen Ebene. Die fünf Knoten der unteren Ebene verfügen über jeweils 500 MHz Rechenleistung, der zentrale Knoten auf der oberen Ebene über zwischen 1100 und 3000 MHz Rechenleistung. Die Selektivität auf der unteren Ebene beträgt 10 %, auf der oberen Ebene 1 %. Ab 2200 MHz ist die Laufzeit der Originalanfrage kürzer als die der transformierten Anfrage.

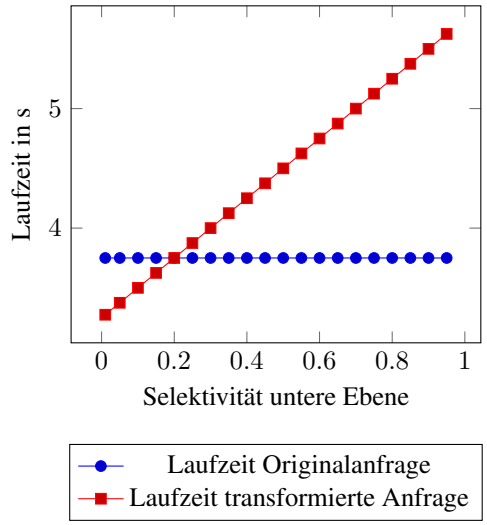


Abbildung 6.22: Laufzeiten der Original- und der transformierten Anfrage unter Variation der Selektivität des Prädikates auf der unteren Ebene. Die fünf Knoten der unteren Ebene verfügen über jeweils 500 MHz Rechenleistung, der zentrale Knoten auf der oberen Ebene über 2000 MHz Rechenleistung. Die Selektivität auf der unteren Ebene beträgt zwischen 1 % und 95 %, auf der oberen Ebene 1 %. Ab einer Selektivität von unter 20 % ist die Laufzeit der transformierten Anfrage kürzer als die der Originalanfrage.

Anzahl	MHz		Selektivität		Laufzeit	
Knoten untere Ebene	untere Ebene	obere Ebene	untere Ebene	obere Ebene	Original-anfrage	transformierte Anfrage
5	500	2000	0,95	0,01	3,75	5,63
5	500	2000	0,90	0,01	3,75	5,50
5	500	2000	0,85	0,01	3,75	5,38
5	500	2000	0,80	0,01	3,75	5,25
5	500	2000	0,75	0,01	3,75	5,13
5	500	2000	0,70	0,01	3,75	5,00
5	500	2000	0,65	0,01	3,75	4,88
5	500	2000	0,60	0,01	3,75	4,75
5	500	2000	0,55	0,01	3,75	4,63
5	500	2000	0,50	0,01	3,75	4,50
5	500	2000	0,45	0,01	3,75	4,38
5	500	2000	0,40	0,01	3,75	4,25
5	500	2000	0,35	0,01	3,75	4,13
5	500	2000	0,30	0,01	3,75	4,00
5	500	2000	0,25	0,01	3,75	3,88
5	500	2000	0,20	0,01	3,75	3,75
5	500	2000	0,15	0,01	3,75	3,63
5	500	2000	0,10	0,01	3,75	3,50
5	500	2000	0,05	0,01	3,75	3,38
5	500	2000	0,01	0,01	3,75	3,28

Tabelle 6.6: Messwerte für den Einfluss des Selektionsprädikates auf der unteren Ebene. Die orangefarbene Zeile markiert den Wert der Selektivität, ab der die transformierte Anfrage schneller ausgeführt wird als die Originalanfrage.

6.7 Zusammenfassung

In diesem Kapitel wurde das Konzept zur Transformation von Anfragen unter Berücksichtigung von Datenschutzansprüchen und Anfragekapazitäten vorgestellt. In Abgrenzung zum Stand der Forschung in Kapitel 5 wurden folgende neue Erkenntnisse gewonnen:

- Mit *Answering Queries using Operators* wurde in Abschnitt 6.1 ein zu *Answering Queries using Views* gegenläufiges Konzept entwickelt. Dieses ermöglicht es, einzelne Teilzeile einer Anfrage auf lokalen Knoten mit begrenzter Anfragekapazität auszuführen, ohne dabei die Vollständigkeit des Anfrageergebnisses zu verletzen.
- In den Abschnitten 6.2 und 6.3 wurde das AQuO-Konzept theoretisch vertieft. Die Schwerpunkte lagen dabei auf Selektionsbedingungen sowie Projektionen und Aggregationen.
- Der konzeptionelle Teil der Arbeit wurde mit der Einführung von *Query Rewriting by Contract* in Abschnitt 6.4 abgeschlossen. Das Konzept ermöglicht die Bestimmung des *Rewriting Supremums*, einer transformierten Anfrage, die die kleinstmögliche Obermenge an Zwischenergebnissen auf den lokalen Rechenknoten erzeugt. Dies ermöglicht es, den Aspekt der Datensparsamkeit technisch zu realisieren, ohne dabei die Funktionalität des dahinterliegenden Informationssystems zu beeinflussen.
- In Abschnitt 6.4 wurden zudem einige Beispieltransformationen für das Rewriting Supremum vorgestellt. Eine vollständige Auflistung der im Rahmen dieser Arbeit entwickelten und umgesetzten Regeln ist in Anhang C zu finden.

Anfrage	Komplexität Originalanfrage	Komplexität transformierte Anfrage
A01–A28	$O(n * log(h))$	$O(n * log(h))$
A29–A30	$O(n * log(h))$	$O(m + n * log(h))$
A31–A47	$O(n * log(h))$	$O(n * log(h))$
A48	$O(n * log(h))$	$O(m + n * log(h))$
A49–A50	$O(n * log(h))$	$O(n * log(h))$
A51	$O(n * log(h))$	$O(m + n * log(h))$
K01	$O(m + n)$	$O(n)$
K02–K03	$O(m + n)$	$O(m + n)$
K04	$O(m + n * log(n))$	$O(m + n * log(h))$
K05–K12	$O(n)$	$O(n)$
K13–K14	$O(n * log(h))$	$O(n * log(h))$
K15–K17	$O(m * n)$	$O(m * n)$
K18–K19	$O(n * log(h))$	$O(n * log(h))$
K20	$O(m + n * log(h))$	$O(m * n * log(h))$
K21	$O(m * n * log(h))$	$O(m + n * log(h))$
L01–L08	$O(n)$	$O(n)$

Tabelle 6.7: Laufzeitkomplexitäten der Anfragen auf dem Amarok-Datensatz. m und n sind dabei die Anzahl der Tupel in den beteiligten Relationen und h ist die Anzahl der Gruppen.

- Analog zu dem Algorithmus zum Erkennen von Quasi-Identifikatoren (siehe Abschnitt 4.4) wurde auch *Query Rewriting by Contract* direkt in den JDBC-Proxy-Treiber von PArADISE (siehe Anhang F) integriert. Dies ermöglicht einen flexiblen Einsatz des Konzepts unabhängig vom eingesetzten Datenbanksystem und der konkreten Anwendung.

Anhand dieser Beispielanfragen wurden die Laufzeit und die Auswirkungen auf die Menge der übertragenen Daten im Vergleich zu unveränderten Anfragen gezeigt. Unter Berücksichtigung der Gegebenheit, dass die Daten zur Übertragung zwischen zwei Knoten ohnehin ausgelesen werden müssen, ist die Laufzeit in vielen Fällen nur unwesentlich höher ausgefallen. In einigen Fällen, insbesondere bei hochselektiven Prädikaten und beim Ersetzen von Allquantoren durch Gruppierungen, ist die Laufzeit sogar gesunken. In konkreten Szenarien, in welchen beispielsweise Smartphone-Anwendungen Daten lokal sammeln und (vor-)auswerten, ist diese Art der Vorverdichtung für die Optimierung der Auslastung von 4/5G- und WLAN-Netzwerken von entscheidender Bedeutung zur Realisierung von Smart Cities bzw. Factories [TDK+17].

6.7.1 Vor- und Nachteile des entwickelten Verfahrens

In diesem Abschnitt gehen wir auf weitere Vor- und Nachteile des entwickelten Verfahrens ein. Den hinlänglich diskutierten Aspekt der Datensparsamkeit klammern wir an dieser Stelle bewusst aus.

Kürzere Kommunikationswege: Ist bereits auf lokaler Ebene bekannt, dass keine Lösung (die Anfrage kein (sinnvolles) Ergebnis liefert) existiert, so kann lokal, ohne Kontakt mit dem Server aufzunehmen, entschieden werden, wie die Anwendung reagieren soll. Ohne die Aufteilung der Anfrage müssten erst alle Daten bis in die Cloud gesendet werden, nur um anschließend festzustellen, dass keine Lösung existiert. Durch das lokale Auswerten entfallen diese Kommunikationswege und der Nutzer erhält eine schnellere Rückmeldung.

Für eine serverseitige Auswertung spricht jedoch das Argument, dass ggf. weitere Informationen von mehreren Quellen vorhanden sind, die evtl. weitere bzw. alternative Informationen enthalten und ggf. eine Lösung für die Anfrage bereitstellen können. Sobald neue Daten in der Cloud verfügbar sind, können diese mit allen verbundenen Endgeräten geteilt werden. Dies widerstrebt jedoch den Datenschutzbestrebungen.

Beschränkte Ressourcen: Ebenso kann die Ressourcenbeschränkung der IoT-Endknoten ein Problem darstellen. Hier kann die Verarbeitung der Daten schnell an Grenzen bzgl. Speicherplatz, Energiebedarf oder Rechenleistung stoßen. Je nach Knoten muss überprüft werden, inwieweit das Vorfiltern bzw. das Aggregieren effizient implementiert ist. Eine vorherige Lastbalancierung (siehe z. B. [Sei17]) kann hier Abhilfe schaffen.

Energiebedarf: Die Fähigkeit zur lokalen Ausführung wird vorrangig durch die unterstützten Operatoren festgelegt, aber auch durch die aktuelle Auslastung des Knotens und – für mobile Geräte – den erforderlichen Energiebedarf. Die Batterielaufzeit mobiler Geräte stellt im IoT-Umfeld eine wesentliche Herausforderung dar. Während die Leistungsfähigkeit mobiler Sensoren und Rechner stetig stark steigt, liegt die Steigerung der Batterielaufzeit bei lediglich 3% pro Jahr [ZL11].

Durch die Verlagerung der Berechnungen hin zu den IoT-Endknoten veranschlagen diese einen erhöhten Energiebedarf. Dieser wird jedoch kompensiert, indem alle Knoten im Edge-Netzwerk weniger miteinander kommunizieren müssen, da weniger Daten übertragen werden. Das daraus resultierende Einsparpotential beim Energiebedarf liegt Studien zufolge bei 30% [SBCD09] bis 95% [CIM+11]. Dies ist insbesondere für mobile Geräte nützlich, da diese an Batterien und Akkumulatoren gebunden sind.

Parallelisierung: Zudem ermöglicht dieses Vorgehen ein höheres Maß an Parallelität. Durch die lokale Vorverarbeitung berechnet einerseits jeder Knoten die von ihm erzeugten Daten und liefert die Daten anschließend an den nächsten Knoten innerhalb der Verarbeitungskette. Da durch die Anfragetransformation mehr Daten parallel verarbeitet werden können und zusätzlich weniger Daten auf der nächsthöheren Ebene anfallen, wird die Anfrageverarbeitung insgesamt beschleunigt.

6.7.2 Ausblick: Anfrageverarbeitung auf moderner, heterogener Hardware

Offen bleibt die Fragestellung, welche Auswirkung die Anfrageverteilung bezüglich der Laufzeit auf realer, heterogener Hardware besitzt. Dabei ist insbesondere darauf zu achten, dass die Knoten der unteren Ebenen zwar eine geringere Rechenleistung besitzen, jedoch die lokal erzeugten Daten parallel verarbeiten können. Zudem müssen die oberen Ebenen durch die Vorverarbeitung weniger Daten analysieren, wodurch die Laufzeit der Anfrage ebenfalls verringert wird.

In zukünftigen Arbeiten sollte weiterhin untersuchten werden, inwieweit weitere Query Containment-Regeln zur Anfragetransformation in *Query Rewriting by Contract* integriert werden können, um weitere – und ggf. bessere – Transformationen zu erzeugen. Zudem sollten weiterführende Techniken der Anfrageoptimierung, wie beispielsweise die Auswirkung der Selektivität unterschiedlicher Anfrageprädikate, untersucht werden, um zu ermitteln, welche Auswirkungen einzelne Selektionsprädikate auf die Laufzeit in verteilten IoT-Umgebungen haben. Dieser Aspekt wurde bereits kurz in [HG20] umrissen.

Kapitel 7

Zusammenfassung und Ausblick

Die Umsetzung von technischen Datenschutzaspekten stellt für viele Entwickler eine große Herausforderung dar. Im Rahmen dieses Kapitels wird über die entwickelten Konzepte der vorliegenden Arbeit, welche Beiträge zur Lösung dieser Herausforderung aufzeigt, resümiert. In Abschnitt 7.1 werden die entwickelten Konzepte und Forschungsergebnisse sowie deren technische Realisierung und Evaluation zusammengefasst. Dabei wird rückblickend auf die Problemstellung, die Zielsetzungen und die Schwerpunkte der Arbeit eingegangen. Abschnitt 7.2 dient abschließend als Impuls für zukünftige Forschungsthemen.

7.1 Zusammenfassung

Die *Problemstellung* der datenschutzkonformen Datenverarbeitung wurde in Kapitel 1.1 definiert. Die zentrale Fragestellung beinhaltet, inwieweit die Ziele des Datenschutzes ohne Anonymisierung und des dabei hergehenden Informationsverlustes realisiert werden können. Als zentrale These wurde vorgeschlagen, dass durch gezielte Anfragetransformationen und -verteilungen auf mehrere Rechenknoten ein hohes Datenschutzniveau erreicht werden kann. Auf Basis dieser These wurden folgende Zielsetzungen (Kapitel 1.1.1) und Schwerpunkte der Arbeit (Kapitel 1.1.2) formuliert:

Zielsetzungen

(i) Entwicklung eines Konzeptes für die datenschutzfreundliche Anfrageverarbeitung, welches die Aspekte der Datensparsamkeit und Datenvermeidung realisiert

(ii) Effiziente Erkennung von schützenswerten Daten in hochdimensionalen Datenbeständen

(iii) Erarbeitung einer Strategie, bei der Daten durch schrittweise Vorberechnungen bereits lokal und ohne Informationsverlust anonymisiert werden

Schwerpunkte

(a) Allgemeines Lösungskonzept (Kapitel 3)

- Modifikation der Anfrageverarbeitung in Datenbanksystemen
- Vorverarbeitung und Transformation von Anfragen
- Anonymisierung der Anfrageergebnisse

(b) Erkennung von Quasi-Identifikatoren (Kapitel 4)

- Charakteristik schützenswerter Daten
- Effiziente Berechnung von Quasi-Identifikatoren in hochdimensionalen Datensätzen
- Verwendung von Quasi-Identifikatoren als Parameter zur Anfragetransformation und Anonymisierung von Analyseergebnissen

(c) Vertikale Anfrageverteilung (Kapitel 6)

- Query Containment als theoretische Grundlage
- Verteilungsstrategien in ressourcenbeschränkten Umgebungen
- Verteilte Berechnung komplexer Aggregat- und Analysefunktionen

Tabelle 7.1 zeigt den Zusammenhang zwischen den gesetzten Zielstellungen und deren Umsetzung in den Schwerpunktkapiteln. Ein Kreuz ($\times$) symbolisiert jeweils eine Zuordnung zwischen Zielsetzung und Schwerpunkt.

Schwerpunkte / Zielsetzungen	(a) Lösungskonzept	(b) Quasi-Identifikatoren	(c) Anfrageverteilung
(i) Konzeption	$\times$	$\times$	$\times$
(ii) Schützenswerte Daten	$\times$	$\times$	
(iii) Vorberechnungen	$\times$		$\times$

Tabelle 7.1: Umsetzung der Zielsetzungen in den einzelnen Schwerpunktkapiteln

7.1.1 Konzeption

Die erste Zielsetzung, die Entwicklung des *Konzepts* zur datenschutzkonformen Datenverarbeitung, wurde primär im Schwerpunkt *Lösungskonzept* beschrieben; die Schwerpunkte *Quasi-Identifikatoren* und *Vertikale Anfrageverteilung* tragen als zwei Kerntechniken zu der Zielsetzung bei. In Kapitel 3 wurde das Konzept zur datenschutzkonformen Anfrageverarbeitung in Form eines zweistufigen Anfrageprozessors vorgestellt. Begleitend zum Überblick wurden

- der Entwurf für eine Datenschutzauszeichnungssprache, die Privacy Policy for Smart Environments (PP4SE) (siehe auch [GH15a] und Anhang B.1),
- die algorithmische Umsetzung von Data Slicing in relationalen Datenbanken (siehe auch [GH15c]) und
- mehrere Anpassungen der Kullback-Leibler-Divergenz für die Messung des Informationsverlustes bei Anonymisierung kategorische Daten

präsentiert. Diese Beiträge sind Teil des Prä- bzw. Postprozessors im entwickelten Konzept. Die primären Funktionen des Anfrageprozessors wurden ebenfalls kurz eingeführt; die genauere Beschreibung erfolgte jedoch in den Schwerpunkten Quasi-Identifikatoren (Kapitel 4) und Vertikale Anfrageverteilung (Kapitel 6).

7.1.2 Schützenswerte Daten

Die zweite Zielsetzung wurde vorwiegend im Kapitel 4 realisiert und beinhaltet die effiziente Berechnung von Quasi-Identifikatoren in hochdimensionalen Daten. Anonymisierungskonzepte unterteilen die Informationen eines Tupels in Schlüssel- bzw. quasi-identifizierende Attributmengen, sowie sensitive und nicht-sensitive Daten. In dieser Arbeit wurde gezeigt, dass, insbesondere in hochdimensionalen Daten, diese Grenzen verschwimmen, da eine Vielzahl von sich überdeckenden Quasi-Identifikatoren in einem Datenbestand präsent sein können. Dadurch können Algorithmen zur Anonymisierung mittels Generalisierung, Permutation oder auf Basis von Differential Privacy nur unzureichend bzw. mit einem zu hohen Informationsverlust parametrisiert und angewendet werden.

7.1.3 Schrittweise Vorberechnung lokaler Daten

Die dritte Zielsetzung und der Kern dieser Arbeit wurde konzeptionell in Kapitel 6 bearbeitet. Es wurde eine Methodik entwickelt, die die Relevanz und Übertragbarkeit der bisherigen Forschung im Gebiet Query Containment zur Lösung von Herausforderungen im Bereich des Datenschutzes, speziell der Datensparsamkeit, aufzeigt. *Answering Queries using Operators* bietet ein leistungsfähiges Regelwerk für die Erstellung von Anfragetransformationen in verteilten Informationssystemen.

Eine vielversprechende Anwendung unserer Technik ist der Einsatz in Sensornetzwerken sowie Assistenzsystemen. Unsere Forschung unterstützt Entwickler von Informationssystemen, indem komplexe Auswertungen automatisiert auf mehrere Rechenknoten verteilt werden können. Dieser Ansatz hat, neben der Einhaltung von Datenschutzaspekten, das Potenzial, dass eine performantere Ausführung der Anwendung innerhalb eines Edge-Computing-Netzwerks im Vergleich zu traditionellen Cloud-Lösungen ermöglicht wird. Obwohl es Einschränkungen durch die Heterogenität der eingesetzten Hard- und Software gibt, sind wir zuversichtlich, dass durch eine standardisierte Programmierschnittstelle der entwickelte Demonstrator leicht in bestehende Systemumgebungen integriert werden kann.

7.2 Ausblick

Wir schlagen vor, dass in den folgenden Bereichen weitere Untersuchungen durchgeführt werden sollten: In Unterabschnitt 7.2.1 wird auf eine alternative Repräsentation der Regeln mittels CHASE [Pop01] eingegangen. Anschließend wird in Unterabschnitt 7.2.2 auf mögliche Erweiterungen des bestehenden Regelwerks eingegangen. Abschließend wird in Unterabschnitt 7.2.3 der Bogen zurück zum PArADISE-Projekt (siehe Unterabschnitt 2.5.3) und dessen weitere Entwicklung gespannt.

7.2.1 Alternative Umsetzung des Verfahrens mittels CHASE

Die eingeführte Anfragetransformation mittels *Query Rewriting by Contract* kann alternativ durch den CHASE [Pop01] und tgd-ähnliche Regeln weiterentwickelt werden. Eine mögliche Transformation könnte hierfür durch eine Regel, wie in Gleichung 7.1 zu sehen, realisiert werden. In dem Beispiel erfolgt die Ersetzung des Verbunds mit Test auf Gleichheit durch einen Verbund mit einem kleiner-gleich-Prädikat, gefolgt von einem Test auf Gleichheit auf der höheren Ebene.

$$
\begin{aligned}
& D_i = \{R(x,y) \bowtie_{y\theta a} S(a,b)\} \\
& \quad \wedge\, \theta =' =' \\
& \quad \wedge\, \neg support(\theta, L_i)
\end{aligned}
\quad \rightarrow \quad
\begin{aligned}
& \exists \theta' : D_i = \{R(x,y) \bowtie_{y\theta' a} S(a,b)\} \\
& \qquad \wedge\, \theta' =' \leq' \\
& \qquad \wedge\, support(\theta', L_i) \\
& \qquad \wedge\, D_{i+1} = \{\sigma_{y\theta a}(D_i)\} \\
& \qquad \wedge\, support(\theta, L_{i+1})
\end{aligned}
\tag{7.1}
$$

L_i und D_i stehen hierbei für die jeweils aktuell betrachtete Ebne bzw. den dazugehörigen zu verarbeitenden Datenbestand. In diesem Fall sollen die beiden Relationen R und S durch einen Gleichverbund miteinander verknüpft werden, jedoch wird die Selektion auf Gleichheit zwischen den Attributen y und a auf L_i nicht unterstützt (siehe letztes Prädikat auf der linken Seite der Regel).

Die Anfragetransformationen mittels CHASE lassen sich dazu grob mittels zweier Templates darstellen: *Template 1* fügt zu einem binären Operator bis zu zwei weitere unäre Operatoren hinzu. Dies kann z. B. zum Hinunterziehen von Selektionsprädikaten unter einen Verbund oder einer Mengenoperation genutzt werden.

Durch *Template 2* wird ein unärer Operator durch einen weiteren unären Operator ergänzt. Dies kann z. B. für das Einfügen von zusätzlichen Selektionsbedingungen genutzt werden, wie sie in den Regeln der Klassen `A` und `L` vorkommen.

Beispieltemplate 1: Ersetzung eines binären Operators θ durch zwei unäre Operatoren θ_1 und θ_2, gefolgt von einem binären Operator θ' auf der nächsthöheren Ebene L_{i+1}.

$$\begin{aligned} D_i = \{R\theta S\} & \\ \wedge \neg support(\theta, L_i) \quad & \rightarrow \quad \exists \theta_1, \theta_2, \theta' : D_i = \{\theta_1(R), \theta_2(S)\} \\ & \qquad \wedge support(\theta_1, L_i) \\ & \qquad \wedge support(\theta_2, L_i) \\ & \qquad \wedge D_{i+1} := \{R\theta' S(D_i)\} \\ & \qquad \wedge support(\theta', L_{i+1}) \end{aligned} \tag{7.2}$$

Beispieltemplate 2: Ersetzung eines unären Operators θ durch Folge zweier unärer Operatoren θ_1 und θ_2; jeweils ein Operator auf der aktuellen und auf der nächsthöheren Ebene L_{i+1}.

$$\begin{aligned} D_i = \theta(R) & \\ \wedge \neg support(\theta, L_i) \quad & \rightarrow \quad \exists \theta_1, \theta_2 : D_i = \{\theta_1(R)\} \\ & \qquad \wedge support(\theta_1, L_i) \\ & \qquad \wedge support(\theta_2, L_{i+1}) \\ & \qquad \wedge D_{i+1} = \{\theta_2(D_i)\} \end{aligned} \tag{7.3}$$

Durch den Einsatz des *Chase&Backchase*-Verfahrens [DPT06] bzw. dessen Erweiterung zum *Provenance-Directed Chase&Backchase* [DH13] können äquivalente Anfragen unter Bewahrung einer Menge von Integritätsbedingungen C bestimmt werden. Die Backchase-Phase reduziert mögliche Anfragepläne auf eine minimale Anfrage, die weiterhin äquivalent zur Originalanfrage ist. In Kombination mit dem AQuO-Konzept können somit neue Transformationen und Regeln erzeugt werden, die das bisher ermittelte *Rewriting Supremum* ersetzen.

7.2.2 Erweiterung des Regelverzeichnisses

In Gleichung 6.27 wurde angesprochen, dass das *Rewriting Supremum* der Anfrage Q aus einer Folge von Anfragetransformationen bzw. Abbildungsfunktionen erstellt wird. Der implementierte Demonstrator umfasst dabei die in Anhang C sowie die in [Kle20] und [Kas21] aufgeführten Transformationsregeln. Unter Beachtung des aktuellen Stands der Forschung können neue Regeln entworfen werden, um weitere Anfragetransformationen zu finden. Diese können dabei das bisher ermittelte *Rewriting Supremum* ersetzen, sofern nicht bereits eine äquivalente Transformation vorliegt. Eine Übersicht zu möglichen Erweiterungen und Ansätzen ist in Tabelle 5.1 zu sehen.

In dieser Arbeit wurden primär SQL-Anfragen mit Aggregatfunktionen und Selektionsprädikaten betrachtet. Dies lässt sich auf komplexere Funktionen ausweiten. Beispielsweise arbeitet Google gegenwärtig an einer Clientseitigen Spracherkennung mittels Neuronalen Netzen [PRS+17].

Erste Ansätze hierzu wurden bereits in [SJMK21] betrachtet. In dem Verfahren wird mittels eines kostenbasierten Optimierers entschieden, ob eine Vorhersage mittels maschineller Lernverfahren vollständig auf einem Edge-Gerät oder der Cloud ausgeführt wird. In Verbindung mit den in dieser Arbeit entwickelten Techniken wären auch hybride Auswertungen, d. h. sowohl in der Cloud als auch auf dem Edge-Gerät, möglich.

Für diese Arten von Analysen ist zu ermitteln, wie sich diese verteilen lassen, beispielsweise zwischen den Schichten des Neuronalen Netzes oder als Partitionierung in Teilgraphen. In [OAF+19] wurde in diesem Zusammenhang gezeigt, wie FPGAs zu einer deutlichen Reduzierung der Latenz bei der parallelen Ausführung von Verfahren des maschinellen Lernens, insbesondere von Entscheidungsbäumen, beitragen können. Werden diese Ergebnisse auf die in dieser Arbeit vorgeschlagene Schichtenarchitektur übertragen, könnte die Beantwortungszeit der Anfragen weiter verkürzt werden.

Als mögliche Erweiterungen ist auch die Einbindung von rekursiven Funktionen sowie die Unterstützung von nutzerdefinierten Funktionen (engl.: *User-defined Functions*; UDFs) und hinterlegten Prozeduren (engl.: *Stored Procedures*; SPs) denkbar. So ist beispielsweise der Operator `Rekursion` ausführbar, wenn dieser auf der aktuellen Ebene unterstützt wird und jeder Operator innerhalb des rekursiven Aufrufs von der aktuellen bzw. durch

eine der unteren Ebenen unterstützt wird. Zusätzlich kann eine Programmflussanalyse durchgeführt werden, damit nicht-unterstützte Operatoren aus der Rekursion ausgelagert werden, beispielsweise wenn ein Wert unabhängig von der Anzahl der Schleifendurchläufe berechnet werden kann. Für Stored Procedures und User-defined Functions gilt eine ähnliche Regel: SPs bzw. UDFs sind ausführbar, wenn deren Aufruf von der aktuellen Ebene unterstützt wird und jeder Operator innerhalb der SP bzw. UDF von der aktuellen Ebene bzw. durch eine der unteren Ebenen unterstützt wird.

7.2.3 PArADISE als Langzeitrahmenprojekt

PArADISE ist als Langzeitrahmenprojekt des Lehrstuhles für Datenbank- und Informationssysteme ausgelegt. Die Schwerpunkte des Projektes liegen auf der massiv parallelen Verarbeitung von Big-Data-Analytics-Anwendungen unter der Leitung von Dennis Marten (siehe dazu u. a. [MH15a], [MH15b] und [MH17]) sowie deren datenschutzkonformen Umsetzung. Letzterer Punkt wurde ausführlich im Rahmen dieser Arbeit behandelt.

In der Arbeit von Marten wird untersucht, in wie weit sich Intraoperatorparallelität für maschinelle Lernverfahren umsetzen lässt. Dabei werden u. a. Hidden-Markov-Modelle in Folgen von SQL-Anweisungen überführt und diese parallelisiert. Als Verbindung zu den Konzepten in dieser Arbeit kann in zukünftigen Forschungsprojekten untersucht werden, inwieweit sich die Regeln auch direkt in den Parallelisierungsprozess integrieren lassen. Bisher wird die Anfragetransformation unter Datenschutzaspekten zur Laufzeit eines Assistenzsystems unabhängig von der Parallelisierung während der Trainingsphase betrachtet (vergleiche Abbildung 2.7). Durch eine Kombination beider Phasen lassen sich ggf. neue, effizientere Anfragetransformationen finden.

In diesem Zusammenhang kann zudem untersucht werden, wie sich Techniken aus dem Bereich Compilerbau (siehe [ULS08] und [BG16]), wie die statischen Codeanalyse, zur Optimierung und Erkennung von parallel ausführbaren Programmteilen nutzen lassen. Dadurch lassen sich nicht nur Operatoren der relationalen Algebra betrachten, sondern auch die Implementierungen der konkret verwendeten Planoperatoren, die bei der Anfrageplanung durch das Datenbankmanagementsystem ausgewählt werden. Dies kann zu weiteren Verbesserungen der horizontalen als auch vertikalen Anfragetransformation führen.

7.2.4 Vision: Das Internet der Dinge als Datenschutz-orientierte Datenbank-Maschine

In [GKB+16] und [HG20] wurde von meinen Mitautoren und mir ein größeres Bild für den Einsatz der datensparsamen Datenverarbeitung im Internet der Dinge skizziert. Die Hauptidee in beiden Publikationen besteht darin, das *gesamte* Internet der Dinge als Datenbankmaschine zu nutzen, anstatt einen Computer-Cluster mit homogenen Knoten und sehr schnellen Verbindungen mit hoher Bandbreite zu verwenden.

Aus datenbankspezifischer Sicht müssen Analyseoperationen auf großen Mengen verteilter Sensordaten, die derzeit auf klassischen Big-Data-Techniken basieren und auf großen, homogen ausgestatteten Parallelrechnern ausgeführt werden, automatisch auf Milliarden von Prozessoren mit Energie- und Kapazitätsbeschränkungen transformiert werden. Dabei sind insbesondere die Anforderungen an

1. Zeitbeschränkungen (Echtzeitfähigkeit zeitkritischer Anwendungen),
2. Platzbeschränkungen (flüchtiger vs. nicht-flüchtiger Speicher auf IoT-Geräten),
3. Qualitätsbeschränkungen (Zusammenführen von Informationen über ähnliche Zeitstempel),
4. Energiebeschränkungen und
5. Datenschutzbeschränkungen nach Vorgaben der Besitzer von IoT-Endgeräten

zu berücksichtigen. Dies führt zu der Fragestellung, inwieweit und für welche Arten von Analysen eine massiv verteilte Datenbankmaschine verwendet werden kann. In diesem Zusammenhang stellt das *Opportunistic Computing*, bei dem sich Geräte spontan für die Bereitstellung von Funktionalitäten zusammenschließen, einen interessanten Anknüpfungspunkt dar. Neben der eigentlichen Anfrageverteilung muss in opportunistischen Edge-Netzwerken auch auf die Verfügbarkeit der Daten in den unterschiedlichsten Netzwerktopologien geachtet werden [Sil21].

Die genannten Aspekte stellen nur eine Auswahl möglicher und zukünftiger Erweiterungen dar. Nachdem in diesem Kapitel die Schlussbetrachtungen gezogen wurden, schließt die Arbeit mit einer Reihe von Anhängen sowie den obligatorischen Verzeichnissen ab.

Anhang

Anhang A

Anhang zu Kapitel 2

In diesem Anhang wird auf die weiterführenden Konzepte aus dem Kapitel 2 eingegangen.

A.1 Assistenzsysteme im Einsatz

Häusliche Pflege

Nach einer Studie der Bundesarbeitsgemeinschaft der Seniorenorganisationen (BAGSO) e. V. [Bun05] fürchten viele Menschen, dass das zunehmende Alter einerseits mit Krankheit und Leiden (67,7% der Befragten) sowie Einsamkeit (51,5%) verbunden ist, aber vor allem auch mit dem Verlust der Selbstständigkeit (76,9%) und das damit verbundene Gefühl, anderen zur Last zu fallen (47,6%), verstärkt wird. Gleichzeitig wohnt der größte Teil der Senioren (88%) auch im hohen Alter weiterhin im eigenen Haushalt. Neben der Pflege durch Angehörige stellen häusliche Pflegedienste ein erprobtes Mittel zur Unterstützung der pflegebedürftigen Menschen für ein weitestgehend selbstständiges und selbstbestimmtes Leben dar.

In der häuslichen Pflege werden zunehmend auch Assistenzsysteme eingesetzt. Dabei kristallisierten sich zwei Einsatzszenarien heraus: Einerseits kann der Betreuungsaufwand vor Ort reduziert werden, indem mittels automatisierter Erstellung von Berichten und Dokumentationen die Arbeit der Pfleger erleichtert wird. Außerdem wird durch den Einsatz von spezialisierten Sensorsystemen in Verbindung mit angepassten Systemen, die direkt in die Wohnung integriert sind, die Möglichkeit geschaffen, pflegebedürftige Menschen eigenständig leben zu lassen und nur im Notfall zu reagieren. Letztgenannte Systeme können einerseits als Erinnerungsfunktion genutzt werden, um die Nutzer an die Einnahme von Medikamenten bzw. das Ausschalten elektrischer Geräte zu erinnern, aber auch um den Tagesablauf weitestgehend zu automatisieren, indem beispielsweise die elektrischen Geräte nach der Nutzung automatisch abgeschaltet werden.

Sturzerkennung: Im Landesforschungsverbund *Mobile Assistenzsysteme*, insbesondere in dessen Teilprojekten MARIKA [BEH06] und MARTA [SDFB09] wurden Verfahren zur Anfrageverarbeitung in verschiedenen Anwendungsszenarien entwickelt, die selbstständig auf die Bereitstellung von kontextbezogenen Informationen reagieren. Dies können beispielsweise kontextabhängige Techniken zur Selektion, Komprimierung und Aggregation sein, die durch Vorarbeiten der DFG-Forschergruppe *MOVI* [Lub00] erarbeitet wurden.

Ein darauf aufbauendes Projekt ist die Sturzerkennung von betreuungsbedürftigen Menschen. In mehreren unabhängigen Arbeiten [RGM+12, KK13] wurde untersucht, inwieweit Stürze in einer Wohnung erkannt werden können – ohne dabei die Privatsphäre der dort lebenden Personen zu verletzten.

Pflegedokumentation: Ambulante Pflegekräfte werden seit einigen Jahren durch mobile Geräte bei der Pflegedokumentation unterstützt. Die Dokumentation der Pflege ist in Deutschland einerseits eine gesetzliche Anforderung, dient aber auch zeitgleich dem Qualitätsmanagement [Ste16]. Die papierbasierten Patientenakten werden

vermehrt durch digitale Pflegeakten ersetzt. Da die manuelle Dokumentation einen nicht unerheblichen Teil der Arbeitszeit ausmacht, kann der Pfleger entlastet werden bzw. sich mehr der eigentlichen Pflege des Pflegebedürftigen beschäftigen. Dabei werden in standardisierten Berichten, die beispielsweise auf HL7 CDA (ISO 10781)[1] basieren, gesundheits- und personenbezogene Daten gespeichert. Ein Beispiel für die Anwendung von HL7 CDA ist der eArztbrief, welcher für eine papierlose Arztpraxis sorgt [HL715]. Um den Pfleger bei der Arbeit nicht unnötig zu behindern, wird die notwendige Sensorik, wie RFID-Chips, meist in die Kleidung integriert bzw. an Objekten befestigt.

Das Trainieren des notwendigen Models erfolgt häufig mit aus der Filmindustrie bekannten Motion-Capturing-Anzügen, welche die Position, die Beschleunigung und das Drehmoment der einzelnen Körperteile mit 120 Hz messen. Die dabei generierte Datenmenge übersteigt die Leistungsfähigkeit vieler Rechner, um eine Echtzeitauswertung auf den Rohdaten zu gewährleisten, wodurch eine Dimensionsreduktion auf die notwendigen Attribute erfolgen muss. Dies kann beispielsweise durch Hauptkomponentenanalyse realisiert werden [Sal16]. Unter Ausnutzung der Markov-Ketten und nicht-überwachter Clusterverfahren können anschließend die Reihenfolgen der Pflegehandlungen und das Beobachtungsmodell abgeleitet werden, wodurch sich die bekannten Pflegeaktivitäten und deren Dauer bestimmen lassen [HK08]. Die erfassten Daten zur Aktivitätserkennung, zur Lokation, zur Pflegedokumentation sowie weitere Gesundheits- und Vitaldaten werden lokal vor Ort erfasst und anschließend im Büro des jeweiligen Pflegedienstes auf deren eigener Hardware gespeichert. Dabei werden ggf. die Daten von mehreren Assistenzsystemen in eine gemeinsame Datenbasis integriert. Aus Datenschutzsicht sollten nur Daten aus dem Pflegeinformationssystem übertragen und auf dem mobilen Gerät angezeigt werden, welche für die aktuelle Tätigkeit, Zeit und Situation der Pflegekraft relevant sind. Eine stufenweise Auswertung der Daten, die möglichst nahe am Datenerzeuger erfolgt, ermöglicht es den Aspekt der Datensparsamkeit in einem integrierten Informationssystem erfolgreich umzusetzen.

Weitere Aspekte: Ebenso werden in der Pflege häufig biometrische Daten, wie die Statur und Größe einer Person, sowie weitere Körpermerkmale, wie Iris und Fingerabdrücke, erfasst. Diese Merkmale werden, neben der medizinischen Auswertung, auch zur Identifikation einer Person erfasst. Datenschutz ist im medizinischen Bereich extrem wichtig, da sich aus Sensordaten und Intentionserkennung eine Vielzahl an ethischen Konflikten ergeben: Werden aus den Sensordaten weitere Informationen abgeleitet, die eigentlich nicht zu den Zielsetzungen des Systems gehören? Ist es gesellschaftlich und moralisch akzeptabel, den Geisteszustand einer Person abzuleiten, mögliche Handlungen vor der eigentlichen Ausübung (Stichwort: Minority Report) zu erkennen und Verhaltensmuster einer Person zu bestimmen? Aus diesem Grund werden für medizinische Informationssysteme häufig Vertrauens- und Registerstellen eingerichtet, um diese Art von Datenschutzvergehen zu melden und die Anonymisierung der Daten umzusetzen [MKSS95].

Fahrerassistenzsysteme

Einer aktuellen Statistik zufolge [Sta18] sind die meisten Unfälle im Straßenverkehr auf menschliches Versagen und Fehlverhalten zurückzuführen. Durch den Einsatz von Assistenzsystemen im Fahrzeug kann die Unfallzahl gesenkt werden, wenn dem Fahrer Handlungen wie Brems- und Lenkmanöver abgenommen werden, bzw. er durch akustische und optische Warnsignale auf Gefahren hingewiesen wird.

Seit 2001 werden von Google und anderen Unternehmen auf öffentlichen Straßen Tests durchgeführt, um Forschungsergebnisse für (teil-)autonomes Fahren unter realen Bedingungen zu überprüfen. Die Ansätze zur Fahrerassistenz sind allerdings schon viel älter und haben sich in den Grundzügen nicht verändert [Yam17]: Bereits Anfang der 1970er Jahre wurde mit Kameras experimentiert, um Gegenstände auf der Fahrbahn zu erkennen und als Folge Ausweichbewegungen auszuführen.

Der Trend des autonomen Fahrens geht sogar so weit, dass die FIA[2] im Jahr 2015 eine Weltmeisterschaft mit unbemannten Fahrzeugen angekündigt hat [Yam17]. Diese sollen auf Basis künstlicher Intelligenz und der aus

[1] Verfügbar unter `https://www.iso.org/standard/57757.html`, zuletzt aufgerufen am 05.01.2022

[2] Fédération Internationale de l'Automobile, Internationaler Automobilverband

der Formel E bekannten Elektrofahrzeuge eigenständig die Rennstrecke entlangfahren und auf die Aktionen ihrer Konkurrenten eingehen können. Seit September 2020 befindet sich dieses Vorhaben im Beta-Test[3].

Weitere Beispiele

Das Smart City Lab der Universität Bamberg [SBKN16] definierte bereits während der Designphase Anforderungen und Ziele für seine grundlegende Architektur. Dazu gehören u. a. der Einsatz von rollenbasierten Zugriffsmodellen zur Sicherstellung der Authentizität, der Einsatz von TLS zur Transportverschlüsselung sowie die Pseudonymisierung von personenbezogenen Daten via Hashing, bevor die Daten in der Datenbank abgelegt werden. Zur Veröffentlichung der Daten, insbesondere von Bewegungsprofilen, werden zudem weitere Anonymisierungstechniken, wie beispielsweise k-Anonymität, eingesetzt.

Das Aspern Smart ICT [DEP+15], auch ein Smart City-Projekt mit Fokus auf Smart Metering, untersucht ebenfalls, inwieweit Datenschutzaspekte in das Design eines solchen Systems einfließen können. Im Projekt werden verschiedene Datenquellen, wie Wetter-, Gebäude- und Smart-Meter-Daten in eine gemeinsame Plattform integriert. Verschiedene Nutzergruppen, wie Datenbereitsteller, Anwendungsentwickler und staatliche Stellen, aber auch der gemeine Bürger, können, sofern sie dazu befugt sind, über diverse Schnittstellen auf die Daten zugreifen. Dabei werden verschiedene Datenschutzaspekte beachtet: Durch organisatorische und technische Maßnahmen wird sichergestellt, dass der Datenschutz stets zugesichert werden kann. Zu den konkreten Techniken zählen flexible Datenschutzrichtlinien für aktuelle und zukünftige (auch unbekannte) Anwendungen, der Einsatz von Zugriffskontrollmechanismen in verschiedenen Granularitäten sowie der Anwendung von Anonymisierungs- und Verschlüsselungsverfahren. Außerdem wird durch Data Provenance überprüft, woher Daten stammen und wie diese weiterverarbeitet wurden.

A.2 Zugriffskontrolle in Datenbanksystemen

Vertraulichkeit, Integrität und Verfügbarkeit sind drei Grundbedürfnisse des Datenschutzes. Um diese Faktoren umzusetzen, ist es notwendig, eine Zugriffskontrolle auf den gesammelten Daten einzuführen. Durch eine feingranulare Modellierung des Zugriffs können sensible Informationen vor unautorisierten Zugriff, Veränderungen bzw. Löschen geschützt werden.

Datenbanksysteme sind in vielfältiger Art und Weise zugangsbeschränkt. Zum einen wird über externe Zugangskontrolllisten (engl.: Access Contol Lists, ACL) geregelt, welcher Nutzer sich von welchem Rechnernetz auf welches Datenbankschema überhaupt verbinden darf. Ist ein Nutzer mit einer Datenbank verbunden, so können vom Datenbanksystem selbst weitere, feingranularere Einschränkungen bezüglich des Datenbankschemas festgelegt werden.

Im SQL:2003-Standard [Mel03] wurde zudem ein Rollenmodell eingeführt. Dies erlaubt, dass Nutzer verschiedene Rollen einnehmen können. Über die Rollen wird der Zugriff auf Basisrelationen, Sichten und Prozeduren festgelegt. Die Sichten und Prozeduren erlauben zudem weitere Einschränkungen hinsichtlich der abfragbaren Datenmenge.

In diesem Unterabschnitt werden zunächst die ACLs näher erläutert, bevor auf die feingranularere Vergabe durch Zugriffsrechte eingegangen wird. Ein Überblick über weitere Techniken schließt die Betrachtung der Zugriffskontrollmechanismen ab.

Zugangskontrolllisten

Zugangskontrolllisten regeln den allgemeinen Zugriff von Nutzern auf Objekte. In Datenbankmanagementsystemen wird dabei im Speziellen festgelegt, von wo ein Nutzer oder eine Nutzergruppe auf festgelegte Datenbanken zugreifen darf. Spezifische Rechte, wie beispielsweise der Zugriff auf einzelne Relationen oder Sichten, werden

[3] `https://roborace.com/`, zuletzt aufgerufen am 05.01.2022

an diesem Punkt noch nicht festgelegt. Der Aufbau einer ACL wird im Folgenden am Beispiel des Datenbankmanagementsystems PostgreSQL[4] erklärt:

Beispiel: Aufbau einer ACL in PostgreSQL

In PostgreSQL existieren vier unterschiedliche Verbindungstypen, die mit bis zu fünf Parametern initialisiert werden können. Der Verbindungstyp `local` steht für einen Socket aus der Unix-Domain, während mit `host`, `hostssl` und `hostnossl` Verbindungen über SSL-verschlüsselte bzw. unverschlüsselte TCP/IP-Sockets hergestellt werden können.

```
local      DATABASE  USER  METHOD  [OPTIONS]
host       DATABASE  USER  ADDRESS  METHOD  [OPTIONS]
hostssl    DATABASE  USER  ADDRESS  METHOD  [OPTIONS]
hostnossl  DATABASE  USER  ADDRESS  METHOD  [OPTIONS]
```

Der Parameter `DATABASE` gibt an, auf welche Datenbank das Verbindungsprofil passt. Typischerweise wird hier der Name der Datenbanken angegeben oder mit dem Schlüsselwort `all` der Zugriff auf alle Datenbanken im Schema gestattet. Alternativ kann mit `sameuser` bzw. `samerole` der Zugriff auf alle Datenbanken des gleichen Nutzers bzw. der gleichen Rolle gewährt werden.

Unter `USER` wird der Name des zugangsberechtigten Nutzers eingetragen bzw. mit `all` allen Nutzern der Zugang gewährt. Die `ADDRESS` legt den Adressraum fest, von welchem sich der Nutzer aus mit der Datenbank verbinden darf. Dabei können wahlweise der Hostname angegeben werden oder IPv4- bzw. IPv6-Adressräume spezifiziert werden.

Mit der `METHOD` wird die Art der Authentifizierung festgelegt. Als bevorzugte Variante wird dabei `md5` angesehen, da das zur Authentifizierung verwendete Passwort verschlüsselt übertragen wird, während bei `password` die Passwörter im Klartext gesendet werden. Als weitere, gängige Alternative steht mit `ldap` (Lightweight Directory Access Protocol) ein Protokoll zur Abfrage der Authentifizierungsdaten zur Verfügung. In den `OPTIONS` können weitere Parameter für die verwendete Authentifizierungsmethode angegeben werden.

Im Folgenden ist ein Ausschnitt der Datei pg_hba.conf einer PostgreSQL-Installation abgebildet. Sie enthält insgesamt fünf Einträge; davon vier für die Kommunikation via TCP/IP. Der erste Eintrag erlaubt es allen Nutzern unter dem lokalen Unix- bzw. Linux-Betriebssystem die Verbindung zu allen Datenbanken ohne weitere Authentifizierung. Sofern der Datenbankserver abgesichert ist und die Nutzer ausschließlich für den Datenbankbetrieb angelegt wurden, ist diese Standardeinstellung unbedenklich. Dies trifft auch auf den zweiten und dritten Eintrag zu, welche die Kommunikation vom lokalen Server via TCP/IPv4 bzw. TCP/IPv6 regeln.

```
local all     all                     trust
host  all     all       127.0.0.1/32  trust
host  all     all       ::1/128       trust
host  all     postgres  10.10.3.1/32  trust
host  music   all       0.0.0.0/0     md5
```

Der vierte Eintrag gestattet es dem Nutzer *postgres* zusätzlich die Verbindung auf alle Datenbanken von der IP 10.10.3.1, welches sich im gleichen lokalen Subnetz wie der Datenbankserver befindet. Über den Nutzer *postgres* wird in der Regel die Datenbank verwaltet. In diesem konkreten Fall wird ein ganzes Cluster von PostgreSQL-Servern verwaltet, welche vom Server mit der angegebenen IP orchestriert werden.

Der letzte Eintrag gestattet es allen Nutzern auf die Datenbank *music* zuzugreifen. Außerdem müssen die Nutzer sich in diesem Fall mit ihrem Passwort authentifizieren, welches als MD5-Hash übertragen wird.

Da die ACL bei vielen Nutzern schnell sehr groß wird, kann die Pflege der Zugriffsrechte sehr unhandlich werden. Um diesem Problem entgegenzuwirken, können Nutzer mit gleichen Rechten zu Gruppen bzw. Rollen

[4] `https://www.postgresql.org/`, zuletzt aufgerufen am 05.01.2022.

zusammengefasst werden. Sofern möglich, kann der Zugriff auf die Datenbank zudem über mehrere Koordinatoren erfolgen, die jeweils ihre eigenen ACLs verwalten. Dadurch kann eine große ACL hinsichtlich der verfügbaren Datenbankinstanzen bzw. Nutzergruppen partitioniert werden.

Zugriffsrechte durch Discretionary Access Control

Discretionary Access Control (DAC, deutsch: Diskretionäre Zugriffskontrolle) ist ein Sicherheitskonzept, bei dem Zugriffsrechte als Tripel der Form (Nutzer, Relation, Operation) aufgefasst werden. Es wird entsprechend festgelegt, welcher Datenbanknutzer bzw. -rolle auf welche Relationen und Sichten zugreifen darf, um dort Daten zu lesen, zu ändern, einzufügen oder zu löschen.

In SQL wird dies mittels des `GRANT`-Befehls umgesetzt. Im Quellcodeausschnitt A.1 ist die Syntax der `GRANT`-Klausel unter PostgreSQL vereinfacht dargestellt. Nach dem Schlüsselwort `GRANT` kann entweder eine Auswahl der vier Befehle für den Zugriff auf die Daten (`SELECT`), dem Einfügen (`INSERT`), dem Aktualisieren (`UPDATE`) oder dem Löschen (`DELETE`) erfolgen oder mittels `ALL` bzw. der Langform `ALL PRIVILEGES` alle Zugriffsrechte eingeräumt werden.

```
1 GRANT ((SELECT | INSERT | UPDATE | DELETE)+ | ALL [PRIVILEGES])
2 ON ([TABLE] tablename (, tablename)*
3   | ALL TABLES IN SCHEMA schemaname (, schemaname)*)
4 TO rolename (, rolename)*
5 [WITH GRANT OPTION]
```

Quellcodeausschnitt A.1: GRANT-Befehl in PostgreSQL (vereinfachte Darstellung)

Die `ON`-Klausel spezifiziert die einzelnen betroffenen Relationen und Sichten bzw. erlaubt mittels `ALL TABLES IN SCHEMA` die Ausweitung auf alle Relationen und Sichten in einem oder mehreren Datenbankschemata. In der `TO`-Klausel werden die autorisierten Nutzer bzw. Rollen angegeben. Die spezielle Rolle `PUBLIC` erlaubt den Zugriff für alle Nutzer. Die optionale Klausel `WITH GRANT OPTION` ermöglicht die Weitergabe der spezifizierten Rechte an weitere Nutzer.

Sollen die vergebenen Rechte wieder entzogen werden, kann dies mittels dem `REVOKE`-Befehl erfolgen. Die Struktur des `REVOKE`-Befehls ist in Quellcodeausschnitt A.2 zu sehen und ist ähnlich zu der `GRANT`-Klausel, weist jedoch zwei Besonderheiten auf. Zum einen kann mittels `GRANT OPTION FOR` nur das Recht auf die Weitergabe der bestehenden Rechte entzogen werden. Außerdem kann entschieden werden, ob die weitergegeben Rechte an andere Nutzer ebenfalls zurückgenommen werden sollen (`CASCADE`) oder ob im Falle einer bereits erfolgten Weitergabe der Rechteentzug abbrechen soll (`RESTRICT`).

```
1 REVOKE [GRANT OPTION FOR] ((SELECT | INSERT | UPDATE | DELETE)+
2   | ALL [PRIVILEGES])
3 ON ([TABLE] tablename (, tablename)*
4   | ALL TABLES IN SCHEMA schemaname (, schemaname)*)
5 FROM rolename (, rolename)*
6 [CASCADE | RESTRICT]
```

Quellcodeausschnitt A.2: REVOKE-Befehl in PostgreSQL (vereinfachte Darstellung)

Das folgende Beispiel zeigt den Einsatz der `GRANT`-Klausel anhand der Musikdatenbank:

Beispiel: Verwendung der GRANT-Klausel

Der folgende Befehl ermöglicht es den Nutzern `holger` und `helga`, Daten aus den Relationen `tracks` und `albums` auszulesen bzw. dort neue Daten einzufügen. Die vergebenen Rechte dürfen `holger` und `helga` an weitere Nutzer weitergeben.

```
GRANT SELECT, INSERT
ON TABLE tracks, albums
TO holger, helga
WITH GRANT OPTION
```

Zugriffsrechte können, neben den vorhandenen Basisrelationen, auch auf Sichten vergeben werden. Durch den Einsatz von Sichten lassen sich somit für verschiedene Nutzer unterschiedliche Ausschnitte über dem gleichen Datenbestand herausgeben, wodurch eine feingranularere Rechtevergabe ermöglicht wird. Die Definition von Sichten und deren Einsatz für eine datenschutzfreundliche Anfrageverarbeitung wird im Folgenden kurz skizziert.

Sichten

Als Sichten (engl.: views) werden virtuelle Relationen bezeichnet, welche eine benannte Anfrage darstellen. Sichten sind externe Datenbank-Schemata, die auf den Basisrelationen aufbauen und über eine Berechnungsvorschrift, beispielsweise einer SQL-Anfrage, generiert werden. Sie werden u. a. zur Vereinfachung von Anfragen eingesetzt, um z. B. häufig generierte Teilanfragen zu realisieren.

Durch die so erfolgte Strukturierung der Datenbank ist es möglich, den Zugriff auf die Basisrelationen weiter einzuschränken. Zugriffsrechte lassen sich, ebenso wie auf den Basisrelationen, für jede einzelne Sicht explizit festlegen. Somit ist es möglich, nutzer- bzw. rollenspezifische Zugriffe zu erlauben.

Für die Definition einer Sicht muss deren Name, das erzeugte Relationenschema und eine Berechnungsvorschrift in Form einer Anfrage angegeben werden. Mittels SQL lassen sich Sichten wie folgt erzeugen:

```
CREATE VIEW <Sichtname> [Schemadeklaration]
AS <SQL-Klausel>
[WITH CHECK OPTION]
```

Quellcodeausschnitt A.3: CREATE VIEW-Klausel (vereinfachte Darstellung)

Das Schlüsselwort `CREATE VIEW` leitet dabei die Sichtdefinition ein. Mit dem *Sichtnamen* wird ein im aktuellen Schema eindeutiger Name zur Identifizierung der Sicht vergeben. Die optionale *Schemadeklaration* erlaubt die Vergabe von eigenen Attributnamen, sofern nicht das durch die *SQL-Klausel* erzeugte Schema verwendet werden soll. Durch die optionale `WITH CHECK OPTION`-Klausel wird die Überprüfung aktiviert, ob in die Sicht neu eingefügte oder veränderte Daten kompatibel mit der Sichtdefinition sind, wodurch Änderungen an die Basisrelationen weitergeleitet werden können.

Beispiel: Beispiel für eine Sichtdefinition

In diesem Beispiel werden die Titel und die Länge aller Pop-Stücke von 1984 aus der Tracks-Relation in einer neuen Sicht namens *popmusic* abgespeichert:

```
CREATE VIEW popmusic AS
SELECT title, length
FROM tracks NATURAL JOIN Genres NATURAL JOIN years
WHERE genres.name = 'Pop'
AND years.name = 1984
```

Problematisch bei Sichten sind die folgenden zwei Punkte: Erstens wird die automatische Anfragetransformation erschwert. Zweitens lassen sich nicht alle Änderungen auf Sichten problemlos durchsetzen.

Speziell in statistischen Datenbanken und Data Warehouses unterliegen die Einzeleinträge den Forderungen des Datenschutzes, während die aggregierten Werte als statistische Informationen durchaus, sofern die Zweckbindung erfüllt ist, veröffentlicht werden dürfen. Die personenbezogenen Einzeleinträge müssen durch Sichten und der

damit verbundenen Rechtevergabe gesondert abgesichert werden, um unbefugten Zugriff vorzubeugen. Es muss zudem zugesichert sein, dass auch durch Folgen von Anfragen der Datenschutz gewährt bleibt. Details zu letzterem Problem und dessen Lösungsansätze werden kurz in Abschnitt 3.4.2 skizziert.

Weitere Sicherheitsmodelle

Neben den oben vorgestellten Konzepten zur Realisierung von Zugriffsrechten existieren – weniger auf Datenbank- als auf Betriebssystemebene – viele weitere Modelle zur Realisierung des Zugriffs auf sensible Daten. Im Folgenden werden einige gebräuchliche Modelle kurz vorgestellt.

Bell-LaPadula Model: Bell und LaPadula [BL73] entwickelten ein Modell, welches den Schwerpunkt auf Vertraulichkeit setzt. Einzelnen Daten und jeder Person wird eine Klassifizierungsstufe, beispielsweise *nicht klassifiziert*, *vertraulich*, *geheim* und *streng geheim*, zugeordnet. Nutzer dürfen nur Daten bis einschließlich ihrer eigenen Stufe lesen und nur Daten überschreiben bzw. anfügen, welche die gleiche oder eine höhere Stufe besitzen. Dadurch wird sichergestellt, dass Informationen nur an die eigenen Vorgesetzten bzw. berechtigte Prozesse weitergegeben werden.

Biba-Modell: Das Biba-Modell [Bib77] stellt eine Umkehrung des Bell-LaPadula-Modells dar: Um die Integrität der Daten zu bewahren, können Nutzer bzw. Prozesse nur von Objekten ihrer eigenen oder einer höheren Stufe lesen und nur auf Objekte ihrer eigenen oder einer niedrigeren Stufe schreiben.

Privacy-aware Role-Based Access Control: In [NBL$^+$10] wird Core P-RBAC vorgestellt, ein feingranulares, rollenbasiertes Regelwerk zur Vergabe von Zugriffsrechten auf hierarchischen Objekten. Im Kern dieser Art von Zugriffskontrolle stehen Rollen, Aktionen und Objekte. Nutzer nehmen Rollen ein. Diesen Rollen werden bestimmte Rechte auf Objekten, Aktionen und Aktionen auf Objekten zugeordnet.

Zusätzlich erfolgt eine Angabe zum Verwendungszweck, zu zusätzlichen Bedingungen und hinsichtlich zu erfüllenden Verpflichtungen (z. B. Benachrichtigung bei Verarbeitung bestimmter Informationen). Dabei existieren zusätzliche Hierarchien für Rollen, Objekte und Verwendungszwecke, um eine feingranulare Rechtevergabe zu erreichen. Für die zusätzlichen Bedingungen können komplexe boolesche Ausdrücke angegeben werden. Zugriffsrechte können zudem negiert formuliert werden, um bestimmte Verwendungen auszuschließen.

Purpose based access control: In [BL08] wird ein Verfahren zur feingranularen Vergabe von Zugriffsrechten mit Zweckbindung vorgestellt. Die Zugriffsrechte lassen sich wahlweise pro Relation, pro Tupel, pro Attribut und sogar pro Attributwert einzeln festlegen. Das Verfahren ist angelehnt an die Klasse der rollenbasierten Zugriffsverfahren (RBAC); es erweitert diese um *conditional roles*, welche auf der Notation für Rollen- und Systemattribute basieren.

Die Realisierung der Zweckbindung erfolgt auf Basis der Modifikation einer gegebenen Anfrage. Diese wird beispielsweise durch Filter oder eingeschränkte Projektionen erweitert. Stellt ein Nutzer eine Anfrage, ermittelt das System, welche Informationen zu welchem Zweck ihm tatsächlich zur Verfügung stehen.

Die Verwendungszwecke (engl: purposes) bilden eine Sammlung von Gründen für die Datensammlung und den Datenzugriff. Diese Zwecke können hierarchisch in einen Baum eingeordnet werden, um Spezialisierungen bzw. Generalisierungen zwischen den einzelnen Verwendungsgründen darzustellen. Zusätzlich existiert eine Unterscheidung in erlaubte (engl.: allowed intended purposes, AIP) und verbotene (engl.: prohibited intended purposes, PIP) beabsichtigte Zwecke. Um sicher zu stellen, dass keine Auswertung für unbeabsichtigte Zwecke erfolgt, werden die PIPs vor den AIPs ausgewertet.

Darüber hinaus existieren weiterhin Zugriffszwecke, die bei einer Anfrage mit übergeben werden. Der Zugriff auf die Daten wird nur erlaubt, wenn der Zugriffszweck sich aus der Implikation der erlaubten, beabsichtigten Zwecke ergibt. Zur Realisierung des Konzepts wird die Syntax der SQL-Anfrage um eine `FOR`-Klausel erweitert, welche den beabsichtigten Verwendungszweck enthält. Zusätzlich erfolgt eine Bindung von Rollenzugriff und Verwendungszweck, wodurch Nutzer nur Verwendungszwecke angeben dürfen, die durch ihre Rolle als zulässig angegeben sind. Die `FOR`-Klausel wird intern mittels Anfragetransformation durch eine `WHERE`-Klausel ersetzt.

In der Datenbank werden die Verwendungszwecke auf Wert-, Attribut-, Tupel- oder Relationsebene angegeben. Die hierfür notwendigen Metadaten werden intern in einer eigenen Relation, der *privacy-policy table*, gespeichert, bzw. als Spaltenerweiterung mit hexadezimaler Kodierung direkt in den Basisrelationen hinterlegt. Die beabsichtigten und erlaubten Verwendungszwecke werden bei der Anfragetransformation bitweise durch das logische UND verknüpft. Falls dabei festgestellt wird, dass ein einzelner Wert nicht rausgegeben werden darf, führt dies zu NULL-Werten in der Ergebnisrelation.

Der folgende Quellcodeausschnitt zeigt die Anfragetransformation für eine einfache Beispielanfrage auf der Musikdatenbank. Es sollen die Namen der einzelnen Nutzer und deren gehörten Musiktitel für zukünftige Musikempfehlungen analysiert werden. Im oberen Teil (Zeile 1 bis 3) ist die originale Anfrage zu sehen. Zeile 1 und 2 enthalten normale SQL-Anweisungen; Zeile 3 enthält die in [BL08] und [BB06] vorgeschlagene Erweiterung um die Zweckbindung, welche mit dem Schlüsselwort `FOR` eingeleitet wird.

```
1 SELECT firstname, lastname, title
2 FROM users NATURAL JOIN listens NATURAL JOIN tracks
3 FOR Recommendation
4
5 SELECT firstname, lastname, title
6 FROM users NATURAL JOIN listens NATURAL JOIN tracks
7 WHERE Comp_Check('0x042', firstname_aip, firstname_pip)
8 AND Comp_Check('0x042', lastname_aip, lastname_pip)
```

Quellcodeausschnitt A.4: Erweiterung von SQL um eine FOR-Klausel zur Realisierung der Zweckbindung)

In den Zeilen 5 bis 8 ist die transformierte Anfrage zu sehen. Durch die Transformation werden zusätzliche Anfrageprädikate in die (in diesem Fall noch nicht existente) `WHERE`-Klausel eingefügt. Dabei wird für jedes in der Projektion vorkommende Attribut mit Personenbezug die nutzerdefinierte Funktion *CompCheck* eingefügt. Als Parameter werden, neben dem zuvor angegebenen Verwendungszweck, die erlaubten und verbotenen Verwendungsmöglichkeiten der einzelnen Attribute übergeben. Der Verwendungszweck wird dabei in einen internen Code umgewandelt (hier: *0x042*), welcher die Position des Verwendungszweckes in der Hierarchie darstellt.

A.3 Gruppierung, Mengenoperationen und neuere Operatoren in SQL

Einige Datenbanksysteme, wie MySQL, die sich nicht an den SQL-Standard halten, erlauben zudem die Angabe von beliebig vielen Attributen innerhalb der `COUNT`-Funktion, sofern mit `DISTINCT` eine Duplikateliminierung erfolgt[5]. Das folgende Beispiel verdeutlicht die Anwendung dieses Features:

Beispiel: COUNT(DISTINCT ...)-Anfragen in MySQL

Die beiden folgenden Anfragen sind – zumindest in MySQL – äquivalent: Sie liefern die Anzahl an unterschiedlichen (`discnumber`, `tracknumber`)-Kombinationen. Die obere Variante wird von anderen Datenbanksystemen – soweit bekannt – nicht unterstützt, wird in MySQL aber fast doppelt so schnell ausgeführt, da ein Zugriff auf eine temporäre Tabelle entfällt.

[5] Weitere Informationen hierzu liefert die MySQL-Dokumentation unter `https://dev.mysql.com/doc/refman/8.0/en/aggregate-functions.html#function_count-distinct`, zuletzt aufgerufen am 05.01.2022.

```sql
SELECT COUNT(DISTINCT discnumber, tracknumber) AS C
FROM tracks;

SELECT COUNT(*) AS C
FROM (
  SELECT DISTINCT discnumber, tracknumber
  FROM tracks
) AS B;
```

Durch das Schlüsselwort `ALL` werden in der Berechnung des Rückgabewertes der Aggregatfunktion `COUNT` Duplikate berücksichtigt – nach dem SQL-Standard eine Voreinstellung, die nicht explizit angegeben werden muss. Nullwerte werden, außer bei `COUNT(*)`, bei der Auswertung durch die Aggregatfunktion nicht berücksichtigt, da sie keine lexikographische oder numerische Ordnung besitzen bzw. nicht durch arithmetische Funktionen verarbeitet werden können.

GROUP BY-Klausel: Die `GROUP-BY`-Klausel dient zur Partitionierung des Wertebereichs vor der Anwendung von Aggregatfunktionen. Die in der `GROUP-BY`-Klausel angegebenen Attribute unterteilen eine Relation in mehrere Gruppen mit unterschiedlichen Attributwerten. Tupel mit den gleichen Attributwerten in den Gruppierungsattributen werden in die gleiche Gruppe einsortiert, sofern das Tupel nicht durch eine vorherige Selektionsbedingung entfernt wurde. Die so erzeugten Gruppen können als virtuell geschachtelte Relationen aufgefasst werden.

HAVING-Klausel: Selektionsbedingungen über die gruppierten und aggregierten Daten werden in der `HAVING`-Klausel angegeben. Die Bedingungen können zum einen an die Gruppierungsattribute gerichtet sein, zum anderen können auch Bedingungen mit Aggregatfunktionen auf den Nicht-Gruppierungsattributen formuliert werden. Die Formulierung der Bedingungen erfolgt analog zur `WHERE`-Klausel. Das folgende Beispiel zeigt eine Anfrage mit einer Kombination aus einer Aggregatfunktion mit Gruppierung und einer Selektionsbedingung auf den einzelnen Gruppen:

Beispiel: SQL-Aggregatanfrage

```sql
SELECT name, Anzahl
FROM (
  SELECT cid, COUNT(tid) AS Anzahl
  FROM tracks
  GROUP BY cid
  HAVING COUNT(tid) > 2
) AS X NATURAL JOIN composers
```

Das folgende Beispiel verdeutlicht die schrittweise Abarbeitung von Aggregatanfragen:

Beispiel: Abarbeitung einer Anfrage mit Aggregation und Gruppierung

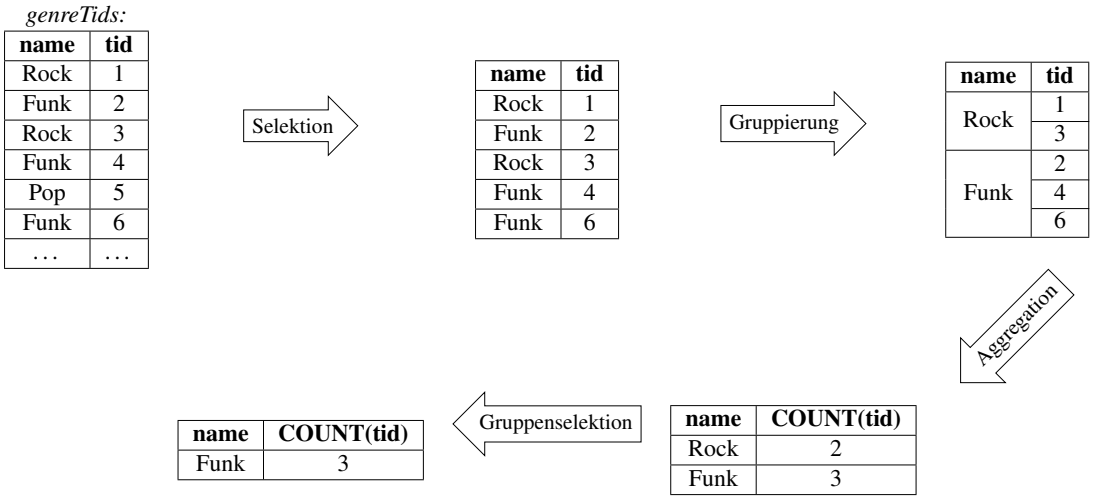
In dieser Anfrage werden alle Genres aufgelistet, die nicht dem Genre *Pop* zugeordnet sind und die mit mindestens drei Titeln in der Datenbank vertreten sind. Als SQL-Anfrage lässt sich dieser Sachverhalt über der Relation `genreTids` (siehe unten) wie folgt formulieren:

```
1 SELECT name, COUNT(tid)
2 FROM genreTids
3 WHERE name = 'Pop'
4 GROUP BY name
5 HAVING COUNT(id) >= 3
```

In der relationalen Algebra wird diese Anfrage folgendermaßen formuliert:

$$\pi_{name,COUNT(tid)}(\sigma_{COUNT(tid)\geq 3}(\gamma_{R;COUNT(tid);name}(genreTids)).$$

Zur Umsetzung werden im ersten Schritt alle Tupel aus der Basisrelation (in der Abbildung rechts oben zu sehen) aussortiert, welche für das Attribut `name` den Wert *Pop* annehmen. Im zweiten Schritt werden den einzelnen Attributwerten des Attributs `name` die IDs der Musikstücke (`tid`) zugeordnet. Pro Genre wird anschließend die Anzahl der Musikstücke bestimmt. Abschließend erfolgt die Selektion auf den gruppierten Werten.



Mengenoperationen: SQL erlaubt die gleichen Mengenoperationen, die auch in der relationalen Algebra bzw. dem Tupelkalkül zur Verfügung stehen. Mit der `UNION`-Klausel wird seit dem ersten Sprachvorschlag, SQL-89, die Vereinigung als Mengenoperation unterstützt. Sie dient zum Vereinigen der Tupel zweier Relationen, die über kompatible Attribute verfügen. In der Ergebnisrelation wird das Schema, d. h., die Attributnamen, der zuerst angegebenen Relation übernommen. Sollen nicht alle Attribute einer Relation verwendet werden, so können diese auch durch Nullwerte ersetzt werden. Seit SQL-92 stehen mit der `INTERSECT`- und der `EXCEPT`-Klausel zwei Konstrukte zur Umsetzung des Durchschnittes bzw. der Differenz auf Mengen zur Verfügung. Durch das optionale Schlüsselwort `CORRESPONDING` kann zudem auf die Angabe der Attributnamen in der `SELECT`-Klausel verzichtet werden, wenn lediglich die gemeinsamen, in beiden Relationen vorkommenden Attribute, verwendet werden sollen.

Beispiel: Mengenoperationen in SQL

Das folgende Beispiel zeigt die Anwendung der Mengenoperationen Vereinigung, Durchschnitt und Differenz. Auf der linken Seite ist die Syntax von PostgreSQL zu sehen. Dieses Datenbanksystem ist nahe am SQL-Standard und unterstützt die genannten Mengenoperationen. Auf der rechten Seite sind die dazu, bis auf die Existenz und Anzahl der Nullwerte, äquivalente Anfrage ohne die Verwendung der Mengenoperatoren aufgeführt.

```
1  SELECT name
2  FROM composers
3  UNION
4  SELECT name
5  FROM artists
6
7
8
9  SELECT name
10 FROM composers
11 INTERSECT
12 SELECT name
13 FROM artists
14
15
16 SELECT name
17 FROM composers
18 EXCEPT
19 SELECT name
20 FROM artists
21
22
```

```
1  SELECT name
2  FROM composers
3  FULL OUTER JOIN (
4    SELECT name
5    FROM artists
6  ) AS stub
7  USING (name)
8
9  SELECT name
10 FROM composers
11 WHERE name IN(
12   SELECT name
13   FROM artists
14 )
15
16 SELECT name
17 FROM composers
18 WHERE name NOT IN(
19   SELECT name
20   FROM artists
21   WHERE name IS NOT NULL
22 )
```

ORDER BY: Die `ORDER BY`-Klausel erzeugt aus einer Menge von Tupeln eine geordnete Liste der Tupel. Die Sortierung erfolgt dabei nach der Attributliste, die in der `ORDER BY`-Klausel spezifiziert wurde, wobei die Abarbeitung von links beginnend erfolgt. Für jedes Attribut kann einzeln angegeben werden, ob es aufsteigend (`ASC`; engl.: ascending; Standardeinstellung) oder absteigend (`DESC`; engl.: descending) sortiert werden soll. Stimmen alle Attributwerte überein, so ist die Reihenfolge der Ausgabe von der konkreten Implementierung des Datenbanksystems abhängig; siehe dazu auch [Mel16, Seite 110; Abschnitt 4.33: Cursors]. Die Sortierung muss nicht zwangsweise nach einem Attribut aus der `SELECT`-Klausel erfolgen – die meisten Datenbanksysteme wandeln intern den Anfrageplan so um, dass die Sortierung vor der Projektion erfolgt und behandeln die sortierte Liste weiterhin als Relation. Ebenso ist die Angabe eines umbenannten Attributes oder eines aggregierten Wertes möglich. Hierbei ist aber auf die Unterstützung seitens des jeweiligen Datenbanksystems zu achten.

Beispiel: SQL-Anfrage mit Sortierung

Die folgende Anfrage gibt den Titel und die dazugehörige Länge der Musikstücke aus der Relation `tracks` aus. Die Sortierung erfolgt dabei absteigend nach der Länge der Titel. Bei gleichlangen Titeln wird derjenige Titel zuerst ausgegeben, der die kleinste `discnumber` hat. Ist auch hier der Attributwert gleich, wird zudem überprüft, welcher Titel die kleinere `tracknumber` hat. Wird auch hier kein Unterschied festgestellt, so ist die Reihenfolge abhängig von der Implementierung im jeweiligen Datenbankmanagementsy-

stem. Gleiches gilt auch für die Einordnung derjenigen Titel, welche für die entsprechenden Attribute über keinen Attributwert verfügen.

```
SELECT title, length
FROM tracks
ORDER BY length DESC, discnumber ASC, tracknumber ASC
```

Die Anfrage ergibt folgende sortierte Ausgaberelation:

title	length
Schwarze Galeere	3894000
Blues Power	3816000
Crystal Flame	3617000
Blues At Sunrise	3209000
...	...

Neuere Erweiterungen in SQL: Durch die neueren SQL-Standards bis einschließlich SQL:2016 [Mel16] wurde der Umfang von SQL schrittweise erweitert. Zu diesen Erweiterungen zählen zusätzliche Gruppierungsmöglichkeiten, neue Aggregatfunktionen und die Fensterfunktionen. Diese werden im Folgenden kurz vorgestellt. Für die analytische Auswertung in unterschiedlichen Granularitätsstufen wurde die `GROUP BY`-Klausel um folgende Konstrukte erweitert:

- `GROUP BY GROUPING SETS`: Für eine Menge festgelegter Gruppierungen werden die aggregierten Werte erstellt.
- `GROUP BY ROLLUP`: Ausgehend von einer Attributmenge wird diese von links nach rechts 'aufgerollt', d. h., es wird eine stets feingranularere Aggregation gebildet. Dabei wird auch die Aggregation über den gesamten Datenbestand ausgeführt.
- `GROUP BY CUBE`: Der Operator bildet die Potenzmenge über einer gegebenen Menge von Attributen. Jedes Element aus der Potenzmenge, einschließlich der leeren Menge, entspricht einer Gruppierung, für welche die aggregierten Werte gebildet werden.

Diese Erweiterungen sind – zumindest von außen betrachtet – nur semantischer Zucker, da die Anfragen äquivalent durch mehrere Gruppierungen, welche mittels Mengenvereinigung verknüpft sind, ausgedrückt werden können. Die `CUBE`-Klausel lässt sich beispielsweise durch die Potenzmenge der Attribute des Datenwürfels in einer `GROUPING SETS`-Klausel darstellen, während die `ROLLUP`-Klausel durch die Angabe der einzelnen aufgerollten Gruppen darstellbar ist. `GROUPING SETS` lassen sich durch einzelne Anfragen mit `GROUP BY`-Klauseln darstellen.

Beispiel: Äquivalente SQL-Anfragen mit und ohne *GROUP BY*-Erweiterungen

Die folgenden Anfragen ermitteln die Gesamtlaufzeiten für alle Komponisten, für alle Alben und für alle Kombinationen aus Komponisten und Alben. Zudem wird die Laufzeit des gesamtem Musikarchivs ausgegeben. Die erste Anfrage ist unter Verwendung des `CUBE`-Operators formuliert, die zweite Anfrage unter Verwendung der `GROUPING SETS`-Klausel und die dritte Anfrage verwendet die `UNION`-Klausel sowie Nullwerte. Zur Übersichtlichkeit werden die Relationen zudem nach den *ID*s der Komponisten und Alben sortiert.

```
SELECT cid, aid, sum(length)
FROM tracks
GROUP BY CUBE (cid, aid)
ORDER BY cid, aid
```

```sql
SELECT cid, aid, sum(length)
FROM tracks
GROUP BY GROUPING SETS((cid, aid), (cid), (aid), ())
```

```sql
(
  SELECT cid, aid, sum(length)
  FROM tracks
  GROUP BY (cid, aid)
)
UNION (
  SELECT cid, NULL AS aid, sum(length)
  FROM tracks
  GROUP BY cid
)
UNION (
  SELECT NULL AS cid, aid, sum(length)
  FROM tracks
  GROUP BY aid
)
UNION (
  SELECT NULL AS cid, NULL AS aid, sum(length)
  FROM tracks
)
ORDER BY cid, aid
```

Die folgende Tabelle zeigt einen Ausschnitt aus der Ergebnisrelation. Der erste Teil des Ausschnittes zeigt die jeweiligen Summen der Laufzeiten für die Alben von *Paul Rodgers* (`cid = 429`). Die letzte Zeile des Komponisten zeigt seine Gesamtlaufzeit über alle Alben an.

In der drittletzten und vorletzten Zeile sind die aufsummierten Gesamtlaufzeiten für die Alben *Distant Worlds III* (`aid = 66603`) und *Distant Worlds IV* (`aid = 66604`) zu sehen. Die Gesamtlaufzeit des Musikarchivs beträgt knapp 75 Tage[a], was der letzten Zeile zu entnehmen ist.

cid	aid	sum
...	...	...
429	4922	533000
429	5137	744000
429	5766	1160000
429	5994	583000
429	6161	461000
429		3481000
...	...	...
	66603	3867000
	66604	3439000
		6458235000

[a] $6458235000\,ms = 6458235\,s = 107637,25\,min = 1793,9541\overline{6}\,h = 74,748090\overline{27}\,d$

Zusätzlich zu den in allen Datenbanksystemen vorhandenen fünf Aggregatfunktionen (Minimum, Maximum, Summe, Anzahl, Durchschnitt) und den evtl. noch ergänzenden Median, werden für komplexere statistische Auswertungen viele weitere Aggregate benötigt. Mit den neueren SQL-Standards stehen u. a. Funktionen zur

Funktion	Mathematische Definition	Bedeutung
$var\_pop(X)$	$= \sigma^2 = \frac{\sum_{i=1}^{n}(x_i - \overline{x})^2}{n}$	Varianz des X-Attributs für die gesamte Relation bzw. Partition
$var\_samp(X)$	$= s^2 = \frac{\sum_{i=1}^{n}(x_i - \overline{x})^2}{n-1}$	Varianz des X-Attributs für eine Stichprobe
$stddev\_pop(X)$	$= \sqrt{\sigma^2}$	Standardabweichung des X-Attributs für die gesamte Relation bzw. Partition
$stddev\_samp(X)$	$= \sqrt{s^2}$	Standardabweichung des X-Attributs für eine Stichprobe
$covar\_pop(X, Y)$	$= \frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{X})(y_i - \overline{Y})$	Kovarianz des X-Attributs für die gesamte Relation bzw. Partition
$covar\_samp(X, Y)$	$= \frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{X})(y_i - \overline{Y})$	Kovarianz des X-Attributs für eine Stichprobe
$corr(X, Y)$	$= \frac{covar\_pop(X,Y)}{stddev\_pop(X) * stddev\_pop(Y)}$	Korrelationskoeffizient
$regr\_slope(Y, X)$	$= b = \frac{covar\_pop(X,Y)}{var\_pop(X)}$	Anstieg der Regressionsgeraden
$regr\_intercept(Y, X)$	$= a = \overline{Y} - b * \overline{X}$	y-Achsenabschnitt

Tabelle A.1: Neue Aggregatfunktionen in SQL

Regressions- bzw. Korrelationsanalyse, sowie zur Standardabweichung, Varianz und Kovarianz zur Verfügung. Diese Aggregate sind auch in den meisten Datenbanksystemen umgesetzt. Die neuen Aggregate und deren Bedeutung sind in Tabelle A.1 aufgelistet. In der Tabelle werden mit $\overline{X}$ und $\overline{Y}$ der Mittelwert der X- bzw. Y-Werte bezeichnet und mit n die Größe der Population bzw. Stichprobe. Detaillierte Informationen zum Stand der Umsetzung der Funktionalitäten in einzelnen (Datenbank-)Systemen werden in [Sch16] betrachtet.

Beispiel: SQL-Anfrage mit komplexeren Aggregatfunktionen

Im untenstehenden Beispiel untersucht eine komplexe SQL-Anfrage den Zusammenhang zwischen der Dateigröße `filesize` und der Länge `length` von Musikstück eines Genres. Für jedes Tupel in der Relation `Tracks` wird der Name des jeweiligen Musikstückes `name` zusammen mit dem Anstieg `slope` und den Achsenabschnitt `intercept` der Regressionsgeraden angegeben. Zusätzlich soll gelten, dass ein Genre durch mehr als 20 Titel vertreten sein muss und dass der Korrelationskoeffizient von `filesize` und `length` für das jeweilige Genre größer als 99,999% ist.

```
1 SELECT name, regr_slope(filesize, length) AS slope,
2 regr_intercept(filesize, length) AS intercept
3 FROM tracks JOIN genres ON tracks.gid = genres.gid
4 GROUP BY name
5 HAVING COUNT(*) > 20
6 AND corr(filesize, length) > 0.99999
7 ORDER BY slope DESC
```

Fensterfunktionen: Fensterfunktionen ermöglichen Berechnungen auf einer Menge von Tupeln, die in Beziehung zu einem aktuell betrachtetem Tupel stehen. Dabei werden aggregierte Werte für jedes Tupel gebildet. Durch die fensterbasierte Partitionierung können gleitende Durchschnitte, kumulierte Summen und Ratio-

to-Total-Analysen auf Folgen von Tupeln mit einer festgelegten Ordnung umgesetzt werden. Der Aufbau einer SQL-Anfrage unter Einsatz einer Fensterfunktion entspricht der folgenden Syntax:

```
agg_func(<arg>) OVER(
  [PARTITION BY <X>]
  [ORDER BY <Y> [<WINDOW>]]
)
```

Quellcodeausschnitt A.5: Aufbau einer Fensterfunktion (vereinfachte Darstellung)

Fensterfunktionen können mittels der `OVER`-Klausel innerhalb der `SELECT`-Klausel hinter einer beliebigen Aggregatfunktion `agg_func` realisiert werden. Durch die optionale Partitionierungsklausel `PARTITION BY` erfolgt die Aufteilung eines Dateneingabestroms in Partitionen mit gleichen Werten.

Die `PARTITION BY`-Klausel muss nicht zwingend angegeben werden, insbesondere wenn das Fenster bezüglich der Attributwerte der gesamten Relation berechnet wird. Partitionen lassen sich zudem vor der Anwendung von Aggregatfunktionen ordnen, um laufende Summen oder gleitende Durchschnitte zu realisieren. Ohne diese Verdichtung wird jeder Eingabewert auf einen Ausgabewert abgebildet.

Zusätzlich lassen sich Tupel innerhalb einer Partition durch eine `ORDER BY`-Klausel mit einer Ordnung innerhalb einer Dimension versehen. Dabei liefert der Funktionsaufruf `rank()` den gleichen Rang bei Duplikaten zurück und nummeriert die folgenden Tupel anschließend mit einer Lücke weiter. Die Funktion `dense_rank()` verhält sich wie `rank()`, nummeriert die Tupel hingegen lückenlos.

Das Aggregationsfenster `WINDOW` gibt den Ausschnitt des Eingabestroms an, auf den die Aggregation angewendet werden soll. Für die Größe des Fensters wird ein *von-bis*-Bereich definiert, der durch folgende Syntax festgelegt wird:

```
RANGE BETWEEN
(CURRENT ROW |(<m>|UNBOUNDED) PRECEDING)
AND
(CURRENT ROW | (<n>|UNBOUNDED) FOLLOWING)
```

Quellcodeausschnitt A.6: Aufbau der Window-Klausel

Durch `PRECEDING` und `FOLLOWING` werden die Anzahl der Tupel vor bzw. nach dem aktuellen Tupel festgelegt. Dabei kann das aktuelle Tupel `CURRENT ROW` eines der beiden Grenzwerte bestimmen. Mit `m` bzw. `n` kann eine feste Anzahl an Tupeln bestimmt werden; `UNBOUNDED` erlaubt hingegen die Einbeziehung aller Tupel vor bzw. nach dem aktuellen Tupel in die Berechnung des aggregierten Wertes.

Beispiel: Beispiel für eine Fensterfunktion

Die folgende SQL-Anfrage zeigt die durchschnittliche Laufzeit (`length`) über sechs Musikstücke pro CD (`discnumber`) jedes Albums (`name`). Dabei werden neben dem aktuellen Tupel immer die beiden vorhergehenden und die drei nachfolgenden Tupel berücksichtigt.

```
SELECT name, discnumber, avg(length) OVER(
   PARTITION BY (name, discnumber)
   ORDER BY tracknumber
   ROWS BETWEEN 2 PRECEDING AND 3 FOLLOWING
) AS avgLength
FROM tracks NATURAL JOIN albums
```

A.4 Datalog

Datalog [MTKW18] ist eine deklarative Anfragesprache, bei der Anfragen als prädikatenlogische Formeln erster Ordnung formuliert werden. Aus syntaktischer Sicht erinnert Datalog stark an Prolog. Datalog wird in wissenschaftlichen Arbeiten eingesetzt, wenn eine Anfrage selbst Objekt einer Analyse ist und die Auswertung auf SQL-Ebene in vielen Fällen zu umständlich ist. Das Ergebnis eines Datalog-Programmes ist die Kombination aller Bindungen der Variablen an Werte, welche für die Formel den Wert `wahr` ergeben.

Datalog-Programme bzw. -Anfragen bestehen aus mehreren Regeln. Jede Regel besteht aus einem Kopf und einem Rumpf. Der Kopf entspricht der Projektion aus der Relationenalgebra bzw. der `SELECT`-Klausel aus SQL. Im Rumpf werden mehrere, durch das logische UND verknüpfte, extensionale Prädikate angegeben. Diese entsprechen den Basis- oder den abgeleiteten Relationen bzw. den Selektionsbedingungen.

Attribute werden in Datalog durch ihre Position angegeben, nicht durch ihren Namen. Mehrere Prädikate werden über gleichbenannte Variablen miteinander verknüpft. Taucht eine Variable im Rumpf nur einmal auf, so kann sie durch einen Unterstrich ersetzt werden, wodurch die Lesbarkeit erhöht wird. Datalog-Anfragen lassen sich wie folgt formalisieren:

Definition: Datalog-Anfrage nach [MTKW18]

Sei V eine Menge von Variablen, C eine Menge von Konstanten, R die Menge der vorhandenen Regeln bzw. Fakten und P eine Menge von Prädikaten. Eine konjunktive Datalog-Anfrage Q ist eine Anfrage der Form

$q(v_1, v_2, \ldots, v_n) : -r_1(a_{1,1}, a_{1,2}, a_{1,m_1}), r_2(a_{2,1}, a_{2,2}, a_{2,m_2}), \ldots r_k(a_{k,1}, a_{k,2}, a_{k,m_k}), p_1, p_2, \ldots, p_j,$

wobei $v_i \in V$, $r_i \in R$, $a_{i,j} \in V \cup C$, $\forall v \in V : \exists i, j : a_{i,j} = v$ und $\forall p \in P$: p hat entweder die Form $v_i \theta c$ oder $v_i \theta v_j$ mit $\theta \in \{<, =, >\}$.

Notation: Im Bereich von Query Containment wird insbesondere Datalog als Anfragesprache zur Formulierung von Sichten verwendet. Zum besseren Verständnis werden an dieser Stelle die Basiskonstrukte von Datalog kurz vorgestellt.

Der Aufbau des Schemas für die Basisrelationen entspricht dem des relationalen Modells; es wird jedoch auf die explizite Kennzeichnung von Primär- und Fremdschlüsseln verzichtet. Eine Verknüpfung mehrerer Relationen erfolgt durch die Verwendung gleicher Attributnamen, wie es beim natürlichen Verbund der Fall ist.

Beispiel: Ausschnitt des Schemas der Musikdatenbank in Datalog-Notation

Das Schema der Musikdatenbank (Ausschnitt) sieht in Datalog-Notation wie folgt aus:

```
1 tracks(TID, AID, CID, GID, Title, Length, ...)
2 albums(AID, Aname)
3 composers(CID, Cname)
4 genres(GID, Gname)
```

Ein Datalog-Programm besteht aus Fakten, welche die Tupel in den Basisrelationen repräsentieren, und Regeln, welche als Sichten neue Informationen aus den Basisrelationen bzw. anderen Regeln ableiten. Im Folgenden sind Fakten aus dem Musikbeispiel dargestellt. Alphanumerische Werte werden in einfachen Hochkommata angegeben, nummerische Werte ohne diese. Attribute von Primär- und Fremdschlüsseln werden als kleingeschriebene, alphanumerische Werte ohne Hochkommata angegeben.

Beispiel: Fakten der Musikdatenbank in Datalog-Notation

```
1 tracks(t6019, a602, c62, g1, 'Dancing Mad', 646, ...).
2 tracks(t6053, a604, c62, g1, 'Torn from the Heavens', 267, ...).
3 albums(a602, 'Distant World II').
```

```
albums(a604, 'Distant Worlds IV').
composers(c62, 'Nobuo Uematsu').
genres(g1, 'Original Soundtrack').
```

Datalog-Regeln leiten neue Fakten ab, wenn die im Regelrumpf durch Kommata separierten, angegebenen Fakten existieren. Die Struktur des neuen Fakts wird durch den Kopf der Regel beschrieben. Werden in der Regel Variablen verwendet, um beispielsweise einen Verbund zu realisieren, werden diese als alphanumerische Werte, die mit einem Großbuchstaben beginnen, angegeben. Kopf und Rumpf einer Datalog-Regel werden durch das Implikationssymbol ': −' voneinander getrennt.

Beispiel: Datalog-Regel

Die folgende Regel gibt alle Komponisten mit dem dazugehörigen Album aus, sofern der dazugehörige Track zum Genre *Videospiel-OST* gehört.

```
ostComposerAlbum(Aname, Cname) :-
  tracks(_, AID, CID, GID, _, ...),
  genres(GID, 'Videospiel-OST'),
  albums(AID, Aname),
  composers(CID, Cname).
```

Nicht benötigte Attribute werden innerhalb von Regeln durch einen Unterstrich ausgeblendet. Die Variablen werden zur Abarbeitung der Regel durch Fakten aus den verwendeten Relationen ersetzt. Dabei werden gleichbenannte Variablen in jeder Relation mit dem gleichen Wert belegt. Wird eine gültige Kombination gefunden, wird ein neuer Fakt generiert, der durch den Regelnamen und die Attributbelegung beschrieben ist. Die Menge der so erzeugten Fakten kann als virtuelle Relation aufgefasst werden.

Beispiel: Datalog-Regel in SQL-Syntax

Als SQL-DDL-Anweisung kann die obige Datalog-Regel wie folgt formuliert werden[a]:

```
CREATE VIEW ostComposerAlbum AS(
  SELECT Aname, Cname
  FROM tracks
    JOIN genres USING (gid)
    JOIN albums USING (aid)
    JOIN composers USING (cid)
  WHERE Gname = 'Videospiel-OST'
```

[a] Es werden, abweichend vom laufenden Beispiel, unterschiedliche *Namen* in den Dimensionsrelationen verwendet. Dies erleichtert die Lesbarkeit des Beispiels.

Anfragen können über die bestehenden Fakten bzw. Regeln ausgeführt werden. Eine Anfrage beginnt mit dem Symbol ?− gefolgt von der Angabe einer Basisrelation bzw. Regel, in welche zusätzlich die Variablen durch Konstanten ersetzt werden können.

Beispiel: Anfrage in Datalog

Soll aus der zuvor definierten Sicht *ostComposerAlbum* nur die Alben des Komponisten *'Nobuo Uematsu'* ausgegeben werden, so ist es ausreichend, im Kopf der Sicht das Attribut *Cname* durch *'Nobuo Uematsu'* zu ersetzen:

```
?- ostComposerAlbum(Aname, 'Nobuo Uematsu').
```

Nach der Auswertung werden alle Fakten zurückgegeben, die auf die Regel `ostComposerAlbum` und der gegebenen Variablenbelegung zutreffen. Duplikate werden, wie in der relationalen Algebra, entfernt:

```
ostComposerAlbum('Distant Worlds', 'Nobuo Uematsu').
ostComposerAlbum('Distant Worlds II', 'Nobuo Uematsu').
ostComposerAlbum('Distant Worlds III', 'Nobuo Uematsu').
ostComposerAlbum('Distant Worlds IV', 'Nobuo Uematsu').
```

Erweitere Konzepte: Seit der Einführung von Datalog wurden zahlreiche Erweiterungen eingeführt (Übersicht nach [MTKW18]):

- Durch die Verwendung des Schlüsselwortes `not` vor einem Prädikat kann dieses negiert werden,
- Arithmetik und Vergleichsoperatoren werden durch speziell ausgezeichnete Prädikate unterstützt. Dazu zählen beispielsweise der Summen-Operator `sum(A,B,C)` mit $A + B = C$ oder der Größer-Als-Operator `gt(X,Y)` mit $X > Y$,
- Aggregationen auf gruppierten Werten,
- Unterstützung von EGDs (Equality Generating Dependencies) und TGDs (Tuple Generating Dependencies) durch Einführung von Existenzquantoren im Regelkopf,
- Datentypen,
- Bedingungen wie Primärschlüssel und referentielle Integrität sowie
- Unterstützung von objektorientierten Konzepten.

Diese Erweiterungen ermöglichen die Lösung von Query-Containment-Problemen für ein breiteres Spektrum an Anfrageklassen. Dazu müssen die Regeln des Datalog-Programms schrittweise ausgewertet werden.

Inkrementelle Auswertung von Datalog-Programmen: Ein Datalog-Programm besteht aus einer Menge von Regeln. Das Ergebnis des Programms ist die Menge der von den Regeln abgeleiteten Grundatomen. Die Auswertung erfolgt dabei durch die wiederholte Anwendung der Regeln, bis keine neuen Atome erzeugt werden können.

Eine mögliche Effizienzverbesserung zur Auswertung von Datalog-Programmen ist die inkrementelle Auswertung (siehe dazu beispielsweise [ULS08, Kapitel 12]). Dabei werden nach der initialen Auswertung im rekursiven Schritt nur diejenigen Regeln untersucht, die mindestens ein neu erzeugtes Prädikat unter ihren Teilzielen besitzen. Diese Regel wird anschließend unter Berücksichtigung des Prädikats und der bereits bekannten Fakten ausgewertet. Dadurch wird der Suchraum eingeschränkt, ohne dass neue Fakten übersehen werden.

Anwendung von Datalog: Datalog wird neben der Analyse und Optimierung von Datenbankanfragen auch im Compilerbau zur interprozeduraler Datenflussanalyse [ULS08, Kapitel 12] verwendet. Diese wird neben der Optimierung von Programmflüssen auch zum Erkennen von Softwareschwachstellen eingesetzt.

A.5 Edge Computing: Fallbeispiel TinyDB

Als eine der ersten Umsetzungen von Edge Computing kann das TinyDB-System, insbesondere der darin enthaltene *Tiny AGgregation Service* (TAG) [MFHH02], angesehen werden. TinyDB [MFHH05] zählt nach der Klassifikation von Datenbankmanagementsystemen nach Rosenmüller et al. [RLAS07] zu den sogenannten Micro-DBMS.

Diese zeichnen sich dadurch aus, dass als Datenmodell meist persistente Tabellen genutzt werden. Anfragen werden meist über eine eigene API gestellt, wobei in der Regel nur eine einzelne Relation angefragt wird. Micro-DBMS werden typischerweise in mobilen Geräten geringer Leistungsfähigkeit, wie Mobiltelefonen, eingebetteten Systemen und Smartcards, eingesetzt.

TinyDB wird primär für den Aufbau von Sensornetzwerken eingesetzt. Die einzelnen Knoten in diesen Netzwerken zeichnen sich dadurch aus, dass sie über eine sehr geringe Rechenleistung sowie geringen Arbeits- und Festplattenspeicher verfügen. Da die Knoten meist kabellos betrieben werden, sind sie im Normalfall auch batteriebetrieben. Zur Reduzierung des Stromverbrauchs leitet TinyDB nur aggregierte Werte über alle Knoten an den dazugehörigen Vaterknoten weiter. Beispielsweise können die Daten mehrerer räumlich benachbarter Sensoren zu einem Durchschnittswert zusammengefasst werden, wobei jede Aggregationsstufe die Daten in einem gröberen Detailgrad darstellt.

Anfragesprache: TinyDB verfügt über eine SQL-ähnliche Anfragesprache. Die Syntax ist im nachfolgenden Quellcodeausschnitt kurz skizziert:

```
1 SELECT <attrs>, agg(<expr>)
2 FROM <table>
3 WHERE <selPreds>
4 GROUP BY <attrs>
5 HAVING <havingPreds>
6 EPOCH DURATION <i>
```

Quellcodeausschnitt A.7: SQL-Syntax in TinyDB

Die `EPOCH DURATION`-Klausel bietet dem Nutzer die Möglichkeit, anzugeben, in welchem Intervall die Daten periodisch angefragt werden sollen. Der Wert `i` gibt dabei an, wie viel Zeit zwischen den einzelnen Übertragungen vergehen soll. Die Bedeutung der weiteren Klauseln entspricht dem SQL-Standard.

Aggregationen: TinyDB bietet ein spezielles Verfahren zur Aggregation von Daten in Sensornetzwerken an, den sogenannten Tiny AGgregation (TAG) Service [MFHH02]. In TAG besteht eine Aggregatfunktion aus drei Komponenten: Eine Initialisierungsfunktion `i` mit

$$i(a) = (a, c) \tag{A.1}$$

erzeugt aus einem lokal erzeugtem Wert `a` ein Tupel `(a,c)`, der eine zusätzliche Konstante `c` enthält. Die Vereinigungsfunktion `f` mit

$$t = f(t_1, t_2) \tag{A.2}$$

fasst zwei Tupel `t_1` und `t_2` zu einem neuen Tupel `t` zusammen. Durch die Evaluierungsfunktion `e` wird aus einem Vektor ein einzelner Wert erzeugt.

Für die Berechnung des Durchschnittes erzeugt `e` jeweils ein Tupel mit den Werten `(a,1)`, wobei `a` der lokale Wert ist und `1` für die initiale Anzahl an Werten steht. Die Funktion `f` summiert anschließend die Werte s und die Anzahlen c auf:

$$f((s_1, c_1), (s_2, c_2)) = (s_1 + s_2, c_1 + c_2). \tag{A.3}$$

Durch die Evaluierungsfunktion wird abschließend die Division beider Werte ausgeführt:

$$e((s, c)) = s/c. \tag{A.4}$$

In TinyDB stehen neben den bekannten Aggregatfunktionen `MAX`, `MIN`, `COUNT`, `SUM` und `AVERAGE` als zusätzliche Funktionen `HISTOGRAM`, `MEDIAN` und `COUNT DISTINCT` zur Verfügung. Die beiden letztgenannten sind allerdings holistische Aggregate, wodurch sie nicht verteilt berechnet werden können.

Verteilte Berechnung von Aggregaten: Sensoren und weitere Rechner bilden ein Netzwerk zur Berechnung von aggregierten Werten. Anfragen werden von einer zuvor ausgezeichneten, zentralen Stelle (dem Wurzelknoten des Netzwerkes) aus gestellt. Die von Sensoren erzeugten Werte werden nicht an zentraler Stelle berechnet, sondern so früh wie möglich verarbeitet. Durch die deklarative Formulierung der Anfrage muss der Programmierer bzw. Nutzer sich keine Gedanken über die verteilte Berechnung machen, sondern überlässt dem System die eigentliche Ausführung und Optimierung des Programmes hinsichtlich Latenz und Stromverbrauch.

Anfrageverarbeitung: Für die Anfrageverarbeitung in TinyDB wird die Topologie des Sensornetzwerks als hierarchische Struktur angesehen. Zur Veranschaulichung gehen wir im Folgenden von Baumstrukturen aus.

An die Wurzel des Baumes wird eine Aggregatanfrage, im folgenden Beispiel eine Berechnung der Summe aller Werte, gestellt. Der Wurzelknoten kennt das globale Schema, um die Anfragen an die untergeordneten Kindknoten rekursiv weiterzuleiten.

Die Kinderknoten aggregieren jeweils ihre eigenen Werte. Die Blattknoten senden ihr eigenes Ergebnis weiter an den übergeordneten Vaterknoten. Der Vaterknoten aggregiert das Ergebnis seiner Kinder und ggf. seine eigenen Werte zu einem neuen Wert. Die Aggregate werden rekursiv nach oben weitergeleitet, bis die Wurzel erreicht ist. Das aggregierte Ergebnis im Wurzelknoten entspricht dem Ergebnis der Anfrage.

Die Anfrage wird nach einer Epoche erneut gestellt. TinyDB ist so konzipiert, dass beim Ausfall bzw. Hinzufügen eines Sensors nur ein Teil der Daten aggregiert wird. Durch diese Ad-hoc-Eigenschaft können dem Netzwerk dynamisch einzelne Sensorknoten beitreten bzw. dieses wieder verlassen. Selektionsprädikate, die in den `WHERE`- und `HAVING`-Klauseln angegeben wurden, werden an die Kindknoten übertragen, um Daten vorzuselektieren.

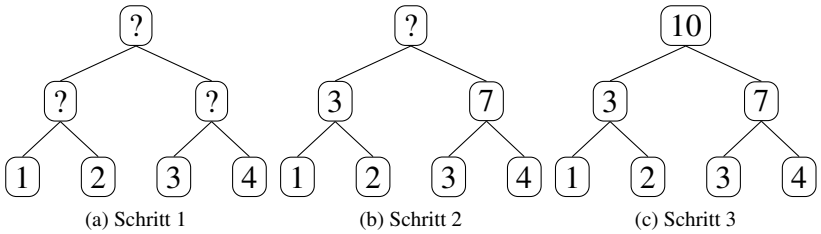


Abbildung A.1: Schrittweise Aggregation in TinyDB innerhalb einer Epoche

Vorteile von TinyDB: TinyDB bietet durch die vorgelagerten Aggregationen die Möglichkeit, Datensparsamkeit konsequent umzusetzen, da hierdurch weniger Daten im Netzwerk übertragen werden. Gleichzeitig wird der Stromverbrauch an den einzelnen Knoten reduziert, was zu einer längeren Laufzeit der batteriebetriebenen Netzwerkknoten führt.

Nachteile von TinyDB: TinyDB dient vorrangig zur Aggregation von Daten. Da das System so ausgelegt ist, dass es selbst wenig Ressourcen benötigt, werden einige Operatoren der relationalen Algebra nicht unterstützt. So verzichtet TinyDB auf Verbundoperatoren. Durch die Architektur des Systems kommt es zudem zu Performanceeinbußen, speziell dann, wenn einzelne Knoten im Netzwerk zu sehr ausgelastet sind und die Daten erst mit Verzögerung weiterleiten. Die Berechnungen im Vaterknoten sind somit erst bei Vorliegen aller Teilergebnisse möglich. Es muss dabei zusätzlich darauf geachtet werden, dass die Gesamtrechenzeit nicht die Dauer einer Epoche übersteigt, da ansonsten Daten aus verschiedenen Epochen miteinander verrechnet werden könnten.

A.6 Datenschutz – Gesetzliche Grundlagen

Für den Begriff *Datenschutz* existieren viele unterschiedliche Definitionen, die teilweise durch kulturelle und gesetzliche Besonderheiten geprägt sind. Die Grundlagen des Datenschutzes in Deutschland entstammen dem Urteil des Bundesverfassungsgerichtes zum Volkszählungsgesetz von 1983 [Bun83], welche den Datenschutz als allgemeines Persönlichkeitsrecht ansehen und sich dabei im ersten Leitsatz auf das Grundgesetz berufen:

> **Zitat:** BVerfG, Urteil vom 15.12.1983: 1. Leitsatz [Bun83]
>
> Unter den Bedingungen der modernen Datenverarbeitung wird der Schutz des Einzelnen gegen unbegrenzte Erhebung, Speicherung, Verwendung und Weitergabe seiner persönlichen Daten von dem allgemeinen Persönlichkeitsrecht des Art. 2 Abs. 1 GG in Verbindung mit Art. 1 Abs. 1 GG umfaßt. Das Grundrecht gewährleistet insoweit die Befugnis des Einzelnen, grundsätzlich selbst über die Preisgabe und Verwendung seiner persönlichen Daten zu bestimmen.

Die Richter befürchten insbesondere, dass *abweichende Verhaltensweisen jederzeit notiert und als Information dauerhaft gespeichert [...] werden* und dass die Betroffenen *versuchen, nicht durch solche Verhaltensweisen aufzufallen [Bun83].* Ferner fordert das Bundesverfassungsgericht im dritten Leitsatz, dass Informationssysteme zwischen Datenverarbeitung für statistische Zwecke, bei denen Daten in anonymer Form erheben werden müssen, und der Verarbeitung für vordefinierte, spezifische Zwecke unterschieden werden muss.

> **Zitat:** BVerfG, Urteil vom 15.12.1983: 3. Leitsatz [Bun83]
>
> Bei den verfassungsrechtlichen Anforderungen an derartige Einschränkungen ist zu unterscheiden zwischen personenbezogenen Daten, die in individualisierter, nicht anonymer Form erhoben und verarbeitet werden, und solchen, die für statistische Zwecke bestimmt sind.
>
> Bei der Datenerhebung für statistische Zwecke kann eine enge und konkrete Zweckbindung der Daten nicht verlangt werden. Der Informationserhebung und Informationsverarbeitung müssen aber innerhalb des Informationssystems zum Ausgleich entsprechende Schranken gegenüberstehen.

Die Richter des Bundesverfassungsgerichtes verweisen in ihrem Urteil auf die ersten beiden Artikel des deutschen Grundgesetzes:

> **Zitat:** Grundgesetz [Deu49], Artikel 1
>
> (1) Die Würde des Menschen ist unantastbar. Sie zu achten und zu schützen ist Verpflichtung aller staatlichen Gewalt.

> **Zitat:** Grundgesetz [Deu49], Artikel 2
>
> (1) Jeder hat das Recht auf die freie Entfaltung seiner Persönlichkeit, soweit er nicht die Rechte anderer verletzt und nicht gegen die verfassungsmäßige Ordnung oder das Sittengesetz verstößt.

Ferner können auch Artikel 10 (Post- und Fernmeldegeheimnis) sowie Artikel 13 (Unverletzlichkeit der Wohnung) des deutschen Grundgesetzes [Deu49] als Pfeiler für das Recht auf eine eigenständige Datenschutzregelung aufgefasst werden. Datenschutz wird in unserer heutigen Gesellschaft als Grundrecht angesehen. Das Recht auf informationelle Selbstbestimmung regelt die Befugnis des Einzelnen, selbst über die Veröffentlichung und Verarbeitung seiner personenbezogenen Daten zu entscheiden. Andere Einzelpersonen, staatliche Einrichtungen und Firmen dürfen somit nicht ohne Weiteres über die Daten anderer entscheiden.

In diesem Unterabschnitt werden kurz die wichtigsten gesetzlichen Grundlagen zum Datenschutzrecht vorgestellt. Die folgenden Zitate beziehen sich, sofern nicht anders angegeben, auf die alte[6] Fassung des Bundesdatenschutzgesetzes (BDSG) [Deu15] bzw. die Datenschutz-Grundverordnung (DSGVO) [Eur16].

Das BDSG richtet sich speziell an öffentliche Stellen des Bundes und des Landes; für letztere nur, wenn kein eigenes Landesrecht vorhanden ist. Zudem regelt das BDSG die automatisierte Verarbeitung von Daten in

[6] In der neuen Fassung des BDSG[Deu18] gelten die Bestimmungen weiterhin, sind jedoch umständlicher beschrieben.

nicht-öffentlichen Stellen. Im Gegensatz dazu regeln die Landesdatenschutzgesetze (LDSG), wie das LDSG von Mecklenburg-Vorpommern [Lan02], den Umgang mit Daten in Landesbehörden, öffentlich-rechtlichen Einrichtungen sowie privat-rechtlichen Vereinigungen, welche öffentliche Aufgaben wahrnehmen. Für eine ausführliche Erläuterung und einen Vergleich zwischen der DSGVO und dem Bundesdatenschutzgesetz sei auf [VvdB18] verwiesen.

Weiterführende Forderungen an Einwilligungserklärungen, Unterrichtungspflichten sowie technische und organisatorische Maßnahmen hinsichtlich der IT-Sicherheit und datenschutzkonformen Systemgestaltung werden näher im Telemediengesetz (TMG) und Telekommunikationsgesetz (TKG) weiter ausgeführt. Diese Gesetze ändern an den Kernprinzipien (wie z. B. Zweckbindung, Datensparsamkeit und Anonymisierung) nur wenig, sondern legen nur die Rahmenbedingungen für deren Anwendung genauer fest. Aus diesem Grund wird an dieser Stelle nicht detaillierter auf diese Gesetze eingegangen.

Anwendungsgebiet: Wie der Name zunächst fälschlicherweise vermuten lässt, regelt der Datenschutz *nicht* den Schutz der Daten, sondern vielmehr den Schutz des *Menschen* vor Daten. Dies wird im Bundesdatenschutzgesetz in der Fassung von 2015 [Deu15] genauer ausgeführt:

> **Zitat:** BDSG [Deu15], §1: Zweck und Anwendungsbereich des Gesetzes, Absatz 1
>
> Zweck dieses Gesetzes ist es, den Einzelnen davor zu schützen, dass er durch den Umgang mit seinen personenbezogenen Daten in seinem Persönlichkeitsrecht beeinträchtigt wird.

In der Neufassung des Bundesdatenschutzgesetzes [Deu18] nach Inkrafttreten der DSGVO ist dieser Zweck leider entfallen. Regelungen zum Datenschutz werden weltweit in unterschiedlichen Ausmaßen und Ausprägungen getroffen. Um innerhalb der Europäischen Union gemeinsame Standards zu schaffen, wurde am 24. Oktober 1995 die Richtlinie 95/46/EG [Eur95] durch das Europäische Parlament erlassen:

> **Zitat:** Richtlinie 95/46/EG [Eur95], Artikel 3
>
> 3) Für die Errichtung und das Funktionieren des Binnenmarktes, der gemäß Artikel 7a des Vertrags den freien Verkehr von Waren, Personen, Dienstleistungen und Kapital gewährleisten soll, ist es nicht nur erforderlich, daß personenbezogene Daten von einem Mitgliedstaat in einen anderen Mitgliedstaat übermittelt werden können, sondern auch, daß die Grundrechte der Personen gewahrt werden.

Ein Kritikpunkt an der Richtlinie 95/46/EG ist, wie obiges Zitat vermuten lässt, die Fixierung auf die wirtschaftlichen Aspekte statt auf die Rechte der einzelnen Person. Dieser Kritikpunkt wird auch durch die neue Datenschutz-Grundverordnung (DSGVO [Eur16]) nicht vollständig beseitigt; der Schwerpunkt des Gesetzes verlagert sich aber in Richtung des Persönlichkeitsrechtes:

> **Zitat:** Erwägungsgrund 2 der DSGVO [Eur16]
>
> (2) Die Grundsätze und Vorschriften zum Schutz natürlicher Personen bei der Verarbeitung ihrer personenbezogenen Daten sollten gewährleisten, dass ihre Grundrechte und Grundfreiheiten und insbesondere ihr Recht auf Schutz personenbezogener Daten ungeachtet ihrer Staatsangehörigkeit oder ihres Aufenthaltsorts gewahrt bleiben. Diese Verordnung soll zur Vollendung eines Raums der Freiheit, der Sicherheit und des Rechts und einer Wirtschaftsunion, zum wirtschaftlichen und sozialen Fortschritt, zur Stärkung und zum Zusammenwachsen der Volkswirtschaften innerhalb des Binnenmarkts sowie zum Wohlergehen natürlicher Personen beitragen.

Zweckbindung und Datensparsamkeit: Zu diesen Einzelangaben zählen nach [BDH+13] Informationen wie der Name, das Alter, die Anschrift und Angaben zu Religion und Familienstand, aber auch Einkommens- und Vermögensverhältnisse, Versicherungsnummern, Vorstrafen, der momentane Aufenthaltsort und sogar genetische Daten und Röntgenbilder. Schutzgegenstand sind jedoch nur Daten, die nicht allgemein bekannt bzw. aus frei zugänglichen Quellen beziehbar sind. Beispielsweise sind Profildaten von Facebook, deren Sichtbarkeit auf *öffentlich* gesetzt wurden vom Datenschutz ausgenommen, da sie freiwillig veröffentlicht wurden. Sind die Profildaten,

Bilder und Beiträge hingegen auf *nur Freund* oder *nur ich* gesetzt, so sind die Daten privat und fallen unter die DSGVO:

> **Zitat:** Artikel 5, Absatz 1b DSGVO [Eur16]: Grundsätze für die Verarbeitung personenbezogener Daten
> Personenbezogene Daten müssen für festgelegte, eindeutige und legitime Zwecke erhoben werden und dürfen nicht in einer mit diesen Zwecken nicht zu vereinbarenden Weise weiterverarbeitet werden; eine Weiterverarbeitung für im öffentlichen Interesse liegende Archivzwecke, für wissenschaftliche oder historische Forschungszwecke oder für statistische Zwecke gilt gemäß Artikel 89 Absatz 1 nicht als unvereinbar mit den ursprünglichen Zwecken ('Zweckbindung');

Zwei der wichtigsten Forderungen des Datenschutzes, die Datensparsamkeit und Datenvermeidung, werden in §3a der alten Fassung des BDSGs [Deu15] als dessen zentrale Regelung vorgestellt:

> **Zitat:** BDSG [Deu15], §3a: Datenvermeidung und Datensparsamkeit
> Die Erhebung, Verarbeitung und Nutzung personenbezogener Daten und die Auswahl und Gestaltung von Datenverarbeitungssystemen sind an dem Ziel auszurichten, so wenig personenbezogene Daten wie möglich zu erheben, zu verarbeiten oder zu nutzen. Insbesondere sind personenbezogene Daten zu anonymisieren oder zu pseudonymisieren, soweit dies nach dem Verwendungszweck möglich ist und keinen im Verhältnis zu dem angestrebten Schutzzweck unverhältnismäßigen Aufwand erfordert.

Die DSGVO geht auf diesen Aspekt nur kurz ein:

> **Zitat:** Artikel 5, Absatz 1c DSGVO [Eur16]: Grundsätze für die Verarbeitung personenbezogener Daten
> Personenbezogene Daten müssen dem Zweck angemessen und erheblich sowie auf das für die Zwecke der Verarbeitung notwendige Maß beschränkt sein ('Datenminimierung');

Die Datenvermeidung hat das Ziel, die größtmögliche Datensparsamkeit zu erreichen, indem bereits vor der Erhebung feststeht, welche Daten zur weiteren Verarbeitung benötigt werden. Sofern möglich, sollten keine oder möglichst wenig personenbezogene Daten verarbeitet werden. Beiden Aspekten wird gegenwärtig nur wenig Beachtung geschenkt [PS17], da die Formulierungen sehr vage gehalten wurden. In den technischen und organisatorischen Maßnahmen der ITGrundschutz-Kataloge [Mün16] des Bundesamtes für Sicherheit in der Informationstechnik werden diesbezüglich auch keine Aussagen getroffen.

Eine Sonderrolle nimmt hierbei die *Datenverarbeitung zu historischen oder statistischen Zwecken sowie zum Zwecke der wissenschaftlichen Forschung* nach Artikel 83 DSGVO [Eur16] ein, welche durch die Erwägungsgründe 156 bis 159 der DSGVO bestärkt werden: Der Nutzung von Archivdaten bzw. der Erhebung neuer Daten werden einer besonderen Bedeutung beigemessen, da durch die Forschung in Medizin und Sozialwissenschaften das Leben der Menschen nachhaltig verbessert wird. Daher wird für diese Zwecke der Datenschutz gelockert, allerdings nicht vollständig aufgehoben. So müssen Daten weiterhin möglichst sparsam erhoben und anschließend anonymisiert werden, jedoch nur so stark (z. B. durch Pseudonymisierung), dass auch zukünftige Auswertungen ermöglicht werden. Gleiches gilt auch für die Verarbeitung sensibler Daten zu Zwecken der öffentlichen Gesundheit (Erwägungsgrund 54).

Dies trifft insbesondere dann zu, wenn die Daten *für die Darstellung von Forschungsergebnissen oder zur Unterstützung der Forschung notwendig [sind]* [Eur16]. Bei der Erforschung und Entwicklung neuer Methoden und Systeme, wie beispielsweise von Assistenzsystemen, können somit durchaus zunächst mehr Daten als erforderlich gesammelt werden – sofern das konkrete, später verwendete System die technischen und organisatorischen Maßnahmen zur Einhaltung des Datenschutzes implementiert. Auf diese Instrumente wird im Folgenden kurz eingegangen.

A.7 Datenschutz – Datensouveränität

Kontrovers diskutiert wird das Konzept der *Datensouveränität* als Alternative zur Datensparsamkeit bzw. anderen Datenschutzaspekten. Datensouveränität bezeichnet nach der Definition des Deutschen Ethikrates [Deu17c] als

”eine den Chancen und Risiken von Big Data angemessene verantwortliche informationelle Freiheitsgestaltung”. Die existierenden Regeln des Datenschutzes sollen dabei nicht außer Kraft gesetzt werden, sondern vielmehr flexibler gestaltet werden. Datenverarbeitende Stellen dürfen prinzipiell personenbezogene Daten untereinander austauschen, sofern dies der Zweckbindung unterliegt und die betroffene Person nichts dagegen hat. Der Betroffene kann, beispielsweise in einem zentralen Register, einerseits durch Voreinstellungen im Vorfeld festlegen, welche Daten überhaupt übertragen werden dürfen, anderseits kann er sich auch über das Register informieren, welche Daten für welche Zwecke übermittelt wurden und ggf. eingreifen.

Der Vorteil der Datensouveränität für den Betroffenen liegt darin, dass er seine Daten nicht mehrfach an mehreren Stellen erneut eingeben muss und er somit entlastet wird. Die Datenverarbeiter, z. B. Behörden, können die Daten einfach untereinander austauschen, wodurch dort ein geringer administrativer Austausch entsteht und auch die Datenintegration erleichtert wird. Insbesondere politische Parteien plädieren (z. B. in [BA18]) für die Einführung eines solchen *Digitalen Bürgerkontos*. Kritiker[7] fürchten hingegen eine Aushebelung des Datenschutzes zu Gunsten der Privatwirtschaft und Lobbyverbänden. Unter den Kritikern befinden sich aber auch solche, die durch die Datensouveränität eine Abschaffung der Bevormundung seitens der Konzerne ersehnen und das Potenzial des Informationsaustausches erkennen[8].

Der Deutsche Ethikrat stellt zum Interessenausgleich in [Deu17c] mehrere Leitlinien auf, welche für die erfolgreiche Einführung und Durchsetzung der Datensouveränität notwendig sind. Zu den Kernpunkten gehören:

- **Potenziale erschließen**: u. a. erleichterte Datenintegration (Interoperabilität, kooperatives Forschungsdatenmanagement) und Daten- und Forschungsqualität fördern.
- **Datenhoheit bewahren**: u. a. Datenhoheit sichern, Einwilligungsmodelle etablieren, Privacy-by-Default und Datentransparenz.
- **Gerechtigkeit und Solidarität sichern**: u. a. Diskriminierung verhindern und Widerspruch ermöglichen.
- **Verantwortung und Vertrauen fördern**: u. a. Anonymisierung, Datenschutzbeauftragte einsetzen, Kodex erarbeiten, Kompetenz bzgl. Umgang mit Daten stärken.

Eine Bewertung dieses Ansatzes ist nicht Teil dieser Arbeit. Wichtig ist vielmehr das Fazit des Ethikrates, welches im abschließenden Sondervotum gezogen wird:

> **Zitat:** Deutscher Ethikrat zur Datensouveränität [Deu17c]
>
> Der Datenschutz bedarf daher einer präzisen gesetzlichen Regelung und das Bundesdatenschutzgesetz einer Präzisierung mit geeigneten Schutzmechanismen und Gestaltungsstrategien, also **einer Bestätigung und Ausweitung, die Datensparsamkeit und Zweckbindung beinhalten**.

Auch die Gesellschaft der Informatik (GI) erkennt die Datensouveränität als zentralen Bestandteil bei der Entwicklung von Informationssystemen an. In ihren ethischen Leitlinien [Ges18] fordert die GI von ihren Mitgliedern, dass die Systeme entsprechend ausgestaltet werden:

> **Zitat:** Ethische Leitlinien der GI [Ges18], Artikel 11: Ermöglichung der Selbstbestimmung
>
> Das GI-Mitglied wirkt darauf hin, die von IT-Systemen Betroffenen an der Gestaltung dieser Systeme und deren Nutzungsbedingungen angemessen zu beteiligen. Dies gilt insbesondere für Systeme, die zur Beeinflussung, Kontrolle und Überwachung der Betroffenen verwendet werden können.

[7] `https://netzpolitik.org/2016/angela-merkel-hat-gehoert-dass-sie-datenschutz-jetzt-datensouveraenitaet-nennen-soll/`, zuletzt aufgerufen am 05.01.2022.

[8] `http://www.spiegel.de/netzwelt/web/datenschutzgrundverordnung-dsgvo-wer-macht-mir-die-geileren-vorschriften-a-1206979.html`, zuletzt aufgerufen am 05.01.2022.

A.8 Anonymisierungsverfahren und -systeme

A.8.1 k-Anonymität

Ein spezielles Verfahren, um k-Anonymität zu erreichen, ist die Mikroaggregation [DMMS06]. Dabei werden statistische Daten aus einer Datenbank so partitioniert und aggregiert, dass einerseits die Privatsphäre des Einzelnen gewahrt bleibt, jedoch keine nennenswerte Auswirkung auf die vorgesehene statistische Analyse erfolgt. Das von Domingo-Ferrer et al. vorgestellt Verfahren bildet mehrere Partitionen mit jeweils einer variablen Größe von k (Minimum für die Wahrung der Anonymität) bis $2k - 1$ Einträgen (Maximum für einen geringen Informationsverlust). Die Einträge in jeder Partition werden dabei so gewählt, dass diese sehr ähnlich sind. Durch Aggregation wird ein Zentroid für jede Partition bestimmt, welcher als Stellvertreter für die gesamte Gruppe steht.

Die Schwierigkeit der Mikroaggregation besteht in der geeigneten Wahl der Partitionen. Einerseits muss der Fehler in der Aggregation der Daten über den gesamten Datenbestand durch die Wahl der Stellvertreter möglichst gering gehalten werden. Andererseits muss gleichzeitig die Anonymität der personenbezogenen Daten sichergestellt werden. Die in [DMMS06] vorgestellte Heuristik liefert dazu eine gute Näherung bei gleichzeitig vertretbarem Aufwand von $O(\frac{n^2}{2})$, wobei n die Anzahl der Attributwerte in der Datenbank ist.

Das Problem, eine optimale Anonymisierung unter Bewahrung der Nützlichkeit zu finden, ist NP-vollständig [MW04]. Die in [DMMS06] und [MW04] vorgestellten Verfahren helfen dabei, eine gute Anonymisierung mit vertretbaren Aufwand zu generieren.

Spezielle Erweiterungen der k-Anonymität: In den letzten Jahren wurden für k-Anonymität viele Erweiterungen entwickelt. An dieser Stelle stellen wir eine Auswahl dieser verschiedenen Ansätze in Kurzform vor. Mit *Mondrian Multidimensional K-Anonymity* [LDR06] wurde ein Verfahren für die mehrdimensionale Generalisierung entwickelt, welches die parallele Generalisierung mehrerer Attribute ermöglicht. Dadurch wird ein geringerer Informationsverlust erreicht, da einzelne Attributwerte weniger stark generalisiert werden müssen.

In [GAZ+14] werden mit *k^m-Anonymity* dynamische Hierarchien eingeführt. Durch eine dynamische Überprüfung, wie viele Kindknoten bei einem Generalisierungsschritt zusammengefasst werden, wird ebenfalls der Informationsverlust minimiert. Als Technik kommt im Hintergrund der aus dem Data Mining bekannte A-priori-Algorithmus [AIS93] zum Einsatz.

In [SCDFSM14] wird gezeigt, wie durch k-Anonymität das nachträglich hinzuzufügende Rauschen für Differential Privacy reduziert werden kann. Zunächst werden über einen k-member-Clusteralgorithmus Cluster mit mindestens k-Elementen gebildet und generalisiert. Ähnlich wie bei k-means besitzen die Elemente innerhalb eines Clusters einen minimalen Abstand voneinander, während der Abstand zwischen den Clustern maximal ist. Anschließend wird via Differential Privacy Rauschen zu den einzelnen Clustern hinzugefügt, sofern diese nicht bereits ϵ-DP sind. Durch die Kombination beider Techniken bleibt ein erhöhter Datennutzen erhalten.

In [CEK16] wird untersucht, welchen Einfluss `NULL`-Werte ($\perp$) auf mögliche Anonymisierungen durch Generalisierungstechniken haben. Hierfür wird der Begriff *extended match* (erweiterte Gleichheit) eingeführt. Dabei gelten zwei Tupel als gleich, wenn sie an allen Stellen eines Quasi-Identifikators den gleichen Wert oder einen `NULL`-Wert, unabhängig von dessen Semantik[9], aufweisen. Ein *extended match* ist formal wie folgt definiert:

> **Definition:** *extended match* nach [CEK16]
> Zwei Tupel $t_1, t_2 \in r(R)$ sind für eine Relation r über R und einen Quasi-Identifikator Q ein *extended match* ($t_1 \sim_e t_2$), falls $\forall q \in Q : t_1[q] = t_2[q] \vee (t_1[q] = \perp \wedge t_2[q] = \perp)$ gilt.

Für eine weniger strikte Auswertung lässt sich nach [CEK16] der *maybe match* anwenden. Dieser unterscheidet sich vom *extended match* darin, dass nur einer der Werte aus beiden Tupel `NULL` sein muss, damit die Gleichheit gilt:

[9] Wert nicht bekannt, Wert nicht vorhanden oder Wert nicht anwendbar

Definition: *maybe match* nach [CEK16]
Zwei Tupel $t_1, t_2 \in r(R)$ sind für eine Relation r über R und einen Quasi-Identifikator Q ein *maybe match* ($t_1 \sim_m t_2$), falls $\forall q \in Q : t_1[q] = t_2[q] \vee t_1[q] = \perp \vee t_2[q] = \perp$ gilt.

Die zu dem Attribut q gehörende Domänengeneralisierungshierarchie wird zur Umsetzung der Generalisierung in allen Einträgen um den Eintrag $\perp$ erweitert. Somit können `NULL`-Werte mit anderen Werten auf einer beliebigen Anonymisierungsstufe zusammengefasst werden.

In [Agg05] wird die Auswirkung der Dimensionalität bezüglich der Anzahl an Attributen auf die k-Anonymität gezeigt. Mit steigender Attributanzahl wird es zunehmend schwerer, k-Anonymität ohne zusätzliche Anonymisierungen sicherzustellen. Die Autoren nennen für ihre getesteten synthetischen und realen Beispiele den Wert von 45 Attributen als den Punkt, an dem in den meisten Datensätzen für jeden Eintrag nicht mehr die 2-Anonymität für fast jedes Tupel sichergestellt werden kann. Gleichzeitig steigt der erwartete Informationsverlust parallel zur Attributanzahl rapide an, insbesondere, wenn nur wenige Äquivalenzklassen initial vorliegen. Durch die Anwendung von Generalisierungstechniken steigt der Informationsverlust teilweise auf bis zu 60 % an. Dies liegt primär daran, dass viele Attributkombinationen nur selten im Datensatz vorkommen, wodurch eine Generalisierung bzw. Unterdrückung zwingend erforderlich ist.

l-Diversität und t-closeness: K-Anonymität wird in der Fachliteratur, insbesondere von verwandten Anonymisierungsmaßen und -algorithmen, für die vielfältigen Angriffsvektoren zur Deanonymisierung kritisiert. Zu diesen Angriffen zählen beispielsweise die Homogenitätsattacke, bei der die fehlende Vielfalt eines oder mehrerer sensitiver Attribute Rückschlüsse auf die Identität einer Person zulässt. Zudem lassen sich durch den Einsatz von Zusatzwissen, beispielsweise aus externen Datenquellen, weitere Informationen aus der anonymisierten Relation ziehen. Diese und weitere Deanonymisierungsverfahren werden in den weiter unten aufgeführten Publikationen skizziert. Des Weiteren werden in [GGH17, Gol17] weitere Angriffsszenarien besprochen.

Als Verschärfung der k-Anonymität gelten die Maße *l-Diversität* [MKGV07] und *t-closeness* [LLV07]. Durch l-Diversität wird gefordert, dass in jeder durch k-Anonymität entstanden Äquivalenzklasse l verschiedene sensitive Werte auftauchen. Ist jede Äquivalenzklasse l-divers, so erfüllt die gesamte Relation l-Diversität.

Das Maß t-closeness fordert zusätzlich, dass die Verteilung der sensitiven Daten innerhalb einer Äquivalenzklasse der Verteilung der gesamten Relation ähnelt, d. h. *t-close* ist. Der Parameter t spiegelt dabei die minimal erlaubte Abweichung der Verteilung innerhalb der Äquivalenzklassen mit der Gesamtverteilung wider.

In [ZAB14] wird eine Kombination aus Generalisierungstechniken und vertikaler Fragmentation vorgeschlagen. Dazu werden mittels k-Anonymität bzw. l-Diversität Äquivalenzklassen bestimmt. Die Relation wird anschließend derart fragmentiert, dass jedes Fragment nur noch anteilig über die zur Bestimmung der Äquivalenzklassen notwendigen Attribute verfügt. Zudem kann die Verteilung so erfolgen, dass wahlweise keine Verknüpfung der Fragmente mehr möglich ist oder dass zumindest die Äquivalenzklassen wiederherstellbar sind. Durch die Kombination beider Techniken wird ein geringerer Informationsverlust (durch weniger Generalisierungsschritte) erreicht und die Zahl der Angriffsmöglichkeiten, wie beispielsweise *linking attacks*, verringert.

A.8.2 Differential Privacy

Algorithmen für Differential Privacy: Als einfacher, einführender Algorithmus wird in dem einführendem Artikel von Dwork [DR14] ein Münzwurfverfahren für boolesche Werte beschrieben: Für jeden betroffenen Attributwert soll eine Münze geworfen werden. Zeigt sie *Zahl*, so wird der Attributwert ohne Änderung übernommen, bei *Kopf* wird eine zweite Münze geworfen. Zeigt diese *Kopf* an, so wird der Attributwert auf `wahr` gesetzt, bei *Zahl* auf `falsch`.

Als weitere Algorithmen werden in [DR14] der Laplace-Mechanismus und der Exponential-Algorithmus eingeführt. Der Laplace-Mechanismus, welcher ein Rauschen auf Basis der Laplace-Verteilung der Daten hinzufügt, ist für einfache Arten von Anfragen, wie Zählanfragen und Histogramme, geeignet. Bei komplexeren Berechnungen führt dies aber unter Umständen zu erheblichen Abweichungen vom nicht-anonymisierten Ergebnis. Der Exponential-Algorithmus versucht dem entgegenzuwirken, indem neben dem Grad der Anonymisierung auch der

Informationsverlust mit berechnet wird. Dieser wird mit einem sogenannten *privacy budget* abgeglichen, welches nicht überschritten werden darf.

Differential Privacy verhindert verschiedene Re-Identifikationsattacken, wie dem Verknüpfen des anonymisierten Datenbestandes mit Hintergrundwissen aus anderen Datenquellen oder die Aufhebung der Anonymisierung durch inverse Funktionen. Durch diese Attacken wird erreicht, dass das Enthaltensein eines Datensatzes, beispielsweise die personenbezogenen Daten einer einzelnen Person, bekannt wird. Zudem könnten die somit preisgegebenen sensitiven Informationen die jeweilige Person schädigen [DR14].

Durch die Wahl des ϵ und des konkreten Algorithmus der Informationsverlust genauer steuern, wodurch sich ein besserer Kompromiss zwischen Datenschutz und Genauigkeit der Anwendung finden lässt. Zu den komplexeren Differential-Privacy-Algorithmen gehört *Subsample and Aggregate*. Dabei werden die Tupel der Datenbank zufällig und gleichmäßig in m Partitionen $B_1, \ldots, B_m$ unterteilt. Die gegebene Aggregatanfrage Q wird anschließend auf jeder Partition ausgeführt. Die Zwischenergebnisse B_i werden im Anschluss miteinander kombiniert, indem die Aggregatfunktion erneut auf der Menge der B_i ausgeführt wird. Abschließend wird Rauschen durch den Laplace-Algorithmus hinzugefügt.

Umsetzungen von Differential Privacy: Eine Umsetzung von Differential Privacy wird von Kifer und Machanavajjhala mit dem Pufferfish-Framework in [KM14] vorgestellt. Das Framework bietet neben den verschiedenen existierenden Mechanismen zur Sicherstellung von Differential Privacy auch die Möglichkeit, eigene Privacy-Definitionen einzupflegen und zu testen. Der Artikel gibt zudem einen guten Überblick über verschiedene DP-Mechanismen und deren Adaptionen für unterschiedliche Anwendungsfälle.

Ein Anwendungsfall wird von Kessler et al. in [KBB14] untersucht. In dem Artikel wird die Auswirkung von Differential Privacy auf die Auswertung von Smart Metern evaluiert. Einzelne Haushaltsgeräte, wie Fernseher oder Kühlschränke, werden in dem Ansatz als *Kanäle* aufgefasst, welche Informationen über den Energieverbrauch enthalten. Die Summe mehrerer Kanäle wird als *Signal* bezeichnet. Aus den Kanälen werden Schwankungen im Energieverbrauch herausgerechnet, um pro Kanal das Starten und Stoppen des Gerätes zu registrieren. Zu den einzelnen Kanälen wird anschließend Rauschen hinzugefügt, jedoch mit der Nebenbedingung, dass die Auswirkungen auf das Signal nur minimal sind. Das Rauschen ist so zu wählen, dass über einzelne Abrechnungszeiträume die Summe der Abweichungen gleich null ist. Als mögliches Verfahren wird vorgeschlagen, den Laplace-Mechanismus auf aggregierte Verbrauchswerte der Geräte anzuwenden. Zu den möglichen Anwendungsszenarien des Verfahrens zählen laut [KBB14] das Verstecken von Lastspitzen bzw. von Zeiten ohne Verbrauch. Dies kann beispielsweise dafür eingesetzt werden, um Aufsteh- bzw. Schlafenszeiten zu verschleiern.

Differential Privacy lässt sich zudem mit einigen Anpassungen für spezielle Klassen von relationalen Anfragen nutzen. Dies betrifft beispielsweise Bereichsanfragen [YWL+20] sowie Vektoren, Matrizen und Graphen [JLY+21].

Erweiterungen von Differential Privacy: Eine konkrete, wenngleich auch weniger streng ausgelegte Umsetzung von Differential Privacy ist in [HMD13] und [HMD14] zu finden. Das dort vorgestellte Blowfish-Framework nutzt statt dem starren Parameter ϵ einen sogenannten *Policy*-Graphen, welcher in realen Anwendungsszenarien einen höheren Datennutzen gewährleisten soll. Der Graph beschreibt Paare von Werten aus einer Domäne, welche als ununterscheidbar für Anfragen gelten sollen. Angewendet auf zwei Relationen, sind beide benachbart, wenn sie sich in genau einem Attributwert unterscheiden. Ausgehend von dem kürzesten Abstand $d(u, v)$ zweier Relationen $r(R)$ und $r'(R)$, wobei u und v die verbundenen Komponenten der Relation sind, wird die Differential-Privacy-Formel wie folgt modifiziert:

$$P[f(r(R)) \in S] \leq e^{\epsilon * d(u,v)} * P[f(r'(R)) \in S] \tag{A.5}$$

Sind $r(R)$ und $r'(R)$ nicht über gemeinsame Komponenten benachbart, so können die Anfragen auf $r(R)$ bzw. $r'(R)$ getrennt voneinander und ohne das Hinzufügen von Rauschen beantwortet werden. Dies gilt ebenfalls für Anfragen mit Aggregatfunktionen wie *COUNT*.

Ding et al. zeigen in [DWZK19], dass die Wahl des konkreten Differential-Privacy-Verfahrens eine ausgesprochen hohe Auswirkung auf die Genauigkeit der Anfrageergebnisse hat. In ihren Untersuchungen zeigen die Au-

toren, dass insbesondere bei Aggregatanfragen die Genauigkeit um bis zu 50 % gesteigert werden kann und bei gleich bleibenden *privacy budget* mehr Anfragen ausgeführt werden können.

In [HLM12] wird eine Erweiterung des *Exponential Mechanism* um multiplikative Gewichte eingeführt. Ziel ist es, eine bessere Näherung an die Verteilung der Originaldaten zu erreichen, indem für einzelne Tupel bzw. Attributwerte positive und negative Gewichte eingefügt werden. In ihrer Evaluation [HLM12] stellen die Autoren insbesondere eine gute Eignung für Bereichsanfragen auf relationalen Strukturen und Datenwürfeln vor.

Koufogiannis [KHP15] schlägt die Verwendung eines variablen ϵ vor, um das hinzugefügte Rauschen auf einem konstanten Niveau zu halten. Dies verhindert somit unnötigen Informationsverlust durch eine verminderte Genauigkeit für zukünftige Anfragen.

In [ZYWZ14] wird anhand zweier Einsatzszenarien, der Aggregation von Daten aus Sensornetzwerken und der Erhebung von Daten aus Mobilgeräten, die optimale Verteilung des Rauschens untersucht. Der Ausgleich zwischen Privatsphäre und Nützlichkeit wird von den Autoren als Optimierungsproblem formuliert. Ziel der Optimierung ist es, einem gegebenen Datenschutzniveau möglichst nahe zu kommen und dabei ein Maximum an Datennutzen zu erhalten. Für die genannten Anwendungsfälle wird diese Optimierung für Aggregatfunktionen betrachtet. Geng und Viswanath stellen in [GV16] einen vergleichbaren Ansatz für die multidimensionale Anonymisierung mittels Differential Privacy vor.

Differential Privacy und t-closeness liefern unter Ausnutzung von stochastischen Methoden ähnliche Ergebnisse für anonymisierte Daten [DFSC15]. Dabei wird die Distanz der Cluster bei t-closeness betrachtet, welche von dem Parameter ϵ für die Anwendung der Differential Privacy abgebildet wird. Gilt $t = \frac{\epsilon}{2}$, so kann durch t-closeness Differential Privacy sichergestellt werden.

Differential Privacy wird in vielen Anwendungsfeldern benutzt. So kann im Einsatzszenario *Smart Metering* der aggregierte Stromverbrauch unter Ausnutzung von Differential Privacy und probabilistischen Modellen nach [JB14] über mehrere Geräte nahezu akkurat berechnet werden, ohne dass der genaue Verbrauch eines einzelnen Gerätes preisgegeben wird. Gleiches gilt für die Vorhersage des Verbrauches. Für den Anwendungsfall von smarten Umgebungen existieren Protokolle [WMYR16, WMYR14], die zusichern, dass Daten mit minimalem Fehler aggregiert werden. In [CCFS21] wird gezeigt, wie sich Differential Privacy für die Anonymisierung von Bewegungsmustern nutzen lässt.

A.9 Datenschutz – Konkrete Verfahren in der Praxis

Im Folgenden wird eine kurze Übersicht zum Einsatz von Anonymisierungsverfahren in der Praxis gegeben. Der Fokus liegt dabei auf der Anwendung in smarten Systemen bzw. smarten Umgebungen und den darin verwendeten Analysemodellen.

Smart Metering mit Aggregationen und SMC: In [BCB+13] wird untersucht, wie eine datenschutzgerechte Auswertung von Daten im Smart Metering umgesetzt werden kann, insbesondere um ungewollte Profilbildungen zu vermeiden. Dabei werden Daten nicht in Echtzeit vom Verbraucher an den Anbieter weitergeleitet, sondern in zeitlich voraggregierter Form (sogenannte *Shards*) übertragen, um so keinen Rückschluss auf die Quellen des Energieverbrauchs zu schließen. Weiterhin können die Shards mehrerer Nutzer aggregiert werden, um weitere statistische Analysen, z. B. um den Energieverbrauch pro Region zu ermitteln, auszuführen.

Zur Umsetzung setzen die Autoren einerseits auf Clustermethoden, die auf k-Means und Fuzzifizierung setzen, um so gleichartige Nutzer zu Gruppen zusammenzufassen. Zudem wird *Secure-Multi-Party-Computation* (SMC) eingesetzt, um die Daten verschiedener Quellen zusammenzuführen. Dabei wird jedoch auf homomorphe Verschlüsselung aus Performanzgründen verzichtet. Stattdessen wird linear *threshold secret sharing* genutzt, wobei die Daten auf mehrere Shards und Anbieter verteilt werden.

Kollaboratives Filtern mit Voraggregationen: In [YP12] wird ein Verfahren zum hybriden[10] kollaborativem Filtern auf horizontal und vertikal verteilten Daten vorgestellt. Dabei werden bei jedem Beteiligten lokal die Be-

[10] Im Sinne von speicher- und modellbasierten Ansätzen

rechnungen zur Ermittlung von Mittelwerten, zur Normalisierung und zur Ermittlung des Kosinuswinkels zur Bestimmung ähnlicher Objekte, z. B. Waren, ermittelt. Zwischen den Beteiligten werden auf globaler Ebene nur die Daten über Subjekte, z. B. Kunden, ausgetauscht, um deren Ähnlichkeit zu bestimmen. Das Wissen um die Beziehung zwischen Objekt und Subjekt wird somit nicht zwischen den Beteiligten ausgetauscht. Auf Secure-Multi-Party-Computation wird in diesem vorgeschlagenen Verfahren zwecks Vermeidung hoher Berechnungskosten verzichtet, es werden stattdessen Voraggregationen genutzt, um beispielsweise Mittelwerte auszutauschen. So wird statt des Mittelwertes über die Eigenschaften der Nutzer ein Paar aus Summe und Anzahl ausgetauscht, wodurch der Mittelwert auch global, ohne Übertragung aller Daten, berechnet werden kann.

Vermeidung diskriminierender Auswertungen: Neben der Wahrung des Datenschutzes werden Anonymisierungstechniken auch zur Vermeidung von — beabsichtigten oder unbeabsichtigten — diskriminierenden Auswertungen und Analysen verwendet [HDF14]. Diskriminierungen hinsichtlich des Geschlechtes, der Rasse oder der Religionszugehörigkeit können beim Data Mining unbeabsichtigt vorkommen, beispielsweise wenn bereits existierende Diskriminierungen in den Daten vorliegen und durch Assoziationsregeln oder Klassifikationen dieser Trend noch weiter verstärkt wird. Durch die Anwendung von auf Generalisierung basierten Anonymisierungsverfahren können die betreffenden Attribute von der Analyse ausgeblendet werden [HDF14].

Datenvermeidung in Sensorsystemen: In [HS17] wird das Prinzip der Datenvermeidung und Datensparsamkeit am Beispiel eines Sensorsystems verdeutlicht. Die Ausgestaltung eines automatisierten Sensornetzwerkes für die Positionsbestimmung einer Person im Raum soll derart erfolgen, dass nicht jede Umgebungsposition erhoben, verarbeitet und gespeichert wird. Beim Entwurf des Systems soll vielmehr darauf geachtet werden, dass nur die notwendigen Informationen erhoben werden und dass eine unverzügliche Filterung der unwichtigen Daten erfolgt. Als konkrete Maßnahmen schlagen die Autoren eine Reduzierung des Sensorerfassungsbereiches auf den minimal benötigten Radius vor. Innerhalb dieses Umkreises soll zudem entschieden werden, welche Daten von Belangen sind. Als Beispiel führen die Autoren die Erfassung der Anzahl von Personen statt der konkreten Nennung der einzelnen Personen auf. Sollen trotzdem personenbezogene Daten erhoben werden, so sollen diese unverzüglich anonymisiert oder gelöscht werden, sobald die Daten zur Erfüllung des Verwendungszweckes nicht mehr benötigt werden.

Weitere Verfahren: In [HDM+15] wird gezeigt, wie die k-Anonymität bzw. Differential Privacy direkt in die Berechnung von häufig vorkommenden Pattern integrieren lassen. Dabei werden die Ansätze des *Privacy-preserving Data Minings* (PPDM) mit den Ansätzen des *Discrimination Prevention in Data Minings* (DPDM) verbunden. Danezis et al. schlagen in [DFKB13] ein weiteres, auf SMC basierendes Verfahren vor. In der Arbeit liegt ein stärkerer Fokus auf Datensparsamkeit durch eine Verteilung der Daten auf die einzelnen Knoten, die anschließend durch ein SMC-Protokoll keine vertraulichen Daten an andere Knoten übertragen.

A.10 Datenschutz – Systeme

Dieser Abschnitt stellt einige konkrete Systeme zur Wahrung des Datenschutzes dar. Im Gegensatz zu den Techniken im vorherigen Abschnitt A.9 wurden die Systeme nicht um Datenschutztechniken ergänzt, sondern wurden von Grund auf mit diesen verzahnt.

A.10.1 Aircloak

Aircloak [MA14, Mei14] ist ein durch TÜViT[11] und der französischen Datenschutzbehörde CNIL[12] zertifiziertes System, welches aus Forschungsarbeiten am Max-Planck-Institut für Softwaresysteme hervorgegangen ist. Anwendungszweck des Systems ist es, vorrangig den Datenschutz der Nutzer zu wahren, zeitgleich aber eine

[11] `https://www.tuvit.de`, zuletzt aufgerufen am 05.01.2022
[12] `https://www.cnil.fr/en/home`, zuletzt aufgerufen am 05.01.2022

zu hohe Anonymisierung verhindern, wenn aufgrund einer zu großen Nutzerbasis eine hohe Anzahl an Quasi-Identifikatoren entstehen. Als Anwendungsgebiete werden insbesondere das Smart Metering und der Informationsaustausch von Gesundheitsdaten hervorgehoben. Für beide Bereiche werden somit datenschutzkonforme OLAP-Anfragen und Analysen auf den Servern von Aircloak ermöglicht.

In Aircloak werden nicht die Daten selbst anonymisiert, sondern lediglich die Antworten, die als Ergebnis zu einer Anfrage zurückgeliefert werden. Die Anonymisierung selbst erfolgt durch bekannte Techniken, die auf k-Anonymität, Differential Privacy und die Unterdrückung von Tupeln basieren.

Die Nutzerdaten innerhalb von Aircloak werden mittels kryptografischer Hardware verschlüsselt, wodurch nicht einmal Systemadministratoren Zugang auf die Rohdaten erhalten können. Zudem wird durch das System sichergestellt, dass nur durch passende Software oder Schnittstellen ein Zugriff auf die Daten erfolgen kann. Zu diesen Schnittstellen gehören auch Plugins zu Datenbankmanagementsystemen wie PostgreSQL und zu Programmen wie Tableau[13]. Um das Vertrauen in das System weiter zu erhöhen, wurden Teile des Quellcodes im Internet[14] zur freien Verfügung bereitgestellt. Zudem existiert ein Programm, um Fehler gegen Zahlung eines Preisgeldes zu melden[15].

A.10.2 Vorschlag vom ADAC

Ein zu Aircloak ähnliches Verfahren schlägt der Allgemeine Deutscher Automobil-Club (ADAC) für die Speicherung und Analyse von Fahrzeugdaten in [Eic18] vor. Über einen neutralen Administrator als Treuhänder sollen die vom Automobil erzeugten Daten zunächst gefiltert und aggregiert werden, bevor Dritte, wie Automobilhersteller oder die eigene Versicherungsgesellschaft, auf die vorverarbeiteten Daten zugreifen können. Zu den gespeicherten Daten gehören die Sitzposition, Navigationsinformationen und gespeicherte Daten von Assistenzsystemen sowie Informationen zum Zustand des Motors oder der Systemsoftware. Diese Daten sollen zudem über Datenschutzeinstellungen, die der Nutzer zuvor festgelegt hat, vor unbefugter Auswertung geschützt werden.

A.10.3 ARX

ARX [PKLK14] ist ein grafisches Tool zur Anonymisierung von relationalen Daten. Nutzer können als Eingabe CSV- und Excel-Dateien sowie Daten aus relationalen Datenbanken über eine JDBC-Schnittstelle in das Programm importieren. Der Datensatz wird anschließend als Stichprobe in Tabellenform dargestellt (siehe Abbildung A.2). Der Nutzer kann anschließend einzelne Attribute auswählen und die für den Anonymisierungsprozess notwendigen Parameter angeben:

- **Typ:** Attribute können als Schlüsselattribute, Quasi-Identifikatoren oder als sensitiv bzw. insensitiv gekennzeichnet werden.
- **Transformation:** Sollen die zum Attribut gehörigen Attributwerte anonymisiert werden, so kann zwischen Generalisierung, Aggregation und Clustering gewählt werden.

Wird Generalisierung als Transformationsmethode gewählt, so kann der Nutzer zusätzlich entscheiden, wie die Domänengeneralisierungshierarchie für dieses Attribut aufgebaut ist. Außerdem kann er festlegen, bis zu welchem Grad die Daten unterdrückt statt generalisiert werden. Als weitere Einstellungen kann der Nutzer festlegen, welches Anonymisierungsmaß (in Abbildung A.2: k-Anonymität mit `k = 5`) und welche Informationsverlustmetrik, u. a. die Kullback-Leibler-Divergenz [KL51], verwendet werden soll. Außerdem können den Attributen Gewichte zugeordnet werden, um einzelne Attribute bevorzugt für die Anonymisierung auszuwählen.

A.10.4 Java PrivMon

In [Sch13] wird das *Privacy Enforcement Framework* vorgestellt, welches das Java Security Framework um eine auf XACML (eXtensible Access Control Markup Language) basierte Datenschutzrichtlinie erweitert. Die

[13] `https://aircloak.com/features.html`, zuletzt aufgerufen am 05.01.2022
[14] `https://github.com/Aircloak/`, zuletzt aufgerufen am 05.01.2022
[15] `https://aircloak.com/compliance/attack-challenge/`, zuletzt aufgerufen am 05.01.2022

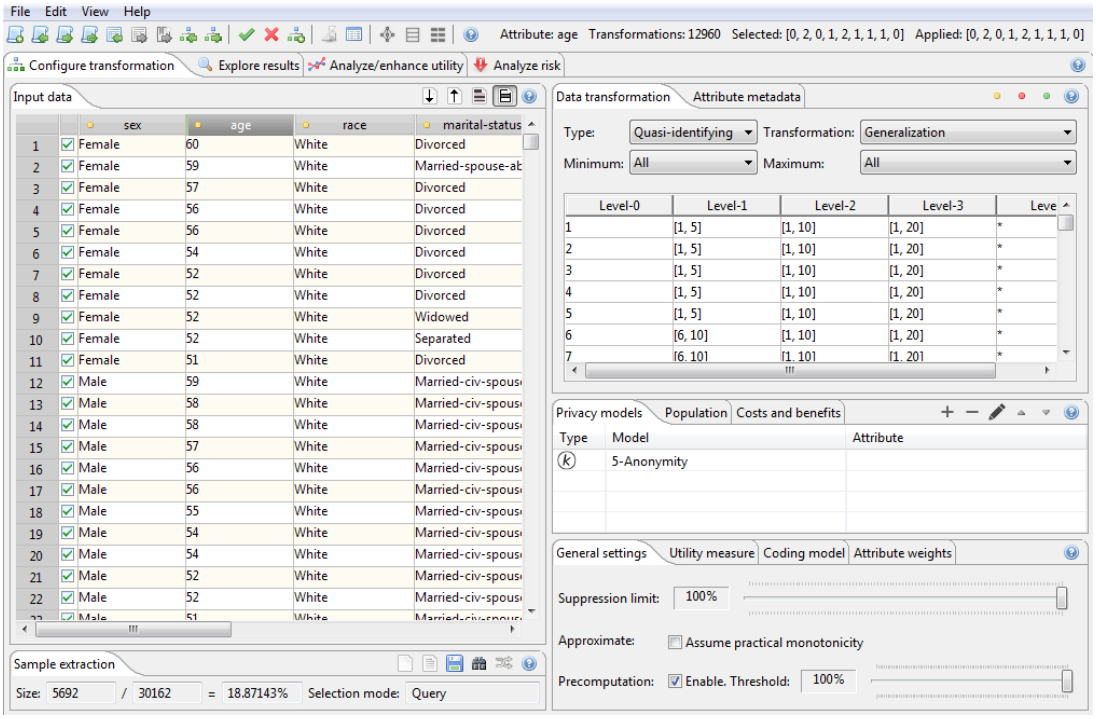


Abbildung A.2: Grafische Oberfläche von ARX [PKLK14]: Im linken Ausschnitt des Fensters ist die Eingaberelation zu sehen, auf der rechten Seite sind die Parameter für die Anonymisierung einstellbar.

---

im jeweiligen Anwendungsprogramm hinterlegte XACML-Datei wird dazu verwendet, um über die Java Virtual Machine Zugriffe auf das Dateisystems des Nutzers zu gewähren bzw. zu verhindern. Das Framework verhindert somit unbefugte Zugriffe auf personenbezogene Daten.

A.10.5 Datenschutztechniken für spezielle Systeme

Neben den oben genannten Systemen für die allgemeine datenschutzfreundliche Verarbeitung von personenbezogenen Daten existieren eine Vielzahl von spezialisierten Systemen für konkrete Anwendungsszenarien. Diese Techniken sind jedoch meist auf spezifische Soft- bzw. Hardwareprodukte beschränkt und lassen sich meist nicht auf andere Anwendungsfälle übertragen.

Smart TV: In [GT14] wird ein Opt-In-Verfahren zur Datenvermeidung bei HbbTV-Fernsehern (Hybrid Broadcast Broadband TV; besser bekannt als *Red Button*) vorgestellt. Das vorgestellte zweistufige Verfahren sieht vor, dass der Nutzer zunächst der Datenübertragung zustimmen muss (*Green Button*), bevor er Inhalte aus dem jeweiligen Sendernetzwerk vorgeschlagen bekommt. Ein ähnliches Verfahren schlägt der Heise-Verlag mit dem Shariff-System[16] für die Übertragung von Inhalten aus bzw. an soziale Netzwerke beim Besuch von Webseiten vor.

Smart Metering: Für das Anwendungsfeld des *Smart Meterings* wird in [REC15] ein Verfahren zur lokalen Vorverarbeitung der gesammelten Stromverbrauchsdaten eingeführt. Es werden sechs Klassen an Vorverarbeitungsschritten vorgestellt, welche die folgenden Bedingungen erfüllen:

[16] `https://www.heise.de/newsticker/meldung/Datenschutz-und-Social-Media-Der-c-t-Shariff-ist-im-Einsatz-2470103.html`, zuletzt aufgerufen am 05.01.2022

1. Die Algorithmen müssen auf eingebetteten Geräten, wie dem Smart Meter oder der dazugehörigen Infrastruktur, lauffähig sein.
2. Aus den vorverarbeiteten Daten lässt sich weiterhin der reale Stromverbrauch ablesen, d. h., die Abweichung von den Originaldaten sollte möglichst gering sein.
3. Durch die Vorverarbeitung kommt es bei den Auswertungen nur zu einer minimalen Verzögerung.

Zu den vorgestellten Verfahren zählen die Verringerung der Messfrequenz, die Durchschnittsbildung über einen bestimmten Zeitraum und die Verknüpfung beider Verfahren. Zudem eignet es sich für Vorberechnungen, dem Hinzufügen von Rauschen sowie dem Runden von Messwerten auf das Vielfache eines bestimmten Messwertes bzw. auf den Mittelpunkt eines vorher berechneten Clusters.

A.10.6 Datenschutz im Internet der Dinge

Die in den letzten Abschnitten gezeigten Methoden und Systeme zeigen, wie bestehende Informationssysteme um datenschutzgerechte Konzepte erweitert werden können. Nicht nur durch *Big Data* allein hat sich die IT-Landschaft in den letzten Jahren gewandelt. Durch das verstärkte Aufkommen des Internets der Dinge erfolgt eine verstärkte Datensammlung – mit teils im Vorfeld unbekannten Auswertungszwecken. Voigt und van dem Bussche erörtern dieses Dilemma in ihrem Praktikerhandbuch:

> **Zitat:** Datenschutzdilemma im Internet der Dinge [VvdB18, Kapitel 9.3]
>
> [Im IoT] wird auch ein gewisses Dilemma sichtbar: denn während vorbenannte Konzepte den Verantwortlichen aufgeben die Produkte so zu entwickeln, dass diese möglichst datensparsam eingesetzt werden, bezwecken IoT-Anwendungen diametral das Gegenteil, werden sie doch extra so designt, dass sie möglichst umfänglich Daten erfassen. [...] Diese technologische Entwicklung kann und sollte nicht durch unbalancierte Durchsetzung der vorbenannten Schutzkonzepte schlicht verboten werden. Vielmehr sind Ansätze zu verfolgen und zu entwickeln, wie die technologisch bedingte Zunahme von Datenerhebungen, insbesondere bei IoT, mit dem Gebot der Datensparsamkeit in Einklang zu bringen ist.

Kern meiner Promotion ist es, genau diese Lücke zu füllen. Dazu wird in Kapitel 6 ein neues Verfahren vorgeschlagen, welches das Gebot der Datensparsamkeit technologisch umsetzt, dabei aber keinen Informationsverlust hinsichtlich der Qualität der vom System vorgegebenen Analysen erzeugt. Die Arbeit ordnet sich in das Langzeitrahmenprojekt *PArADISE* ein, welches als Abschluss dieses Grundlagenkapitels im folgenden Unterabschnitt vorgestellt wird.

Anhang B

Anhang zu Kapitel 3

In diesem Anhang wird auf die weiterführenden Konzepte aus dem Kapitel 3 eingegangen.

B.1 'Privacy Policy for Smart Environments'

In diesem Anhang wird näher auf die in Abschnitt 3.2 eingeführte *Privacy Policy for Smart Environments* (PP4SE) eingegangen. Es wird näher auf den Aufbau und das dazugehörige XML-Schema eingegangen. Zuvor gehen wir kurz auf die verwandten Auszeichnungssprachen ein.

B.1.1 Anforderungen an Richtlinien

Um den hohen Anforderungen der Datenschutzgesetze zu genügen, müssen in Datenschutzrichtlinien mit hoher Präzision, aber auch mit ausreichender Deutlichkeit und Transparenz, angegeben werden,

- welche Daten unter den Datenschutz fallen,
- wie diese für welche Zwecke von welcher Stelle ausgewertet werden und
- wie die Daten ggf. datenschutzgerecht aufbereitet werden müssen.

Die Datenschutzbestimmungen dürfen dabei nicht ausufern, da bekannt ist, dass zu viele – und vor allem zu lange – Bestimmungen wenig wahrgenommen werden. Dabei ist bereits durch frühe Studien [MC04] bekannt, dass lediglich 4,5% der Nutzer von Online-Diensten regelmäßig die Datenschutzrichtlinien lesen – ca. zwei Drittel tun dies selten oder nie. Nach Einführung der europäischen Datenschutz-Grundverordnung sind, nach der in [LHF18] durchgeführten Studie, die auf Webseiten hinterlegten Datenschutzrichtlinien im Schnitt um 23% länger geworden, wobei ca. 30% der Webseiten ihre Richtlinien noch nicht angepasst haben. Um eine bessere Lesbarkeit und ein leichteres Verständnis seitens der Nutzer zu erreichen, sollten Datenschutzrichtlinien einerseits ausreichend spezifisch sein, den Nutzer andererseits auch nicht überfordern.

Datenschutzrichtlinien formulieren in ihren Grundzügen die Ansprüche eines digitalen Rechtemanagementsystems für personenbezogene Daten. Die formulierten Rechte müssen anschließend durch eine vertrauenswürdige Softwarekomponente – hier: der entwickelte, mehrstufige Anfrageprozessor – durchgesetzt werden. Die Datenschutzrichtlinie dient zur Parametrisierung der Datenschutzalgorithmen, indem sie Anforderungen an die Anonymitätsmaße stellt und die Datenverarbeitung im System steuert und passend für den jeweiligen Einsatzzweck respektive Nutzer beschränkt.

Daten dürfen aus der Datenbank nur freigegeben werden, wenn die entsprechenden Freigaben in der Datenschutzrichtlinie existieren. Gemäß der Zweckbindung muss ein System bzw. eine Anwendung Gründe für den

Zugriff auf die einzelnen, erforderlichen Daten angeben. Neben der ständigen Freigabe sollten auch zeitlich beschränkte Freigaben angegeben werden, die den Zugriff nur für einen bestimmten Zeitraum oder in vordefinierten Intervallen gestatten.

Zur formalen Definition dieser Ansprüche existieren verschiedene Auszeichnungssprachen, welche im Anhang B.1.2 kurz präsentiert werden. Auf Basis dieser Sprachen wurde im Rahmen von PArADISE ein eigener Sprachentwurf, die *Privacy Policy for Smart Environments*, entwickelt, welche den Anforderungen von Assistenzsystemen gerecht wird.

B.1.2 Datenschutzauszeichnungssprachen

Das World Wide Web Consortium (W3C) stellt zwei Vorschläge für die Formulierung von Datenschutzrichtlinien vor: das *Platform for Privacy Preferences Project* (P3P, [W3C07]) und die *Enterprise Privacy Authorization Language* (EPAL, [W3C03]). Während P3P auf die Kommunikation zwischen Nutzer und Webseitenbetreiber abzielt, regelt EPAL darüber hinaus die innerbetrieblichen Regelungen zur Datennutzung. EPAL versteht sich nicht nur als reine Erweiterung zum P3P, sondern sieht sich als komplementär in Hinblick auf die Einsatzzwecke an. In EPAL ist es möglich, die Anwendungszwecke, die Nutzer der Daten und die Datenkategorien hierarchisch anzuordnen.

Beide regulieren über regelbasierte Zugriffsbeschreibungen, die in XML formuliert sind, den Zugriff auf Teile von Datenbeständen. In einzelnen Regeln werden Rollen und Bedingungen definiert, unter denen Schreib- und Leserechte vergeben bzw. abgewiesen werden können. Am Beispiel des P3P wird im Folgenden der Aufbau einer Datenschutzrichtlinie erklärt.

P3P: Die Platform for Privacy Preferences (P3P) ist ein Format zum Austausch von Datenschutzinformationen, welches primär von webbasierten Anwendungen genutzt wird. Der Betreiber definiert auf seiner Server-Seite eine XML-Datei, welche seine selbstbestimmten Datenschutzrichtlinien beschreibt. Diese Richtlinien legen den Verwendungszweck und die Dauer der Speicherung von den erhobenen, personenbezogenen Daten fest. Dabei spezifiziert der Betreiber zudem, welche Arten von Daten vom Nutzer erhoben und – beispielsweise in Form von Cookies – gespeichert werden.

Gleiches legt der Nutzer auf der Client-Seite fest. Dabei spezifiziert er, welche seiner personenbezogenen Daten für welche Zwecke verwendet werden dürfen. Erfolgt ein Zugriff auf eine P3P-unterstützende Website, werden beide Profile miteinander abgeglichen. Liegt dabei ein Widerspruch vor, wird der Nutzer darüber informiert. Er entscheidet anschließend, ob er die Website trotzdem nutzen möchte. Praktische Anwendung findet P3P kaum noch – Microsoft stellte als letzter Webbrowser-Anbieter die Unterstützung für den Internet Explorer und dessen Nachfolger Edge Ende 2016 ein[1].

Weitere Auszeichnungssprachen: Neben diesen Standards existieren noch viele weitere, ähnliche Vorschläge, wie die *eXtensible Access Control Markup Language* (XACML, siehe [OAS13]) und die *Security Assertion Markup Language* (SAML, siehe [CKPM05]), die sich im Wesentlichen aber nicht von der P3P-Auszeichnungssprache unterscheiden bzw. in diese eingearbeitet wurden. XACML ist gegenwärtig der am besten gepflegten Standard und verfügt über mehre Implementierungen für verschiedene Programmiersprachen[2]. Mit der *XPACML* [BBL10] wurde beispielsweise ein Sprachvorschlag geschaffen, welcher XACML und P3P vereint. Ein guter Überblick zu den bereits erwähnten und weiteren Sprachen zur Formulierung von Datenschutzrichtlinien ist in [Sch13] zu finden.

In der Praxis gängige Systeme, wie das Betriebssystem *Android* von Google bzw. Alphabet, setzen die Auszeichnungssprachen nicht vollständig um. Stattdessen verwenden sie eigens entwickelte Strukturen, um die Rechtevergabe bezüglich der personenbezogenen Daten zu regeln. Im Falle von Android existiert zwar eine feingranulare

[1] `https://msdn.microsoft.com/en-us/ie/mt146424`, zuletzt aufgerufen am 05.01.2022.

[2] siehe `https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml#tc-tools`, zuletzt aufgerufen am 05.01.2022.

Aufteilung der Aktionen, die eine mobiles Endgerät ausführen kann, jedoch wird nur sehr eingeschränkt auf den Verwendungszweck sowie dem Umfang der Erhebung informiert[3].

B.1.3 Privacy Policy for Smart Environments (PP4SE)

In PArADISE wird eine eigene Auszeichnungssprache zur Formulierung von Privatheitsansprüchen, die Privacy Policy for Smart Environments (PP4SE), verwendet. Ausgehend von den Anforderungen des Bundesdatenschutzgesetzes [Deu15] müssen Informationssysteme folgende Informationen bereitstellen, sofern sie personenbezogene Daten speichern oder verarbeiten:

1. Die Dauer, der Umfang, die Art und der Zweck der Verarbeitung bzw. Erhebung,
2. die Kategorie der erfassten Daten,
3. die beteiligte Akteure bei der Verarbeitung und Nutzung der Daten, sowie
4. die Herkunft der Daten.

PArADISE bietet den Entwicklern und Nutzern von Assistenzsystemen vielfältige Möglichkeiten zur datenschutzkonformen Verarbeitung von personenbezogenen Daten. Neben der Option, kritische Attributkombinationen automatisch zu erkennen (Details dazu in Kapitel 4), existiert für beide Seiten, Entwicklern und Nutzern, die Möglichkeit zur Formulierung und Hinterlegung eines Datenschutzprofils in Form der PP4SE.

Durch feingranulare Datenschutzprofile kann der Zugriff auf personenbezogene Daten genauer und zweckgebundener gestaltet werden. Der Nutzer wird durch sie in die Lage versetzt, anzugeben, welche Informationen in bestimmten Kontexten genutzt und verarbeitet werden dürfen. Durch die Bindung von Daten an ausgewählte Analysemethoden können Daten gezielt verdichtet werden, wodurch ungewollte Analysen zwar nicht immer verhindert, aber zumindest reduziert werden können.

PP4SE basiert in seinen Grundzügen auf dem Sprachvorschlag P3P vom W3C [W3C07], verzichtet aber auf technische bzw. plattformspezifische Details, wie beispielsweise die Verwaltung von Cookies durch Browser. Es erfolgt vielmehr eine Erweiterung um eine datenzentrierte Sichtweise: Dem Nutzer wird die Möglichkeit angeboten, Anfrageintervalle festzulegen, in denen die Daten auf seinen Endgeräten abgefragt werden dürfen. Zudem kann er explizit festlegen, welche Operatoren (z. B. Aggregationen und Skalarfunktionen) auf seinen Daten angewendet werden dürfen.

B.1.4 Interne Darstellung

Bevor die formale Definition der PP4SE-Richtlinie vorgestellt wird, illustriert ein kleines Beispiel (siehe Quellcodeausschnitt B.1) deren Aufbau. Als interne Repräsentation und zum Austausch zwischen den beteiligten Akteuren wird, wie bereits bei den zuvor vorgestellten Richtlinien, XML als Austauschformat verwendet.

```
<policy>
  <apps>
    <application id="Amarok">
      <appDescription>
        Das ist meine private Amarok-Musikdatenbank.
      </appDescription>
      <modules>
        <module id="playMusic" type="basic" required="yes">
          <moduleDescription>
            Zugriff auf die gekauften Musiktitel zum Abspielen.
          </moduleDescription>
```

[3] siehe u. a. https://android-developers.googleblog.com/2019/03/giving-users-more-control-over-their.html, zuletzt aufgerufen am 05.01.2022.

```
12            <attribute-list>
13              <attribute name="title">
14                <allow>true</allow>
15                <third-party-access>false</third-party-access>
16                <condition>length &gt;= 10000</condition>
17              </attribute>
18            </attribute-list>
19          </module>
20          ...
21        <modules>
22      </application>
23    <apps>
24  </policy>
```

Quellcodeausschnitt B.1: Beispiel für eine Datenschutzrichtlinie im PP4SE-Schema

---

In dem Quellcodeausschnitt B.1 zum laufenden Beispiel verfügt die Anwendung *Amarok* über eine Funktion *playMusic*, welche die gekauften Musikstücke abspielt und für die grundlegende Funktionalität (`type="basic"`) des Programmes zwingend erforderlich ist (`required="yes"`). Dabei greift das Programm auf den Titel des Musikstückes zu. Die Verwendung dieses Attributes wird dem Programm erlaubt (`allow`), jedoch nicht für Dritte (`third-party-access`; z. B. Werbefirmen). Als weitere Einschränkung (`condition`) darf nur auf Stücke zugegriffen werden, deren Laufzeit unter 10 Sekunden liegt, um beispielsweise den Zugriff auf private Aufnahmen zu verhindern.

Durch die Unterteilung der Anwendungen in deren Funktionalitäten wird dem Aspekt der Zweckbindung genüge getan. Der Nutzer kann einer Anwendung verschiedene Rechte einräumen, wodurch ein feingranular beschränkter Zugriff auf die Daten erfolgt. Gleichzeitig kann er für gleiche Funktionalitäten verschiedener Anwendungen unterschiedliche Zugriffsregelungen formulieren. Ein feingranularerer Informationszugriff, der zusätzliche Abstufungen zwischen Sperren und Freigeben einfügt, kann sowohl den Informationsbedarf des Assistenzsystems als auch die Privatheitsansprüche der Nutzer zufriedenstellen. Mögliche Zwischenstufen können durch Selektion, Projektion und Aggregation, aber auch durch das Einfügen von Fuzzy-Werten und der Anwendung von Anonymisierungsverfahren, erreicht werden.

PP4SE wurde auf Basis von XML Schema erstellt. Details zum Aufbau der Richtlinie sind in Anhang B.1 zu finden. Im Folgenden wird die für PP4SE zugrunde liegende formale Semantik vorgestellt.

Formale Semantik

Formal wird PP4SE als 9-Tupel der Form
`P := (APP, M, ATTR, C, I, A, AGG, ANO, f)`
definiert, wobei folgende Elemente verwendet werden:

- **APP**: Menge von Anwendungen, für die Datenschutzprofile angelegt wurden,
- **M**: Menge der bereitgestellten Funktionalitäten,
- **ATTR**: Menge von Attributen, auf die zugegriffen wird,
- **C**: Menge von definierten Bedingungen,
- **I**: Menge von Anfrageintervallen,
- **A**: Menge von Zugriffsberechtigungen,
- **AGG**: Menge von Aggregatfunktionen,

- **ANO**: Menge von Anonymisierungsfunktionen und
- **f**: Eine partiell definierte Funktion mit $f := (app, m, attr) \rightarrow (C', i, A', agg, ano)$.

Die Funktion f bildet eine Anwendung $app \in APP$, eine Funktionalität $m \in M$ und ein Attribut $attr \in ATTR$ auf eine Teilmenge der Bedingungen $C' \subseteq C$, ein Anfrageintervall $i \in I$, eine Teilmenge der Zugriffsberechtigungen $A' \subseteq A$, eine Teilmenge der Aggregatfunktionen $AGG' \subseteq AGG$ sowie eine Anonymisierungsfunktion $ano \in ANO$ ab. Durch f werden die Zugriffsbedingungen der Funktionen eines Programmes auf die einzelnen Attribute spezifiziert.

Zusätzlich zum obigen Beispiel können Anfrageintervalle definiert werden, welche die Häufigkeit des Zugriffs auf einzelne Attribute beschränken. Die Angabe von Anonymisierungs- und Aggregatfunktionen erlaubt es dem Nutzer, optional festzulegen, wie seine Daten vorverarbeitet bzw. anonymisiert werden sollen. Dies dient primär der Initialisierung der Anonymisierungsverfahren im Postprozessor (siehe Abschnitt 3.4). Darüber hinaus kann im Präprozessor durch die Angabe der Datenschutzeinstellungen die Anfrage im Idealfall bereits so transformiert werden, dass kein Zugriff auf die betroffenen Attribute erfolgt (siehe Abschnitt 3.3). Dazu werden die Zugriffsbedingungen in Integritätsbedingungen transformiert, welche in die Anfrage integriert werden.

Adaption von Datenschutzrichtlinien für neue Datenquellen

Assistenz- bzw. Informationssysteme sind in ubiquitären Umgebungen meist sehr dynamisch ausgelegt: Neue Sensoren können ad-hoc in das System eingebunden werden, um bestehende Sensorsysteme zu ersetzen oder zu ergänzen; ebenso werden die Algorithmen für die Situations- und Intentionserkennung ausgetauscht. Für das veränderte Umfeld ist es zwingend erforderlich, die Datenschutzprofile anzupassen, um den Datenschutzanforderungen gerecht zu werden – im besten Fall ohne Eingriffe seitens des Nutzers. Daher verfügt der Anfrageprozessor in PArADISE zusätzlich über die Fähigkeit, Datenschutzprofile für neue Anwendungen bzw. Systeme auf Basis von bereits angelegten Profilen des Nutzers zu generieren.

Das in PArADISE entwickelte dreistufige Verfahren [GH15b] nutzt verschiedene aus den Bereichen Schemaintegration und Data Mining bekannte Techniken. Im ersten Schritt werden Abbildungen mittels Similarity Flooding [MGMR02] zwischen den Relationenschemata der bisher verwendeten und den neu hinzugekommenen Sensoren erzeugt. Anschließend werden die Einstellungen desjenigen Sensors mit der höchsten Ähnlichkeit auf den neuen Sensor übertragen. Dabei werden nur die Einstellungen transferiert, für welche abgebildete Attribute existieren. Für alle weiteren Attribute werden Data-Mining-Verfahren (im Speziellen: Clustering und Assoziationsanalyse; siehe [Mei15]) genutzt, um ähnliche bzw. gemeinsam verwendete Attribute und deren hinterlegte Datenschutzprofile zu bestimmen. Werden solche Attributkombinationen entdeckt, wird deren Datenschutzprofil auf die nichtabgebildeten Attribute übertragen. Abschließend wird für die restlichen Attribute festgelegt, dass kein Zugriff auf diese erfolgen darf.

Das auf diese Weise erzeugte Datenschutzprofil für die neue Anwendung bzw. das neue Gerät stellt nur eine Empfehlung dar – es liegt in der Hand des Nutzers sie zu akzeptieren bzw. nach seinen Wünschen anzupassen. Im folgenden Beispiel wird der Ablauf des Verfahrens kurz skizziert:

Beispiel: Automatisierte Adaption von Datenschutzprofilen

Zur Illustration des Adaptionsalgorithmus gehen wir von einer Erweiterung der `User`-Relation des laufenden Beispiels aus. Zusätzlich zu den bereits erfassten Daten werden auf den verschiedenen Endgeräten des Nutzers Geokoordinaten und der aktuelle Zeitstempel erfasst. Zum leichteren Verständnis gehen wir von einer vereinfachten Datenschutzeinstellung in Form eines einzelnen binären Attributes mit den Werten `PRIVATE` (Daten dürfen nicht herausgegebenen werden) und `PUBLIC` aus. Auf detaillierte Informationen, beispielsweise die Datentypen der Attribute, wird ebenfalls verzichtet.

Endgerät `E1` speichert die Position als Triplet der Form *(*x, y, z), die aktuelle Zeit im Attribut *timestamp* sowie den Vor- und Nachnamen. Für `E1` hat der Nutzer festgelegt, dass sein Nachname nicht veröffentlicht werden darf. Die folgende Tabelle zeigt die Einstellungen für jedes Attribut in `E1`:

Attribut	Einstellung
lastname	PRIVATE
firstname	PUBLIC
x	PUBLIC
y	PUBLIC
z	PUBLIC
timestamp	PUBLIC

Endgerät `E2` speichert zusätzlich die eindeutige ID des Nutzers (*uid*) und seine Bewegungsrichtung (*direction*), dafür wird kein Zeitstempel erfasst. Auf diesem Endgerät wurde die Datenschutzeinstellung so gewählt, dass Vor- und Nachname nicht herausgegeben werden dürfen:

Attribut	Einstellung
uid	PUBLIC
lastname	PRIVATE
firstname	PRIVATE
x	PUBLIC
y	PUBLIC
z	PUBLIC
direction	PUBLIC

Wird die bestehende Systemumgebung um ein neues Gerät `E3` erweitert, beispielsweise durch den Kauf eines neuen Abspielgerätes für Musik, so existiert zunächst kein Datenschutzprofil (Spalte *initial* in der nachfolgenden Tabelle; gekennzeichnet durch `null` als Platzhalter bzw. den NULL-Wert). Durch die Ermittlung von Schemaabbildungen wird festgestellt, dass `E2` die höchste Ähnlichkeit zu `E3` besitzt. Dadurch werden die Datenschutzeinstellungen für die Attribute *firstname* und *lastname* auf `PRIVATE` gesetzt, für *uid, x, y, z* und *direction* hingegen auf `PUBLIC` (siehe Spalte *Mapping*). Die Einstellungen für die Attribute *timestamp* und *tid* (die ID des aktuell gespielten Titels) konnten durch die Abbildung nicht ermittelt werden.

Attribute	initial	Mapping	Data Mining	final
uid	`null`	PUBLIC	PUBLIC	PUBLIC
firstname	`null`	PRIVATE	PRIVATE	PRIVATE
lastname	`null`	PRIVATE	PRIVATE	PRIVATE
x	`null`	PUBLIC	PUBLIC	PUBLIC
y	`null`	PUBLIC	PUBLIC	PUBLIC
z	`null`	PUBLIC	PUBLIC	PUBLIC
timestamp	`null`	`null`	PUBLIC	PUBLIC
direction	`null`	PUBLIC	PUBLIC	PUBLIC
tid	`null`	`null`	`null`	PRIVATE

Durch Anwendung von Data Mining wird anschließend ermittelt, dass die Attribute *lastname, x, y* und *z* in `E1` zusammen mit *timestamp* mit den gleichen Datenschutzeinstellungen (in diesem Fall `PUBLIC`) verwendet werden. Daher wird für *timestamp* die Freigabe ebenfalls auf `PUBLIC` gesetzt (siehe Spalte *Mining*). Da *tid* von keinem der bisher bekannten Geräte verwendet wird, erfolgt eine Belegung mit dem Standardwert `PRIVATE`. Das endgültige Datenschutzprofil ist in der obigen Tabelle in der Spalte *final* hinterlegt.

PP4SE in der XML-Repräsentation

Jede mit PP4SE definierte Richtlinie besteht aus einer Liste (`apps`) von Anwendungen (`application`):

```
<xs:element name="policy">
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="apps" minOccurs="0" maxOccurs="1"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>

<xs:element name="apps">
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="application" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>
```

Unter Anwendungen werden Computerprogramme, Hintergrundprozesse eines Betriebssystems oder Assistenzsysteme als Ganzes aufgefasst. Eine Anwendung besteht aus der Beschreibung, der `app_description`, einer eindeutigen ID (`app_ID`) und einer Liste (`modules`) von bereitgestellten Funktionalitäten, genannt `module`:

```
<xs:element name="application">
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="app_description" minOccurs="0" maxOccurs="1"/>
   <xs:element ref="modules" minOccurs="0" maxOccurs="1"/>
  </xs:sequence>
  <xs:attribute ref="app_ID" use="required"/>
 </xs:complexType>
</xs:element>

<xs:attribute name="app_ID" type="xs:string"/>
<xs:element name="app_description" type="xs:string" />

<xs:element name="modules">
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="module" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>
```

Ein Modul besteht aus seiner Beschreibung (`module_description`), einer eindeutigen ID (`module_ID`), dem Typ des Moduls (`module_type`) und der Angabe, ob dieses Modul zwingend erforderlich für die Anwendung ist (`module_required`). Für jedes Modul wird eine Liste (`attributeList`) von verwendeten bzw. angefragten Attributen (`attribute`) erfasst:

```
<xs:element name="module">
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="module_description" minOccurs="0" maxOccurs="1"/>
   <xs:element ref="attributeList" minOccurs="0" maxOccurs="1"/>
  </xs:sequence>
  <xs:attribute ref="module_ID" use="required"/>
  <xs:attribute ref="module_type" use="required"/>
  <xs:attribute ref="module_required" use="optional"/>
 </xs:complexType>
</xs:element>

<xs:attribute name="module_ID" type="xs:string"/>
<xs:attribute name="module_type" type="xs:string"/>
<xs:attribute name="module_required" type="xs:boolean"/>
<xs:element name="module_description" type="xs:string" />

<xs:element name="attributeList">
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="attribute" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>
```

Ein Attribut wird durch seinen Namen (`name`) identifiziert. Es werden Angaben erfasst, ob das Attribut direkt (`allow`) oder indirekt durch Dritt-Anwendungen (`third_party_access`) angefragt werden darf. Zudem können allgemeine Bedingungen (`condition`), Aggregationsstufen (`aggregation`) und das Anfrageintervall (`interval`) festgelegt werden, unter denen das Attribut freigegeben wird. Falls der Nutzer diese Angaben nicht explizit angeben möchte, kann auch die Stufe der Anonymisierung (`privacyLevel`) direkt angegeben werden:

```
<xs:element name="attribute">
 <xs:complexType>
  <xs:choice>
   <xs:sequence>
    <xs:element ref="allow" minOccurs="0" maxOccurs="1"/>
    <xs:element ref="third_party_access" minOccurs="0" maxOccurs="1"/>
    <xs:element ref="condition" minOccurs="0" maxOccurs="1"/>
    <xs:element ref="aggregation" minOccurs="0" maxOccurs="1"/>
    <xs:element ref="interval" minOccurs="0" maxOccurs="1"/>
   </xs:sequence>
   <xs:element ref="privacyLevel" minOccurs="0" maxOccurs="1"/>
  </xs:choice>
  <xs:attribute ref="name" use="required"/>
 </xs:complexType>
</xs:element>

<xs:attribute name="name" type="xs:string"/>
<xs:element name="allow" type="xs:boolean"/>
<xs:element name="third_party_access" type="xs:boolean"/>
<xs:element name="interval" type="xs:string" />
```

Für die allgemeinen Freigabestufen lassen sich je nach Anwendungsumgebung eigene Werte spezifizieren. Typische Klassifizierungsebenen sind beispielsweise *öffentlich* (`public`), *vertraulich* (`confidential`), *geheim* (`secret`) und *streng geheim* (`top secret`). Durch studentische Projektarbeiten inspiriert, kann im `privacyLevel` alternativ auch die Angabe der Parameter für k-Anonymität, l-Diversität und t-closeness erfolgen (Details siehe [RLRG16]):

```
<xs:element name="privacyLevel">
 <xs:simpleType>
  <xs:restriction base="xs:string">
   <xs:enumeration value="public"/>
   <xs:enumeration value="confidential"/>
   <xs:enumeration value="secret"/>
   <xs:enumeration value="top-secret"/>
  </xs:restriction>
 </xs:simpleType>
</xs:element>
```

Eine allgemeine Freigabebedingung lässt sich in atomare (`atomicCondition`) und zusammengesetzte Bedingungen unterteilen. Eine atomare Bedingung lässt sich nicht weiter untergliedern. Für die Formulierung der Bedingungen lassen sich – je nach Anwendungsgebiet – beispielsweise die in der `WHERE`-Klausel von SQL oder die in Object Constraint Language [RG98] verwendeten Konstrukte verwenden. Eine *UND*-Bedingung (`andCondition`) besteht aus mindestens zwei atomaren Bedingungen oder *ODER*-Bedingungen (`orCondition`). Gleiches gilt für eine *ODER*-Bedingung. Bei der internen Verarbeitung der Bedingungen werden die Klauseln zur besseren Auswertung in eine disjunktive Normalform überführt:

```
<xs:element name="atomicCondition" type="xs:string" />

<xs:element name="andCondition">
 <xs:complexType>
  <xs:choice minOccurs="2" maxOccurs="unbounded">
   <xs:element ref="atomicCondition" minOccurs="1" maxOccurs="unbounded"
   />
   <xs:element ref="orCondition" minOccurs="1" maxOccurs="unbounded"/>
  </xs:choice>
 </xs:complexType>
</xs:element>

<xs:element name="orCondition">
 <xs:complexType>
  <xs:choice minOccurs="2" maxOccurs="unbounded">
   <xs:element ref="atomicCondition" minOccurs="1" maxOccurs="unbounded"
   />
   <xs:element ref="andCondition" minOccurs="1" maxOccurs="unbounded"/>
  </xs:choice>
 </xs:complexType>
</xs:element>
```

Bei der Aggregation werden die erlaubten Aggregationsfunktionen (`aggregationType`) und die dazugehörigen Gruppierungsattribute (`groupBy`) angegeben. Die im unteren Quellcodeausschnitt angegeben Aggregatfunktionen dienen nur als Beispiele – denkbar sind alle von Datenbanksystemen unterstützten Funktionen bzw. auch nutzerdefinierte Prozeduren und Funktionen:

```
<xs:element name="aggregation">
 <xs:complexType>
  <xs:sequence minOccurs="1" maxOccurs="1">
   <xs:element ref="aggregationType" minOccurs="1" maxOccurs="1"/>
   <xs:element ref="groupBy" minOccurs="0" maxOccurs="1"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>

<xs:element name="groupBy" type="xs:string" />

<xs:element name="aggregationType">
 <xs:simpleType>
  <xs:restriction base="xs:string">
   <xs:enumeration value="min"/>
   <xs:enumeration value="max"/>
   <xs:enumeration value="sum"/>
   <xs:enumeration value="count"/>
   <xs:enumeration value="avg"/>
  </xs:restriction>
 </xs:simpleType>
</xs:element>
```

B.2 Anonymisierungsverfahren im Postprozessor

B.2.1 Generalisierung und Differential Privacy

Im Postprozessor werden aus der Klasse der Generalisierungs-basierten Verfahren die Standardalgorithmen zur Gewährleistung der Anonymitätsmaße k-Anonymität, l-Diversität und t-closeness unterstützt. Details zu deren Implementierung sind in den studentischen Projekt- bzw. Abschlussarbeiten von [RLRG16] bzw. [Joh16] zu finden. Gleiches gilt für Differential Privacy, welches am Beispiel des Münzwurf-basierten Verfahrens im Rahmen eines studentischen Projektes [TL17] implementiert wurde.

Beide Verfahren können zur Anonymisierung der Attributwerte das Konzept der Generalisierungsbäume nutzen. Im PArADISE-Framework werden sowohl kategorische als auch numerische Attribute sowie deren verallgemeinerte Werte durch Generalisierungsbäume repräsentiert. Die Umsetzung erfolgt durch zusätzliche Relationen, die parallel zu den Anwendungsdaten im Datenbanksystem gespeichert werden. In dieser Relation wird für jedes Attribut jeder Basisrelation angegeben, welcher übergeordnete Wert bei der Anonymisierung als Ersatz verwendet werden soll. Dies wird solange rekursiv angegeben, bis die Wurzel des Generalisierungsbaumes erreicht ist. Am Beispiel des Attributs *Genre* wird in Tabelle B.1 die relationale Darstellung der DGH aus Abbildung 2.5 dargestellt.

Zusätzlich werden benutzerdefinierte Funktionen (User-defined Functions; UDF) als Schnittstellen bereitgestellt, damit Anonymisierungsmethoden auf die Hierarchie Zugriff erhalten. Durch diesen modularen Aufbau können leicht weitere Anonymisierungskonzepte, welche auf Generalisierung beruhen, in das PArADISE-Framework integriert werden, ohne den genauen Datenbestand zu kennen. Die Implementierung für verschiedene relationale Datenbanken wird im technischen Bericht von Martin Müller [Mül16] näher erläutert.

dgh_id	value	dgh_id_children	value_children
1	***	1	Blues
1	***	1	Metal
...	...	...	...
1	Blues	1	Bluesrock
1	Blues	1	West Coast Blues
...	...	...	...
1	Metal	1	Death Metal
1	Metal	1	Doom Metal
1	Metal	1	Viking Metal
...	...	...	...
1	Death Metal	1	Death 'n Roll
1	Death Metal	1	Technical Death Metal
...	...	...	...
1	Doom Metal	1	Death Doom
1	Doom Metal	1	Stoner Doom
...	...	...	...

Tabelle B.1: Die Tabelle zeigt die relationale Darstellung des DGH-Baumes aus Abbildung 2.5. Die Attributwerte für *dgh_id* bzw. *dgh_id_children* verweisen auf das betroffene Attribut Genre aus der Musikdatenbank. Die Relation ist so konzipiert, dass bestehende DGHs als Teilbäume für andere DGHs dienen können, wenn für *dgh_id* und *dgh_id_children* unterschiedliche Werte verwendet werden. Die Attribute *value* und *value_children* kennzeichnen die Beziehung zwischen dem allgemeineren bzw. spezielleren Wert. Alle Werte, die in *value_children*, aber nicht in *value* auftauchen, sind die Blattknoten des DGH-Baumes.

B.2.2 Permutation

Als weitere Klasse von Anonymisierungsverfahren wird die Permutation durch die Implementierung des Slicing-Verfahrens [LLZM12] unterstützt. Für das Verfahren wurde eine Implementierung für relationale Datenbanken erarbeitet, welche in [GH15c] näher vorgestellt wurde. Der Algorithmus realisiert das Slicing in vier Schritten:

Horizontale Partitionierung: Zunächst wird die gegebene Relation R in n disjunkte Partitionen $R_1, \ldots, R_n$ fragmentiert. Für jede dieser Partitionen R_i wird ein Selektionsprädikat c_i formuliert, welches die Tupel aus R auf eine Partition R_i allokiert:

$$R_1 := \sigma_{c_1}(R), ..., R_n := \sigma_{c_n}(R). \tag{B.1}$$

Die Selektionsbedingungen werden durch den Postprozessor dabei auf Basis der Quasi-Identifikatoren und den daraus resultierenden Äquivalenzklassen gewählt. Eine einzelne Selektionsbedingung kann dabei entweder eine Punktselektion der Form $\sigma_{attributname='Wert'}$ bzw. eine Bereichsselektion der Form $\sigma_{x \leq attributname < y}$ sein. Müssen feingranularere Äquivalenzklassen gebildet werden, so können auch komplexere Selektionsprädikate über mehrere Attribute gebildet werden, solange die Partitionen vollständig und disjunkt sind.

Vertikale Partitionierung: Im zweiten Schritt werden die Teilrelationen R_i durch eine feste Anzahl m zuvor definierter Projektionen weiter partitioniert. Zu jedem R_i werden Projektionen der Form

$$R_{ij} := \pi_{A_j}(R_i) \tag{B.2}$$

mit $\forall j \in \mathbb{N} : A_j \subset R \wedge 1 \leq j \leq m$ aufgestellt.

Die gewählten Attributmengen A_j müssen nicht unbedingt disjunkt sein. Zwar fordert der Slicing-Algorithmus von Li et al. [LLZM12] Disjunktheit, diese Forderung kann aber soweit entschärft werden, dass im Schnitt von

je zwei Attributmengen A_i und A_j keine quasi-identifizierende Attributmenge qi aus der Menge der bekannten Quasi-Identifikatoren QI enthalten ist:

$$\forall qi \in QI \ \nexists (A_i, A_j) : A_i \cap A_j \supseteq qi. \tag{B.3}$$

Im Folgenden wird o. B. d. A. davon ausgegangen, dass die Partitionierung disjunkt ist. Grundsätzlich sollte keine der Attributmengen so gewählt werden, dass sie einen Quasi-Identifikator enthält, da sonst Implikationen auf die Identität einer Person bzw. dessen weiterer Merkmale gezogen werden kann. Es müssen somit die Attribute eines jeden minimalen Quasi-Identifikators über der Relation `R` auf mindestens zwei Attributmengen verteilt werden. Zudem sollte die Auswahl der Attributmengen in Abstimmung mit dem erforderlichem Informationsbedarf der konkreten Anwendung getroffen werden. Dadurch kann sichergestellt werden, dass die Auswertung multivariater Verfahren, wie die Ermittlung eines Korrelationskoeffizienten, weiterhin möglich ist.

Permutation: Die konkrete Anonymisierung erfolgt beim Slicing-Verfahren durch die Permutation der entstandenen Teilrelationen R_{ij}. Die Permutation kann durch eine einfache Sortierung nach den Attributwerten der Teilrelationen oder nach zuvor erzeugten Zufallswerten für jedes Tupel erfolgen. Während das einfache Sortieren einfach realisierbar ist, kann es jedoch zu unerwünschten Seiteneffekten, wie Scheinkorrelationen, führen. Dies ist bei einer zufälligen Sortierung nicht der Fall.

Beispiel: Scheinkorrelationen bei Sortierung nach Attributwerten beim Slicing

Die folgende Tabelle zeigt einen Ausschnitt aus der Musikdatenbank vor Anwendung des Slicings. In den beiden darauffolgenden Tabellen wurden zwei Teilrelationen mit den Attributen `Künstler` und `Jahr` bzw. `Länge` und `Titel` gebildet. Die Permutation erfolgte durch die Sortierung der Attribute `Jahr` bzw. `Länge` in aufsteigender Reihenfolge. Durch die Ordnung der Attributwerte wird der Eindruck vermittelt, dass die Länge von Musikstücken mit zunehmendem Alter abnimmt.

Künstler	**Jahr**	**Länge**	**Titel**
Rosenstolz	1995	270	Samstags
Rosenstolz	1995	208	Mann im Ohr
Gonzales	2000	195	Figga Please
Genesis	1991	286	Home By The Sea
Genesis	1991	264	That's All
Cat Stevens	1971	247	Peace Train
Cat Stevens	1971	170	Moonshadow
ABBA	1992	163	Waterloo
ABBA	1992	232	Thank You For The Music
ABBA	1992	232	Dancing Queen
Cher	1972	151	Let Me Down Easy
Cher	1972	166	Half Breed

Künstler	Jahr
Cat Stevens	1971
Cat Stevens	1971
Cher	1972
Cher	1972
Genesis	1991
Genesis	1991
ABBA	1992
ABBA	1992
ABBA	1992
Rosenstolz	1995
Rosenstolz	1995
Gonzales	2000

Länge	Titel
151	Let Me Down Easy
163	Waterloo
166	Half Breed
170	Moonshadow
195	Figga Please
208	Mann im Ohr
232	Dancing Queen
232	Thank You For The Music
247	Peace Train
264	That's All
270	Samstags
286	Home By The Sea

Unabhängig von der Sortierstrategie und den gewählten `Attributen` wird die Permutation durch den Ordnungsoperator τ auf den Teilrelationen R_{ij} realisiert. Als Ergebnis der Permutation erhalten wir eine geordnete Liste L_{ij}:

$$L_{ij} := \tau_{<Attribute>}(R_{ij}). \tag{B.4}$$

Dies stellt jedoch eine Verletzung der relationalen Algebra dar, da der Ordnungsoperator τ nur als letzter Operator eines Ausdruckes der Relationenalgebra verwendet werden darf [GM08]. Um die Listen wieder zurück in Relationen zu überführen und die Sortierreihenfolge aufrecht zu erhalten, wird jedes Tupel mit einer fortlaufenden Ordnungszahl `Ord`, welche die Position in der Liste repräsentiert, zu einer neuen Relation R'_{ij} verknüpft[4]:

$$R'_{ij} := \pi_{Ord,*}(L_{ij}). \tag{B.5}$$

Verbund und Vereinigung: Zur Rekonstruktion der Teilrelationen R'_{ij} zu einer anonymisierten Relation $r'(R)$ (kurz: R') über dem Relationenschema R müssen zunächst die permutierten Relationen R'_{ij} mittels natürlichem Verbund über die generierten Ordnungszahlen `Ord` verknüpft werden:

$$R'_i := \pi_{A_1,\ldots,A_o}(R'_{i1} \bowtie_{R'_{i1}.Ord=R'_{i2}.Ord} R'_{i2} \ldots \bowtie_{R'_{i1}.Ord=R'_{im}.Ord} R'_{im}). \tag{B.6}$$

Dabei wird die intern genutzte Ordnungszahl durch die Projektion ($\pi_{A_1,\ldots,A_o}$, wobei o die Anzahl der Attribute in R ist) wieder entfernt. Abschließend werden die permutierten Partitionen R'_i mengentheoretisch vereinigt:

$$R' := \bigcup_{i=1}^{n} R'_i. \tag{B.7}$$

Die Relation $r'(R)$ stellt das Ergebnis der Anonymisierung dar. Weitere Details zu dem Verfahren und dessen Implementierung können dem Artikel [GH15c] entnommen werden.

Komplexität des Verfahrens: Der Aufwand für die Partitionierung steigt linear mit der Anzahl der Partitionen. Dies betrifft sowohl die horizontale, als auch die vertikale Partitionierung. Gleiches gilt auch für den abschließenden Verbund bzw. die abschließende Vereinigung; hier sind die Laufzeiten jedoch nicht ausschlaggebend.

Die meiste Rechenzeit wird für die Permutation benötigt. Bei einer höheren Anzahl an Partitionen sinkt die Zeit für die Permutation. Durch die geringe Anzahl an Tupeln pro Partition ist der Aufwand für das Sortieren/Permutieren innerhalb einer Partition geringer.

[4] Die Syntax der Gleichung lehnt sich an die Syntax aus relationalen DBMS zur Ermittlung der internen ID bzw. Zeilennummer eines Tupels an, da dies ebenso in der SQL-basierten Implementierung [GH15c] genutzt wird.

Besteht die Relation R aus insgesamt n Tupeln, die in x Partitionen fragmentiert werden, so enthält, unter Berücksichtigung einer Gleichverteilung, jede Partition ca. $\frac{n}{x} = m$ Tupel. Für die Permutation jeder dieser x Partitionen ergibt sich eine Komplexität von

$$\begin{aligned} O(x * log(m) * m) &= O(x * log(\frac{n}{x}) * \frac{n}{x}) \\ &= O(n * (log(n) - log(x))) \\ &= O(n * log(n) - n * log(x)). \end{aligned} \tag{B.8}$$

Da die Anzahl der Tupel konstant bleibt, sinkt die Laufzeit für die Permutation mit steigender Zahl an Partitionen. Die Partitionen lassen sich einfacher permutieren bzw. sortieren, da die Zahl der zu sortierenden Tupel sinkt. Die Komplexität der Permutation ergibt sich aus der Wahl des Sortierverfahrens seitens des DBMS. Die Komplexitäten für das Partitionieren, dem natürlichen Verbund[5] und der mengentheoretischen Vereinigung sind lediglich linear und haben nur einen geringen Einfluss auf die Gesamtkomplexität bzw. -laufzeit.

Slicing stellt eine interessante Alternative zu Generalisierungstechniken und Differential Privacy dar. Insbesondere multivariate Verfahren, wie Regressions- und Korrelationsanalysen, können ohne Informationsverlust bzw. Verfälschung der Ergebnisse ausgeführt werden, sofern die zu untersuchenden Attribute nicht durch die vertikale Partitionierung aufgetrennt wurden. Sind jedoch die Daten einzelner Tupel Gegenstand einer Analyse, so eignet sich das Verfahren nur bedingt, da keine verlässlichen Aussagen mehr getroffen werden können. Auf Basis einer gegebenen Analyse, beispielsweise in Form einer SQL-Anfrage, und dem erforderlichen Informationsbedarf muss eine Entscheidung für ein Anonymisierungsverfahren und dessen Parametrisierung getroffen werden.

Entscheidung für ein passendes Anonymisierungsverfahren

Nach [DDK+15] bestehen zwei Sichtweisen auf die Anonymisierung von Daten: Der *Utility-first-Ansatz* verwendet heuristische Methoden zur Parameterwahl, sodass zunächst ein hoher Datennutzen gewährleistet wird, bevor das Risiko zur Deanonymisierung bestimmt wird. Ist dieses Risiko zu hoch, wird das Anonymisierungsverfahren so angepasst, dass die Nützlichkeit minimal verringert wird, das Datenschutzniveau dafür steigt. Laut [DDK+15] wird der Utility-first-Ansatz insbesondere von Anbietern von statistischen Informationen bevorzugt, da dies ihr Geschäftsmodell ist.

Beim *Privacy-first-Ansatz* wird zunächst garantiert, dass ein Mindestmaß an Datenschutz garantiert wird. Dazu werden die entsprechenden Parameter des gewählten Anonymisierungsverfahren auf die geforderten Werte gesetzt. Ist der Datennutzen durch das Anonymisierungsverfahren zu gering, soll entweder ein anderes Verfahren gewählt werden, welches mehr Nutzen verspricht oder, sofern möglich, die Parameter des Anonymisierungsverfahrens weniger strikt gewählt werden, auch wenn dies das Risiko einer Deanonymisierung hervorbringen könnte.

Entscheidungsalgorithmus in PArADISE: Es existiert keine universale Lösung bzw. kein bestes Verfahren, um das Ergebnis einer Anfrage zu anonymisieren, da der verbleibende Nutzen stark von der Anfrageart abhängt. Werden vorrangig spaltenbasierte Analyseverfahren, wie z. B. lineare Regressionen, Korrelationsanalysen oder auch einfache Minimal- und Maximalwertberechnungen ausgeführt, so sind Verfahren basierend auf Permutation und Differential Privacy besser geeignet als Generalisierungsverfahren, da die Originalwerte bzw. deren Verteilung erhalten bleibt. Steht jedoch das einzelne Tupel im Vordergrund, so bietet Generalisierung einen höheren Datennutzen, da mehr Informationen pro Tupel erhalten bleiben. Eine detaillierte Beschreibung des in PArADISE verwendeten Entscheidungsalgorithmus für Anonymisierungsverfahren ist der Bachelorarbeit von Bodo John [Joh16] zu finden.

[5] Da die Tupel in den Teilrelationen 1-zu-1 miteinander verknüpft werden, können die Relationen linear durchlaufen werden.

B.3 Kullback-Leibler-Divergenz

In der Praxis hat sich die Kullback-Leibler-Divergenz [KL51] als geeignetes Maß zur Berechnung des Informationsverlustes von anonymisierten Daten hervorgetan [HS10, LLBW11]. Hardt et al. [HLM12] schlagen zudem die Verwendung der KLD für die Messung des Informationsverlustes für Differential Privacy vor.

Aufbauend auf den Definitionen für Entropie und Informationsgehalt lässt sich die Kullback-Leibler-Divergenz als Distanzmaß für die Messung des Informationsverlustes von einer Originalrelation R zu einer anonymisierten Relation R' ausnutzen.

Für die weiteren Betrachtungen gehen wir von folgenden Begriffsbestimmungen aus:

- R, R' = Relationenschemata
- A_i = Attribut eines Quasi-Identifikators
- O = weitere Attribute
- C = Attribut(e) für Annotationen oder zur Klassifikation der Tupel von R.
- a_{ij} = ein Attributwert j aus dem Wertebereich des Attributs A_i
- $|A_i|$ = Mächtigkeit des Wertebereiches für das Attribut A_i
- $freq$ = Funktion zur Berechnung der relativen Häufigkeit des Attributwertes

Das originale Relationenschema R ist definiert als:

$$R := (A_1, A_2, \ldots, A_n, O, C) \tag{B.9}$$

und das anonymisierte Relationenschema R' als:

$$R' := (A'_1, A'_2, \ldots, A'_n, O, C). \tag{B.10}$$

Die Daten unterscheiden sich somit nur in der Belegung von den Attributen A_i, welche in den anonymisierten Attributwerten von A'_i hinterlegt sind. Die Messung des Informationsverlustes für ein Attribut, d.h., wie stark sich der Datenbestand für dieses verändert hat, erfolgt mittels folgender Formel:

$$KLD(A_i||A'_i) := \sum_{j=1}^{|A_i|} freq(a_{ij}) * log_2 \left(\frac{freq(a_{ij})}{freq(a'_{ij})} \right). \tag{B.11}$$

Zur Normierung des Informationsverlustes auf einen Wert zwischen 0 und 1 wird der errechnete Wert anschließend durch die auftretende Entropie $H(A_i)$ für das Attribut A_i geteilt. Die Entropie $H(A_i)$ ist wie folgt definiert:

$$H(A_i) := \sum_{j=1}^{|A_i|} freq(a_{ij}) * log_2(freq(a_{ij})). \tag{B.12}$$

Aus dem Informationsverlust $KLD(A_i||A'_i)$ und der Entropie $H(A_i)$ für das Attribut A_i lässt sich der normalisierte Informationsverlust $KLD_N(A_i||A'_i)$ berechnen:

$$KLD_N(A_i||A'_i) := \frac{KLD(A_i||A'_i)}{H(A_i)}. \tag{B.13}$$

Die Summe der normierten Kullback-Leibler-Divergenzen über alle Attribute A_i aus R ergeben den gesamten Informationsverlust KLD_R der anonymisierten Relation $r(R')$ im Verhältnis zur unveränderten Relation $r(R)$:

$$KLD_R := \sum_{i=1}^{n} KLD_N(A_i||A'_i). \tag{B.14}$$

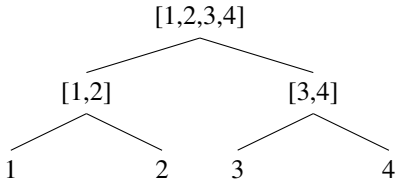


Abbildung B.1: Generalisierungsbaum für die Anonymisierung eines Attributes. Die Werte 1 und 2 werden zu $[1, 2]$, die Werte 3 und 4 zu $[3, 4]$ zusammengefasst. In der zweiten Stufe werden die Werte zu $[1, 2, 3, 4]$ verallgemeinert.

In [Nie13] werden die Nachteile der KL-Divergenz für den Einsatz zur Messung der Anonymität aufgezeigt. Dazu zählt beispielsweise die fehlende Definition für Attributwerte, die in einer Relation nicht vorkommen sowie die nicht einbezogene Semantik der Attribute in die Abstandsberechnung. Durch die nachfolgend vorgeschlagenen Modifikationen der KL-Divergenz lassen sich diese Probleme jedoch umgehen.

Weitere Verfahren, wie beispielsweise die *Sum of Squared Errors*, eignen sich insbesondere für verrauschte Daten, aber nicht für generalisierte Daten [SCDFSM14]. Die Kullback-Leibler-Divergenz eignet sich, mit den im Folgenden vorgestellten Anpassungen, für jede Art von Anonymisierung und auch zur Berechnung des Informationsverlustes durch Projektionen und Selektionen. Um die Kullback-Leibler-Divergenz für die Anonymisierung beliebiger, d. h. auch kategorischer, Werte nutzbar zu machen, wurden drei Änderungen vorgenommen:

1. Die Frequenz für generalisierte Werte wird durch die Addition der zum generalisierten Wert gehörigen Werte bestimmt.
2. Die Normalisierung erfolgt für alle Berechnungen auf Basis der Verteilung des Originaldatenbestandes.
3. Bei mehrmaliger Generalisierung wird der Informationsverlust zum bisherigen Informationsverlust aufsummiert.

Im Folgenden werden die Anpassungen genauer beschrieben. Anhand von einfachen Rechenbeispielen wird die jeweilige Auswirkung der Anpassungen verdeutlicht. Abschließend wird gezeigt, wie die angepasste Kullback-Leibler-Divergenz zusammen mit Domänengeneralisierungshierarchien für einen iterativen Anonymisierungsalgorithmus genutzt werden kann.

Anpassung der Kullback-Leibler-Divergenz: Abbildung auf Wertebereich: Damit die Kullback-Leibler-Divergenz auch für durch Generalisierung anonymisierte kategorische Daten verwendet werden kann, wurde eine kleine Anpassung an der Berechnungsvorschrift vorgenommen. Betrachten wir die Generalisierungshierarchie in Abbildung B.1, so kommt der generalisierte Wert [3,4] nicht im Originaldatenbestand vor. Folglich kann die Kullback-Leibler-Divergenz nicht angewendet werden, da eine Division durch 0 erfolgt.

Um dieses Problem zu umgehen, wurde die $freq$-Funktion wie folgt abgeändert: Statt die Anzahl der Vorkommen eines Attributwertes zu zählen (die im generalisierten Fall immer 0 beträgt), wird der Wert auf den generalisierten Wert abgebildet. Die Anzahl für den generalisierten Wert ergibt sich aus der Summe der einzelnen Vorkommen des Originaldatenbestandes:

$$freq([x, y]) = \sum_{i=x}^{y} freq(i) = freq(x) + ... + freq(y). \tag{B.15}$$

Beispiel: Wertebereichsabbildungen für die KLD

Soll beispielsweise die Anzahl der Elemente für den generalisierten Wert $[3, 4]$ berechnet werden, so muss die Anzahl der Elemente mit den Werten 3 bzw. 4 berechnet und anschließend addiert werden:

$$freq(3) \rightarrow freq([3,4]) = freq(3) + freq(4) \tag{B.16}$$

Gleiches gilt auch für Elemente, die in der Generalisierungshierarchie höher angesiedelt sind. Die Anzahl für das Wurzelelement $[1,2,3,4]$ ergibt sich aus der Summe seiner Kinder $[1,2]$ und $[3,4]$, die sich wiederum aus den einzelnen Werten für die Elemente 1, 2, 3 und 4 ergeben:

$$freq(3) \rightarrow freq([1,2,3,4]) = freq([1,2]) + freq([3,4]) = freq(1) + freq(2) + freq(3) + freq(4) \tag{B.17}$$

Durch diese Anpassung eignet sich die Kullback-Leibler-Divergenz nicht nur für numerische Daten, sondern auch für kategorische Attribute. Dadurch lassen sich z. B. Nationalstaaten in verschiedene geografische Teilregionen einordnen, diese wiederum in Kontinente, usw., ohne dass die Kullback-Leibler-Divergenz ungültige Berechnungen durchführen muss.

Beispiel: Einfache Generalisierungshierarchie für nummerische Werte

Die folgende Tabelle zeigt einen Beispieldatensatz für ein Attribut A mit zwei möglichen Ausprägungen der Generalisierung. Die erste Spalte zeigt den originalen Wert ohne vorherige Generalisierung, die zweite Spalte den Wert nach einmaliger, die dritte Spalte nach zweimaliger Generalisierung gemäß dem Generalisierungsbaum in Abbildung B.1.

A	A'	A''
1	[1, 2]	[1,2,3,4]
2	[1, 2]	[1,2,3,4]
2	[1, 2]	[1,2,3,4]
3	[3, 4]	[1,2,3,4]
3	[3, 4]	[1,2,3,4]
3	[3, 4]	[1,2,3,4]
4	[3, 4]	[1,2,3,4]
4	[3, 4]	[1,2,3,4]
4	[3, 4]	[1,2,3,4]
4	[3, 4]	[1,2,3,4]

Ausgehend von der obigen Tabelle kann die relative Häufigkeit für die einzelnen Werte von A, bzw. von A' und A'', berechnet werden. Das Ergebnis ist in der folgenden Tabelle für beide Generalisierungen dargestellt:

Wert	$freq(a_j)$	$freq(a'_j)$	$freq(a''_j)$
1	$\frac{1}{10} = 0,1$	$\frac{3}{10} = 0,3$	$\frac{10}{10} = 1,0$
2	$\frac{2}{10} = 0,2$	$\frac{3}{10} = 0,3$	$\frac{10}{10} = 1,0$
3	$\frac{3}{10} = 0,3$	$\frac{7}{10} = 0,7$	$\frac{10}{10} = 1,0$
4	$\frac{4}{10} = 0,4$	$\frac{7}{10} = 0,7$	$\frac{10}{10} = 1,0$

Ausgehend von den relativen Häufigkeiten ergeben sich folgende Divergenzen:

- Vom Originaldatenbestand A zur ersten Generalisierungsstufe A':

$$\begin{aligned} KLD(A||A') &= 0,1 * log_2 \frac{0,1}{0,3} + 0,2 * log_2 \frac{0,2}{0,3} \\ &+ 0,3 * log_2 \frac{0,3}{0,7} + 0,4 * log_2 \frac{0,4}{0,7} \\ &\approx -0,96515 \end{aligned} \tag{B.18}$$

- Von der ersten Generalisierungsstufe A' zur zweiten Generalisierungsstufe A'':

$$\begin{aligned} KLD(A'||A'') &= 0,3 * log_2 \frac{0,3}{1,0} + 0,7 * log_2 \frac{0,7}{1,0} \\ &\approx -0,88122 \end{aligned} \tag{B.19}$$

- Vom Originaldatenbestand A direkt zur zweiten Generalisierungsstufe A'':

$$\begin{aligned} KLD(A||A'') &= 0,1 * log_2 \frac{0,1}{1,0} + 0,2 * log_2 \frac{0,2}{1,0} \\ &\quad + 0,3 * log_2 \frac{0,3}{1,0} + 0,4 * log_2 \frac{0,4}{1,0} \\ &\approx -1,84644 \end{aligned} \tag{B.20}$$

Anpassung der Kullback-Leibler-Divergenz: Normalisierung: Bisherige Untersuchungen für den Informationsverlust von anonymisierten Daten nutzten die Kullback-Leibler-Divergenz zum Vergleich von verschiedenen Generalisierungen. Dabei sollte aus einer Auswahl von mehreren Anonymisierungen diejenige ausgewählt werden, die den geringsten Informationsverlust aufweist. Die Normalisierung findet in diesem Szenario jeweils auf dem zuletzt generalisierten Datenbestand bzw., wenn noch keine Generalisierung erfolgt ist, auf dem Originaldatenbestand statt. Dadurch lassen sich zwar die einzelnen Generalisierungsschritte gut miteinander vergleichen, doch der gesamte Informationsverlust ist so nicht messbar, da sich die Verteilung der Daten mit jedem Schritt ändert. Aus diesem Grund wird in PArADISE die Kullback-Leibler-Divergenz immer auf Basis der Originalverteilung berechnet.

Beispiel: Angepasste Normalisierung der Kullback-Leibler-Divergenz

Der Normalisierungfaktor $H(A)$ basierend auf der Verteilung der Originalwerte von A sieht für das laufende Beispiel wie folgt aus:

$$\begin{aligned} H(A) &= 0,1 * log_2(0,1) + 0,2 * log_2(0,2) \\ &\quad + 0,3 * log_2(0,3) + 0,4 * log2(0,4) \\ &\approx -1,84664 \end{aligned} \tag{B.21}$$

Dies ergibt folgende normalisierten Werte für die oben berechneten Divergenzen von $KLD(A||A')$, $KLD(A'||A'')$ bzw. $KLD(A||A'')$:

$$\begin{aligned} KLD_N(A||A') &= \frac{-0,96515}{-1,84644} \approx 0,52270 \\ KLD_N(A'||A'') &= \frac{-0,88122}{-1,84644} \approx 0,47729 \\ KLD_N(A||A'') &= \frac{-1,84644}{-1,84644} = 1,0 \end{aligned} \tag{B.22}$$

Der Informationsverlust von den Originaldaten A zu den vollständig generalisierten Daten A" beträgt somit 100%, wodurch kein Nutzen mehr aus den Daten gezogen werden kann. Wird hingegen die Zwischenstufe der Generalisierung betrachtet, so beträgt der Informationsverlust lediglich 52,2%. Wird aus der ersten Anonymisierung die weitere Generalisierung zur Wurzel des Generalisierungsbaumes ausgeführt, so steigt der Informationsverlust um weitere 47,8% (ausgehend vom Originaldatenbestand). Der gesamte Informationsverlust liegt somit ebenfalls bei 100%.

Anpassung der Kullback-Leibler-Divergenz: Additivität: Durch die Normierung der Kullback-Leibler-Divergenz auf die Originalverteilung anstatt auf die jeweils vorherige Generalisierung wurde erreicht, dass die Kullback-Leibler-Divergenz additiv berechnet werden kann (siehe Abbildung B.2). Dadurch lassen sich die einzelnen Ebenen der Generalisierung schrittweise nacheinander berechnen, anstatt jede Berechnung wieder auf die Originalverteilung aufzusetzen.

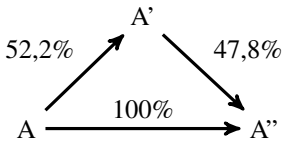


Abbildung B.2: Additivität der Kullback-Leibler-Divergenz

Beweis Additivität. Sei o_i mit $1 \leq i \leq n$ die Frequenz für einen Originalwert aus A mit Mächtigkeit n und g_j mit $1 \leq j \leq m$ bzw. g'_j mit $1 \leq j \leq m'$ die Frequenz für einen generalisierten Wert, wobei m bzw. m' die Mächtigkeiten der generalisierten Domänen sind. Der Beweis für die Additivität der Kullback-Leibler-Divergenz mit den oben genannten Änderungen beruht auf folgenden Eigenschaften:

- Quotientenregel für den Logarithmus: $log_x \frac{y}{z} = log_x y - log_x z$ und
- Die Summe der Anzahlen einzelner Werte entspricht der Anzahl des korrespondierenden generalisierten Wertes (siehe Gleichung B.15): $\sum_{i=x}^{y} freq(o_i) = freq([x, y])$.

O. B. d. A. gilt für zwei Generalisierungsstufen:

$$\begin{aligned}
KLD(A||A') &= o_1 * log_2 \frac{o_1}{g_1} + o_2 * log_2 \frac{o_2}{g_1} + ... \\
&\quad + o_{n-1} * log_2 \frac{o_{n-1}}{g_m} + o_n * log_2 \frac{o_n}{g_m} \\
&= o_1 * log_2 o_1 - o_1 + log_2 g_1 + o_2 * log_2 o_2 - o_2 + log_2 g_1 + ... \\
&\quad + o_{n-1} * log_2 o_{n-1} - o_{n-1} + log_2 g_m + o_n * log_2 o_n - o_n + log_2 g_m \\
&= o_1 * log_2 o_1 + o_2 * log_2 o_2 + ... \\
&\quad + o_{n-1} * log_2 o_{n-1} + o_n * log_2 o_n \\
&\quad - o_1 * log_2 g_1 - ... - o_n * log_2 g_m
\end{aligned} \tag{B.23}$$

$$\begin{aligned}
KLD(A'||A'') &= o_1 * log_2 \frac{g_1}{g'_1} + o_2 * log_2 \frac{g_2}{g'_1} + ... \\
&\quad + o_{m-1} * log_2 \frac{g_{m-1}}{g'_{m'}} + o_m * log_2 \frac{g_m}{g'_{m'}} \\
&= o_1 * log_2 g_1 - o_1 * log_2 g'_1 + o_2 * log_2 g_2 - o_2 * log_2 g'_1 + ... \\
&\quad + o_{m-1} * log_2 g_{m-1} - o_{m-1} * log_2 g'_{m'} + o_m * log_2 g_m - o_m * log_2 g'_{m'} \\
&= o_1 * log_2 g_1 + o_2 * log_2 g_2 + ... \\
&\quad + o_{m-1} * log_2 g_{m-1} + o_m * log_2 g_m \\
&\quad - o_1 * log_2 g'_1 - ... - o_{m'} * log_2 g'_{m'}
\end{aligned} \tag{B.24}$$

Werden die beiden Divergenzen aufsummiert, ergibt sich Folgendes:

$$\begin{aligned} KLD(A||A') + KLD(A'||A'') &= o_1 * log_2 o_1 + ... + o_n * log_2 o_n \\ &\quad - o_1 * log_2 g_1 - ... - o_m * log_2 g_m \\ &\quad + o_1 * log_2 g_1 + ... + o_m * log_2 g_m \\ &\quad - o_1 * log_2 g'_1 - ... - o_{m'} * log_2 g'_{m'} \\ &= o_1 * log_2 o_1 + ... + o_n * log_2 o_n \\ &\quad - o_1 * log_2 g'_1 - ... - o_{m'} * log_2 g'_{m'} \end{aligned} \tag{B.25}$$

Betrachten wir zum Vergleich die direkte Berechnung der Kullback-Leibler-Divergenz auf den Originaldaten zur zweiten Generalisierungsstufe, erhalten wir

$$\begin{aligned} KLD(A||A'') &= o_1 * log_2 \frac{o_1}{g'_1} + o_2 * log_2 \frac{o_2}{g'_1} + ... \\ &\quad + o_{n-1} * log_2 \frac{o_{n-1}}{g'_{m'}} + o_n * log_2 \frac{o_n}{g'_{m'}} \\ &\quad + o_1 * log_2 o_1 - o_1 * log_2 g'_1 + o_2 * log_2 o_2 - o_2 * log_2 g'_1 + ... \\ &\quad + o_{n-1} * log_2 o_{n-1} - o_{n-1} * log_2 g'_{m'} + o_n * log_2 o_n - o_n * log_2 g'_{m'} \\ &= o_1 * log_2 o_1 + o_2 * log_2 o_2 + ... \\ &\quad + o_{n-1} * log_2 o_{n-1} + ... + o_n * log_2 o_n \\ &\quad - o_1 * log_2 g'_1 - ... - o_{m'} * log_2 g'_{m'}. \end{aligned} \tag{B.26}$$

Wie leicht zu erkennen ist, sind die Divergenzen für beide Generalisierungen gleich. Dies lässt sich auf folgende, allgemeinere Rechenregel für Logarithmen zurückführen:

$$\begin{aligned} log_x(\frac{a}{b}) + log_x(\frac{b}{c}) &= log_x(a) - log_x(b) + log_x(b) - log_x(c) \\ &= log_x(a) - log_x(c) \\ &= log_x(\frac{a}{c}), \end{aligned} \tag{B.27}$$

bzw. für den allgemeinen Fall:

$$\begin{aligned} log_x(\frac{n_1}{n_2}) + log_x(\frac{n_2}{n_3}) + \cdots + log_x(\frac{n_{m-1}}{n_m}) &= log_x(n_1) - log_x(n_2) + log_x(n_2) - log_x(n_3) \\ &\quad + \cdots + log_x(n_{m-1}) - log_x(n_m) \\ &= log_x(n_1) - log_x(n_m) \\ &= log_x(\frac{n_1}{n_m}). \end{aligned} \tag{B.28}$$

□

Beispiel:

Für das laufende Beispiel sind die Umformungen der Kullback-Leibler-Divergenzen in B.29 bzw. B.30 gegeben. Von Interesse sind jeweils die Terme nach dem letzten Gleichheitszeichen:

$$
\begin{aligned}
KLD(A||A') &= 0,1 * log_2\frac{0,1}{0,3} + 0,2 * log_2\frac{0,2}{0,3} \\
&+ 0,3 * log_2\frac{0,3}{0,7} + 0,4 * log_2\frac{0,4}{0,7} \\
&= 0,1 * log_2 0,1 - 0,1 + log_2 0,3 + 0,2 * log_2 0,2 - 0,2 + log_2 0,3 \\
&+ 0,3 * log_2 0,3 - 0,3 + log_2 0,7 + 0,4 * log_2 0,4 - 0,4 + log_2 0,7 \\
&= 0,1 * log_2 0,1 + 0,2 * log_2 0,2 + 0,3 * log_2 0,3 + 0,4 * log_2 0,4 \\
&- 0,3 * log_2 0,3 - 0,7 * log_2 0,7
\end{aligned}
\tag{B.29}
$$

$$
\begin{aligned}
KLD(A'||A'') &= 0,3 * log_2\frac{0,3}{1,0} + 0,7 * log_2\frac{0,7}{1,0} \\
&= 0,3 * log_2 0,3 - 0,3 * log_2 1,0 + 0,7 * log_2 0,7 - 0,7 * log_2 1,0 \\
&= 0,3 * log_2 0,3 + 0,7 * log_2 0,7 - 1,0 * log_2 1,0 \\
&= 0,3 * log_2 0,3 + 0,7 * log_2 0,7
\end{aligned}
\tag{B.30}
$$

Werden beide Divergenzen aufsummiert, so kürzt sich der Term $0,3 * log_2 0,3 - 0,7 * log_2 0,7$ heraus. Übrig bleiben lediglich die Terme für der Originalwerte und der verbleibende Informationsgehalt für die neue Generalisierungsstufe — in diesem Fall der Wert 0, da der Logarithmus von 1 zu einer beliebigen Basis immer 0 ergibt.

$$
\begin{aligned}
KLD(A||A') + KLD(A'||A'') &= 0,1 * log_2 0,1 + 0,2 * log_2 0,2 + 0,3 * log_2 0,3 + 0,4 * log_2 0,4 \\
&- 0,3 * log_2 0,3 - 0,7 * log_2 0,7 \\
&+ 0,3 * log_2 0,3 + 0,7 * log_2 0,7 \\
&= 0,1 * log_2 0,1 + 0,2 * log_2 0,2 + 0,3 * log_2 0,3 + 0,4 * log_2 0,4
\end{aligned}
\tag{B.31}
$$

Zum Vergleich zeigen wir abschließend noch die Berechnung der Kullback-Leibler-Divergenz für die direkte Anonymisierung von A zu A'':

$$
\begin{aligned}
KLD(A||A'') &= 0,1 * log_2\frac{0,1}{1,0} + 0,2 * log_2\frac{0,2}{1,0} \\
&+ 0,3 * log_2\frac{0,3}{1,0} + 0,4 * log_2\frac{0,4}{1,0} \\
&= 0,1 * log_2 0,1 - 0,1 + log_2 1,0 + 0,2 * log_2 0,2 - 0,2 + log_2 1,0 \\
&+ 0,3 * log_2 0,3 - 0,3 + log_2 0,7 + 0,4 * log_2 0,4 - 0,4 + log_2 1,0 \\
&= 0,1 * log_2 0,1 + 0,2 * log_2 0,2 + 0,3 * log_2 0,3 + 0,4 * log_2 0,4 \\
&- 1,0 * log_2 1,0 \\
&= 0,1 * log_2 0,1 + 0,2 * log_2 0,2 + 0,3 * log_2 0,3 + 0,4 * log_2 0,4
\end{aligned}
\tag{B.32}
$$

Anwendung der Kullback-Leibler-Divergenz für einen iterativen Anonymisierungsalgorithmus

Die vorgestellten Anpassungen zur Berechnung der Kullback-Leiber-Divergenz erfolgten nicht ohne Grund: Anonymisierungsalgorithmen, die auf k-Anonymität und verwandten Konzepten aufbauen, versuchen, einen möglichst geringen Informationsverlust bei der Generalisierung von Daten zu erreichen. Dabei werden die Rohdaten

gemäß der Domänengeneralisierungshierarchie schrittweise auf den nächsten, allgemeineren Wert abgebildet bis ein vorher festgelegtes Maß an Anonymität erreicht ist. Zwar lässt sich der Informationsverlust für jede Stufe erneut aus den Rohdaten errechnen, doch dies ist aufgrund der hohen Datenmenge nicht sehr effizient. Durch eine iterative Berechnung, wie sie durch die vorgestellten Anpassungen möglich ist, wird die Einbindung direkt in die entsprechenden Anonymisierungsalgorithmen ermöglicht.

Pseudocode 10 skizziert einen Algorithmus, der die Anonymisierung für ein einzelnes Attribut A auf einer Datenbankrelation R ausführt. Als Eingabe erhält der Algorithmus zudem die Domänengeneralisierungshierarchie DGH von A und das gewünschte Maß an minimaler Anonymität s und den maximal erlaubten Informationsverlust t.

Algorithmus 10: Iterative Anonymisierung

Data: Datenbankrelation R, Attribute A aus R, Domänengeneralisierungshierarchie DGH für A, Grenzwert s und t für Anonymisierungsgrad bzw. maximalen Informationsverlust
Result: Eine ausreichende Generalisierung A' bzw. A''
s' = 0;
t' = 0;
A" = A;
while $s' < s \wedge t' < t$ **do**
  $A' = A''$;
  $A'' = getNextGeneralization(A', DGH)$;
  $s' = getAnonymity(D)$;
  $t' = t' + calculateKLD(A', A'')$;
end
if t' < t **then**
  //Fall 1: Anonymisierung ausreichend
  return A'';
else
  //Fall 2: Informationsverlust nach letztem Schritt zu hoch
  return A';
end

Der Algorithmus verwaltet intern zwei Generalisierungen von A: A', und A'', die beide zunächst mit den Rohdaten initialisiert werden. In jedem Generalisierungsschritt erfolgt ein Schleifendurchlauf, der prüft, ob die Daten ausreichend anonymisiert sind und ob der maximale Informationsverlust bereits überschritten wurde. Innerhalb der Schleife erfolgen folgende vier Anweisungen:

1. Die geringere Anonymisierung A' wird durch die stärkere Anonymisierung A'' ersetzt.
2. A'' wird um eine Stufe weiter generalisiert.
3. Das Maß der Anonymität wird neu berechnet.
4. Der zusätzliche Informationsverlust wird hinzugefügt.

Das Verlassen der Schleife kann zwei Ursachen haben: Entweder ist das gewünschte Maß an Anonymität erreicht (Fall 1) und die zuletzt generalisierten Daten werden zurückgegeben, oder der maximale Informationsverlust wurde überschritten (Fall 2). Im zweiten Fall wird die vorherige, wenn auch nicht ausreichende, Anonymisierung zurückgegeben.

Der Algorithmus kann auch leicht auf mehrdimensionale Verfahren angewendet werden, indem mehrere Attribute übergeben werden und der Informationsverlust KLD_R für die gesamte Relation R berechnet wird. Die Auswahl des oder der zu anonymisierenden Attribute hängt vom jeweils angewendeten Anonymisierungsverfahren ab.

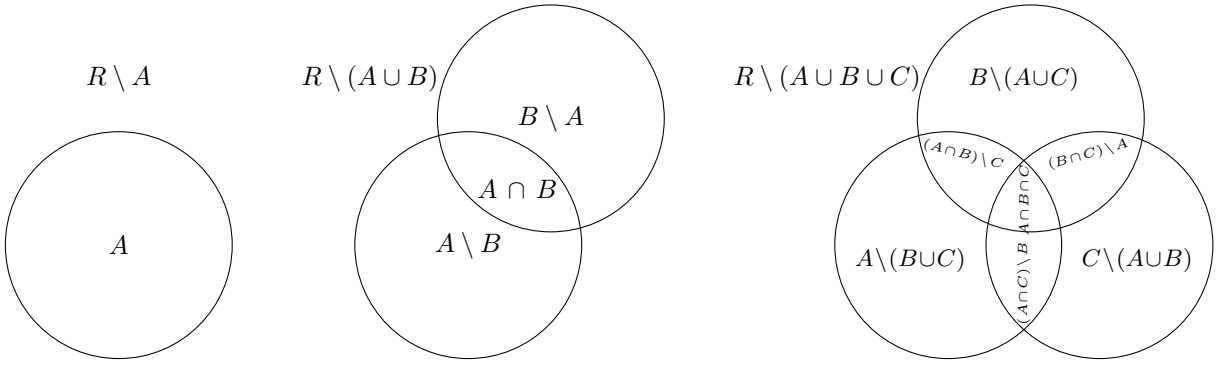


Abbildung B.3: Entwicklung der möglichen Ergebnisrelationen über eine Folge von mehreren Anfragen.

Durch die Integration der Berechnung der modifizierten Kullback-Leibler-Divergenz in die jeweiligen Anonymisierungsalgorithmen ergeben sich Effizienzvorteile. Es muss lediglich ein einzelner initialer Full-Table-Scan durchgeführt werden, um die Verteilung der Rohdaten für den ersten Generalisierungsschritt $KLD(A||A')$ und für den Normalisierungsfaktor $H(A)$ zu berechnen. Alle folgenden Berechnungen der Kullback-Leibler-Divergenz können anschließend verkürzt werden, da die Hälfte der Berechnungen (Logarithmus des Zählers) bereits durchgeführt wurden. Es verbleibt lediglich der Nenner des Logarithmus, dessen Ausgangsdaten aber lokal vom Algorithmus berechnet werden. Dadurch entfallen weitere, kostenintensive Full-Table-Scans.

B.4 Mengenorientierter Ansatz

Bei der Anonymisierung von Daten wird in vielen Metriken, wie der k-Anonymität, überprüft, ob pro Äquivalenzklasse eine Mindestanzahl an Tupeln vorhanden ist. Voraussetzung dafür ist, trivialerweise, dass der gesamte Datenbestand mindestens so viele Tupel enthält, wie pro Äquivalenzklasse enthalten sein müssen.

Für das Ergebnis A einer Anfrage Q_1 über einer Relation $r(R)$ ist dies einfach nachvollziehbar, da sich die Tupel einfach abzählen lassen. Für Folgen von Anfragen, bei denen der Anfragende manuell die Schnittmengen der Teilergebnisse bilden kann, ist dies jedoch problematisch.

Stellt der Anfragende eine zweite Anfrage Q_2, erhält er als Ergebnis die Menge B. Zudem kann er, sofern die beiden Ergebnismengen vereinigungskompatibel sind, den Schnitt $A \cap B$ bilden, wodurch er erfährt, welche Tupel die Selektionsprädikate beider Anfragen erfüllen. Außerdem gelangt er Kenntnis darüber, welche Tupel in A, aber nicht in B vorkommen und umgekehrt ($A \setminus B$ bzw. $B \setminus A$). Für jede dieser drei Ergebnismengen muss sichergestellt sein, dass sie eine Mindestanzahl an Tupeln enthalten, da ansonsten die Gütemaße der Anonymisierungsmetriken nicht erreicht werden können. Die weitere Entwicklung der entstehenden Teilmengen ist in Abbildung B.3 zu sehen.

Wird eine Datenbank bzw. eine einzelne Relation über einen längeren Zeitraum angefragt, steigt die Anzahl der abgesetzten Anfragen stetig an. Die Anzahl der zu überprüfenden Teilmengen steigt dabei exponentiell an: Wurden bereits n Anfragen gestellt, so ergeben sich 2^n–1 zu überprüfende Teilmengen[6]. Dies führt zu zwei Problemen: Einerseits sinkt die Performance, da ggf. jede der Teilmengen anonymisiert werden muss. Zudem steigt das Risiko, dass die Anfragen abgelehnt werden, da die Mindestzahl an Tupeln für jede Teilmenge nicht mehr erreicht werden kann.

Als Lösung wurde in PArADISE ein gleitendes Fenster für eingehende Anfragen entwickelt. Dieser Ansatz wurde erstmals in [Gru14] vorgestellt. Algorithmus 11 zeigt den groben Ablauf dieses Konzeptes. Im Kern wird die Anzahl der zu betrachtenden Anfragen pro Anfragenden bzw. Verwendungszweck auf einen zuvor festgesetzten

[6]Das exponentielle Wachstum ergibt sich aus der Zweiteilung jeder bekannten Teilmenge in zwei neue Teilmengen: den Teil, der auch die neue Anfrage erfüllt, und den Teil, der dies nicht tut.

Wert n limitiert. Für die letzten n Anfragen werden die entsprechenden Teilmengen, wie oben beschrieben, gebildet und die Gütekriterien der Anonymisierungsmetriken geprüft. Wird der Grenzwert n überschritten, wird die älteste Anfrage aus dem Fenster gelöscht und durch die neue Anfrage ersetzt.

Algorithmus 11: Sicherstellung einer Mindestanzahl an Tupeln pro Schnittmenge über mehrere Anfragen

```
Data: Menge von Anfrage Q, Fenster W der Größe n über Anfragen, Grenzwert k
Result: Pro Anfrage eine anonymisierte Anfrage q' oder NULL
B = ∅;
foreach q ∈ Q do
    if |W| < n then
        W.add(q);
    end
    else
        q_old = W.take();
        B.remove(q_old);
    end
    if |B| ≠ 0 then
        foreach b ∈ B do
            B.add(b ∧ q);
            B.add(b ∧ ¬(q));
        end
    end
    B.add(q ⋀(¬(b ∈ B));
    foreach b ∈ B do
        if count(b) < k then
            return NULL;
        end
    end
    return q' = anonymized_result(q);
end
```

Verwandte Verfahren

In [Döl16, FBD14] wird ein Verfahren vorgestellt, welches die Zuordnung der Identifikatoren zu den möglichen sensitiven Attribute durch bipartite Graphen realisiert. Auf den Graphen werden Approximationsalgorithmen angewendet, um mögliche eindeutige Identifizierungen von einzelnen Tupeln respektive Personen zu erkennen. Für jede Anfrage wird ein bipartiter Graph aufgestellt und die möglichen Zuordnungen mit denen vorheriger Anfragen via Überdeckungen verglichen. Tritt eine Verletzung der Anonymität auf, wird entweder die Beantwortung der Anfrage verweigert oder zusätzliche, künstliche Tupel zurückgegeben.

The Tracker [DDS79] ist ein statistisches Verfahren, welches durch eine minimale Folge von Aggregatanfragen mit Selektionen, wie Summen oder das Zählen von Werten, auf die Werte von einzelnen Tupeln schließen kann. Das Verfahren beruht auf der geschickten Auswahl von Bereichsanfragen, sodass die Anzahl der zurückgegebenen Werte sich in der Differenz zwischen zwei Anfragen nur um den Wert 1 unterscheidet. Durch die gewählten Selektionsprädikate lassen sich anschließend die Attributwerte des fehlenden Tupels bestimmen. Der von Denning und Denning vorgeschlagene *General Tracker* erlaubt die Identifizierung eines beliebigen Tupels in der Datenbank mit einer Komplexität von $O(n)$, wobei n die Anzahl der Attribute in der Datenbankrelation ist. In realen Anwendungsfällen sollen die Selektionsprädikate zur Definition des Trackers durch einfaches Raten leicht zu finden

sein, da durch die Gleichverteilung der Daten schnell identifizierende Merkmale gefunden werden können. Dies entspricht auch den Ergebnissen der in dieser Arbeit durchgeführten Quasi-Identifikator-Analyse in Kapitel 4.

In [DJL79] wird untersucht, wie viele Anfragen auf aggregierten Werten zur Ermittlung eines einzelnen Tupels benötigt werden. Dobkin, Jones und Lipton kommen zu dem Ergebnis, dass $1 + \frac{(m-1-p)}{k}$ Anfragen benötigt werden, wobei m die maximale Anzahl der Tupel ist, welche in die Aggregatfunktion einfließen; die Parameter p und k stehen für die Anzahl der bereits bekannten Tupel bzw. die maximale Größe der Überlappung zweier Anfragen. Da eine zu große Anzahl an Anfragen unweigerlich zur Ablehnung aller Anfragen ab einen bestimmten Punkt zur Folge hat, wird in [Ull82] vorgeschlagen, die Anzahl der betrachteten Anfragen zu begrenzen bzw. statt einzelnen Tupeln ganze Partitionen von Daten zu überwachen.

Die drei vorgestellten Verfahren eignen sich als performantere Alternative zum eigenen Ansatz, bevorzugen jedoch die Ablehnung weiterer Anfragen anstatt der Anonymisierung der Daten. Eine Implementierung in PArADISE ist allerdings bei Abschluss dieser Arbeit noch nicht erfolgt.

Anhang C

Regelverzeichnis

In diesem Anhangskapitel werden die zum Zeitpunkt der Einreichung konzipierten und implementierten Regeln zur Anfragetransformation im Detail aufgelistet. Zu jeder Regel werden die formale Beschreibung sowie die für die Anwendung der Regel notwendigen Invarianten, Vorbedingungen und Nachbedingungen angegeben. Visualisierungen der Anfragetransformationen sowie Beispielanfragen für Original- und transformierte Anfrage runden die Präsentation ab.

Das Zeichen $\square$ steht für eine nicht vorhandene Operation. Dies dient zur Veranschaulichung dafür, dass ein Operator an einer Stelle vor bzw. nach der Transformation nicht vorhanden war bzw. ist.

C.1 Linear-arithmetische Vergleiche, Klasse 1+2

Im Folgenden werden die Anfragetransformationen für die einfachen linear-arithmetischen Vergleiche der Klassen 1 (*LAC1*) und 2 (*LAC2*) nach [Tür99] vorgestellt. Insgesamt besteht diese Regelmenge aus acht verschiedenen Regeln: Für die drei Vergleichsoperatoren $<$, $=$ (in zwei Varianten) und $>$ wird je eine Regel für den Vergleich einer Variablen mit einer Konstanten bzw. einer weiteren Variable eingeführt. Da für die Vergleichsoperatoren $\leq$, $\neq$ und $\geq$ bereits einfache Ersetzungen existieren[1], brauchen diese nicht gesondert betrachtet zu werden.

Anwendung finden diese Regeln z. B. bei der Ausführung von einfachen Anfragen mit Einbezug von FPGAs. Zwar kann ein einzelner FPGA, sofern entsprechend konfiguriert, jeden der sechs Vergleiche zeitgleich unterstützen, jedoch ist dies meist nicht der Fall. Neben den arithmetischen Vergleichen werden beispielsweise auch Aggregatfunktionen unterstützt. Nach dem Aufbauschema für FPGAs (siehe Abbildung 5.3) lassen sich maximal vier verschiedene Kombinationen von Vergleichs- und Aggregatsfunktionen unterstützen. Entsprechend muss beim Design des FPGAs – und folglich auch bei der Formulierung von Anfragen – eine Entscheidung über die am häufigsten genutzten Operatoren getroffen werden.

[1] Beispielsweise gilt $x \leq c \equiv x < c \vee x = c$.

L01: Kleiner zu Kleiner-gleich (Attribut-Konstante)

Regel: $\sigma_{x<c}(r) \sqsubseteq_K \sigma_{x\leq c}(r)$
Invarianten: —
Vorbedingungen:

- Untere Ebene L_i: $\{<_c\} \not\subseteq O_i$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: $\{\leq_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: $\{<_c\} \subseteq O_{i+1}$

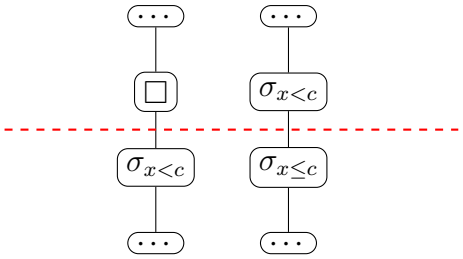


Abbildung C.1: Transformation L01

Originalanfrage:

```
SELECT aid, length
FROM tracks
WHERE length < 250000
```

Transformierte Anfrage:

```
WITH X AS (
  SELECT aid, length
  FROM tracks
  WHERE length <= 250000
)
SELECT aid, length
FROM X
WHERE length < 250000
```

L02: Gleich zu Kleiner-gleich (Attribut-Konstante)

Regel: $\sigma_{x=c}(r) \sqsubseteq_K \sigma_{x\leq c}(r)$
Invarianten: —
Vorbedingungen:

- Untere Ebene L_i: $\{=_c\} \not\subseteq O_i$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: $\{\leq_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: $\{=_c\} \subseteq O_{i+1}$

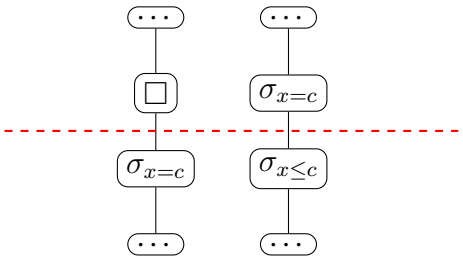


Abbildung C.2: Transformation L02

Originalanfrage:

```
SELECT aid, length
FROM tracks
WHERE length = 250000
```

Transformierte Anfrage:

```
WITH X AS (
  SELECT aid, length
  FROM tracks
  WHERE length <= 250000
)
SELECT aid, length
FROM X
WHERE length = 250000
```

L03: Gleich zu Größer-gleich (Attribut-Konstante)

Regel: $\sigma_{x=c}(r) \sqsubseteq_K \sigma_{x \geq c}(r)$
Invarianten: —
Vorbedingungen:

- Untere Ebene L_i: $\{=_c\} \not\subseteq O_i$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: $\{\geq_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: $\{=_c\} \subseteq O_{i+1}$

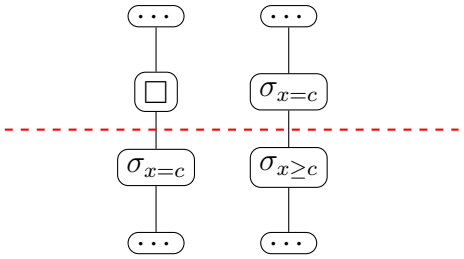


Abbildung C.3: Transformation L03

Originalanfrage:

```
SELECT aid, length
FROM tracks
WHERE length = 250000
```

Transformierte Anfrage:

```
WITH X AS (
  SELECT aid, length
  FROM tracks
  WHERE length >= 250000
)
SELECT aid, length
FROM X
WHERE length = 250000
```

L04: Größer zu Größer-gleich (Attribut-Konstante)

Regel: $\sigma_{x>c}(r) \sqsubseteq_K \sigma_{x \geq c}(r)$
Vorbedingungen:

- Untere Ebene L_i: $\{>_c\} \not\subseteq O_i$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: $\{\geq_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: $\{>_c\} \subseteq O_{i+1}$

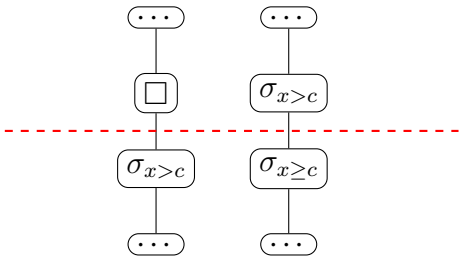


Abbildung C.4: Transformation L04

Originalanfrage:

```
SELECT aid, length
FROM tracks
WHERE length > 250000
```

Transformierte Anfrage:

```
WITH X AS (
  SELECT aid, length
  FROM tracks
  WHERE length >= 250000
)
SELECT aid, length
FROM X
WHERE length > 250000
```

L05: Kleiner zu Kleiner-gleich (Attribut-Attribut)

Regel: $\sigma_{x<y}(r) \sqsubseteq_K \sigma_{x\leq y}(r)$
Invarianten: —
Vorbedingungen:

- Untere Ebene L_i: $\{<_a\} \not\subseteq O_i$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: $\{\leq_a\} \subseteq O_i$
- Obere Ebene L_{i+1}: $\{<_a\} \subseteq O_{i+1}$

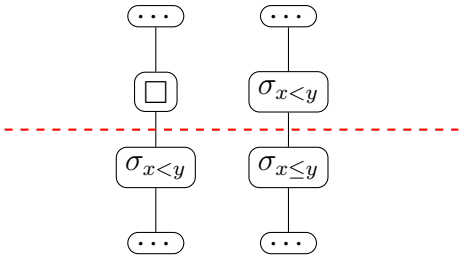


Abbildung C.5: Transformation L05

Originalanfrage:

```
SELECT aid, gid, bitrate
FROM tracks
WHERE gid < bitrate
```

Transformierte Anfrage:

```
WITH X AS (
  SELECT aid, gid, bitrate
  FROM tracks
  WHERE gid <= bitrate
)
SELECT aid, gid, bitrate
FROM X
WHERE gid < bitrate
```

L06: Gleich zu Kleiner-gleich (Attribut-Attribut)

Regel: $\sigma_{x=y}(r) \sqsubseteq_K \sigma_{x\leq y}(r)$
Invarianten: —
Vorbedingungen:

- Untere Ebene L_i: $\{=_a\} \not\subseteq O_i$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: $\{\leq_a\} \subseteq O_i$
- Obere Ebene L_{i+1}: $\{=_a\} \subseteq O_{i+1}$

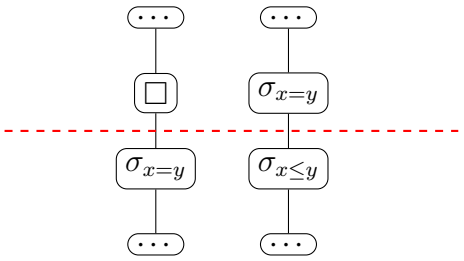


Abbildung C.6: Transformation L06

Originalanfrage:

```
SELECT aid, gid, bitrate
FROM tracks
WHERE gid = bitrate
```

Transformierte Anfrage:

```
WITH X AS (
  SELECT aid, gid, bitrate
  FROM tracks
  WHERE gid <= bitrate
)
SELECT aid, gid, bitrate
FROM X
WHERE gid = bitrate
```

L07: Gleich zu Größer-gleich (Attribut-Attribut)

Regel: $\sigma_{x=y}(r) \sqsubseteq_K \sigma_{x \geq y}(r)$
Invarianten: —
Vorbedingungen:

- Untere Ebene L_i: $\{=_a\} \not\subseteq O_i$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: $\{\geq_a\} \subseteq O_i$
- Obere Ebene L_{i+1}: $\{=_a\} \subseteq O_{i+1}$

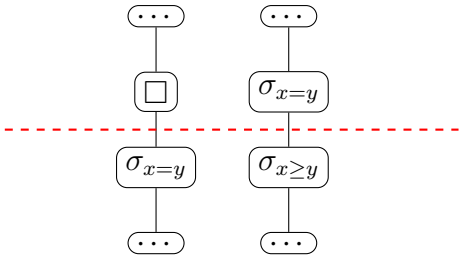


Abbildung C.7: Transformation L07

Originalanfrage:

```
SELECT aid, gid, bitrate
FROM tracks
WHERE gid = bitrate
```

Transformierte Anfrage:

```
WITH X AS (
  SELECT aid, gid, bitrate
  FROM tracks
  WHERE gid >= bitrate
)
SELECT aid, gid, bitrate
FROM X
WHERE gid = bitrate
```

L08: Größer zu Größer-gleich (Attribut-Attribut)

Regel: $\sigma_{x>y}(r) \sqsubseteq_K \sigma_{x \geq y}(r)$
Invarianten: —
Vorbedingungen:

- Untere Ebene L_i: $\{>_a\} \not\subseteq O_i$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: $\{\geq_a\} \subseteq O_i$
- Obere Ebene L_{i+1}: $\{>_a\} \subseteq O_{i+1}$

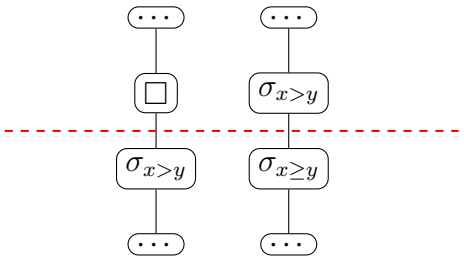


Abbildung C.8: Transformation L08

Originalanfrage:

```
SELECT aid, gid, bitrate
FROM tracks
WHERE gid > bitrate
```

Transformierte Anfrage:

```
WITH X AS (
  SELECT aid, gid, bitrate
  FROM tracks
  WHERE gid >= bitrate
)
SELECT aid, gid, bitrate
FROM X
WHERE gid > bitrate
```

C.2 Aggregatanfragen mit Query Rewriting by Contract

Im folgenden Abschnitt werden auf Basis der Ergebnisse von Can Türker in [Tür99] Kontrakt-basierte Anfragetransformationen abgeleitet. Diese lassen sich in drei Unterklassen einteilen:

- Ersetzung einer Aggregatfunktion durch eine andere Aggregatfunktion (Regel A01 bis A26),
- Ersetzung eines Allquantors durch eine Aggregatfunktion (Regel A27 bis A45) und
- Ersetzung einer Aggregatfunktion durch einen Allquantor (Regeln A46 bis A51).

Die Regeln eignen sich wie bereits die Transformationsregeln der Klasse L für extrem ressourcenbeschränkte Umgebungen, die beispielsweise FPGAs nutzen. Diese können zum Beispiel so konfiguriert sein, dass nur die am häufigsten genutzten Kombinationen aus Aggregatfunktion und Vergleichsoperator unterstützt werden. Um keine vollständige Neukonfiguration bzw. einen Neuaufbau des FPGAs vorzunehmen, kann stattdessen die Anfrage mit den unten aufgeführten Regeln in eine andere Menge von Operatoren transformiert werden.

A01: Minimum-gleich zu Maximum-größer-gleich (Attribut-Konstante)

Regel: $\sigma_{min(X)=c}(r) \sqsubseteq_K \sigma_{max(X)\geq c}(r)$
Invarianten: —
Vorbedingungen:

- Untere Ebene L_i: $\{min, =_c\} \not\subseteq O_i$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: $\{max, \geq_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: $\{min, =_c\} \not\subseteq O_{i+1}$

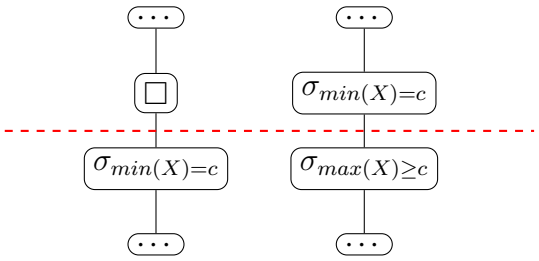


Abbildung C.9: Transformation A01

Originalanfrage:

```
SELECT aid
FROM tracks
GROUP BY aid
HAVING min(length) = 250000
```

Transformierte Anfrage:

```
SELECT aid
FROM (
  SELECT aid, length
  FROM tracks
  WHERE aid IN (
    SELECT aid
    FROM tracks
    GROUP BY aid
    HAVING max(length) >= 250000
  )
) AS X
GROUP BY aid
HAVING min(length) = 250000
```

A02: Minimum-größer-gleich zu Maximum-größer-gleich (Attribut-Konstante)

Regel: $\sigma_{min(X)\geq c}(r) \sqsubseteq_K \sigma_{max(X)\geq c}(r)$
Invarianten: —
Vorbedingungen:

- Untere Ebene L_i: $\{min\} \not\subseteq O_i$, $\{\geq_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: $\{max, \geq_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: $\{min, \geq_c\} \subseteq O_{i+1}$

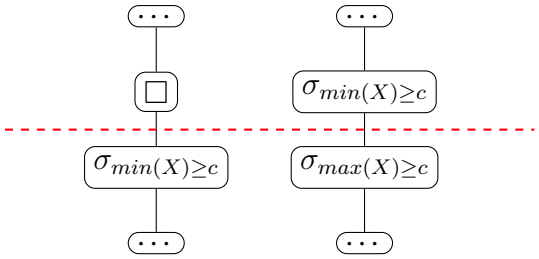


Abbildung C.10: Transformation A02

Originalanfrage:

```
SELECT aid
FROM tracks
GROUP BY aid
HAVING min(length) >= 250000
```

Transformierte Anfrage:

```
SELECT aid
FROM (
  SELECT aid, length
  FROM tracks
  WHERE aid IN (
    SELECT aid
    FROM tracks
    GROUP BY aid
    HAVING max(length) >= 250000
  )
) AS X
GROUP BY aid
HAVING min(length) >= 250000
```

A03: Minimum-größer zu Maximum-größer (Attribut-Konstante)

Regel: $\sigma_{min(X)>c}(r) \sqsubseteq_K \sigma_{max(X)>c}(r)$
Invarianten: —
Vorbedingungen:

- Untere Ebene L_i: $\{min\} \not\subseteq O_i, \{>_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: $\{max, >_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: $\{min, >_c\} \subseteq O_{i+1}$

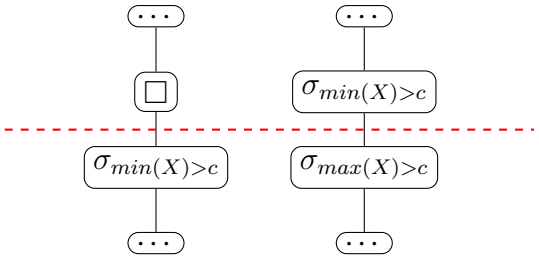


Abbildung C.11: Transformation A03

Originalanfrage:

```
SELECT aid
FROM tracks
GROUP BY aid
HAVING min(length) > 250000
```

Transformierte Anfrage:

```
SELECT aid
FROM (
  SELECT aid, length
  FROM tracks
  WHERE aid IN (
    SELECT aid
    FROM tracks
    GROUP BY aid
    HAVING max(length) > 250000
  )
) AS X
GROUP BY aid
HAVING min(length) > 250000
```

A04: Maximum-gleich zu Minimum-kleiner-gleich (Attribut-Konstante)

Regel: $\sigma_{max(X)=c}(r) \sqsubseteq_K \sigma_{min(X)\leq c}(r)$
Invarianten: —
Vorbedingungen:

- Untere Ebene L_i: $\{max, =_c\} \not\subseteq O_i$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: $\{min, \leq_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: $\{max, =_c\} \subseteq O_{i+1}$

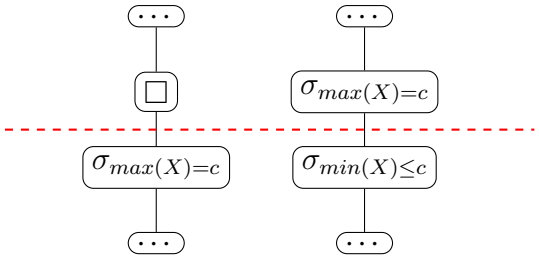


Abbildung C.12: Transformation A04

Originalanfrage:

```
SELECT aid
FROM tracks
GROUP BY aid
HAVING max(length) = 250000
```

Transformierte Anfrage:

```
SELECT aid
FROM (
  SELECT aid, length
  FROM tracks
  WHERE aid IN (
    SELECT aid
    FROM tracks
    GROUP BY aid
    HAVING min(length) <= 250000
  )
) AS X
GROUP BY aid
HAVING max(length) = 250000
```

A05: Maximum-kleiner-gleich zu Minimum-kleiner-gleich (Attribut-Konstante)

Regel: $\sigma_{max(X)\leq c}(r) \sqsubseteq_K \sigma_{min(X)\leq c}(r)$
Invarianten: —
Vorbedingungen:

- Untere Ebene L_i: $\{max\} \notin O_i, \{\leq_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: $\{min, \leq_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: $\{max, \leq_c\} \subseteq O_{i+1}$

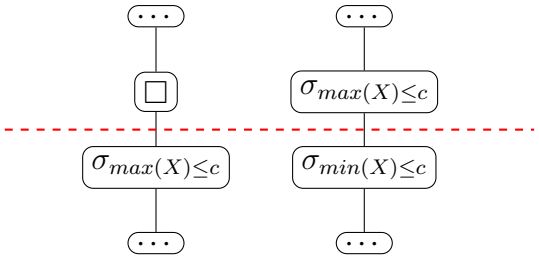


Abbildung C.13: Transformation A05

Originalanfrage:

```
SELECT aid
FROM tracks
GROUP BY aid
HAVING max(length) <= 250000
```

Transformierte Anfrage:

```
SELECT aid
FROM (
  SELECT aid, length
  FROM tracks
  WHERE aid IN (
    SELECT aid
    FROM tracks
    GROUP BY aid
    HAVING min(length) <= 250000
  )
) AS X
GROUP BY aid
HAVING max(length) <= 250000
```

A06: Maximum-kleiner zu Minimum-kleiner (Attribut-Konstante)

Regel: $\sigma_{max(X)<c}(r) \sqsubseteq_K \sigma_{min(X)<c}(r)$
Invarianten: —
Vorbedingungen:

- Untere Ebene L_i: $\{max\} \not\subseteq O_i, \{<_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: $\{min, <_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: $\{max, <_c\} \subseteq O_{i+1}$

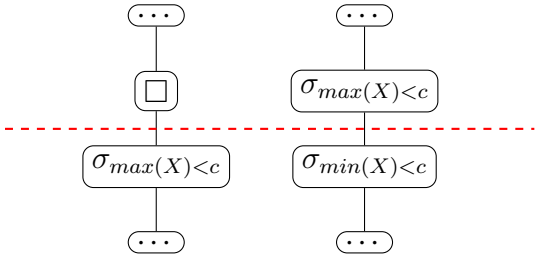


Abbildung C.14: Transformation A06

Originalanfrage:

```
SELECT aid
FROM tracks
GROUP BY aid
HAVING max(length) < 250000
```

Transformierte Anfrage:

```
SELECT aid
FROM (
  SELECT aid, length
  FROM tracks
  WHERE aid IN (
    SELECT aid
    FROM tracks
    GROUP BY aid
    HAVING min(length) < 250000
  )
) AS X
GROUP BY aid
HAVING max(length) < 250000
```

A07: Minimum-gleich zu Durchschnitt-größer-gleich (Attribut-Konstante)

Regel: $\sigma_{min(X)=c}(r) \sqsubseteq_K \sigma_{avg(X)\geq c}(r)$
Invarianten: —
Vorbedingungen:

- Untere Ebene L_i: $\{min, =_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: $\{avg, \geq_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: $\{min, =_c\} \subseteq O_{i+1}$

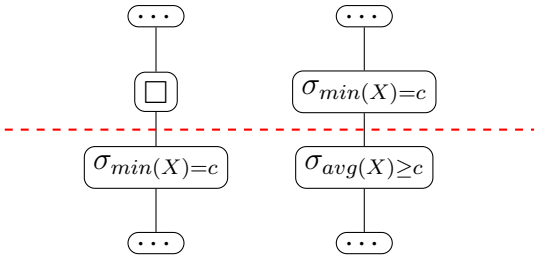


Abbildung C.15: Transformation A07

Originalanfrage:

```
SELECT aid
FROM tracks
GROUP BY aid
HAVING min(length) = 250000
```

Transformierte Anfrage:

```
SELECT aid
FROM (
  SELECT aid, length
  FROM tracks
  WHERE aid IN (
    SELECT aid
    FROM tracks
    GROUP BY aid
    HAVING avg(length) >= 250000
  )
) AS X
GROUP BY aid
HAVING min(length) = 250000
```

A08: Minimum-größer-gleich zu Durchschnitt-größer-gleich (Attribut-Konstante)

Regel: $\sigma_{min(X)\geq c}(r) \sqsubseteq_K \sigma_{avg(X)\geq c}(r)$
Invarianten: —
Vorbedingungen:

- Untere Ebene L_i: $\{min\} \not\subseteq O_i, \{\geq_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: $\{avg, \geq_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: $\{min, \geq_c\} \subseteq O_{i+1}$

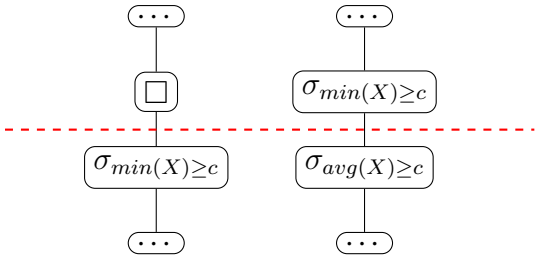


Abbildung C.16: Transformation A08

Originalanfrage:

```
SELECT aid
FROM tracks
GROUP BY aid
HAVING min(length) >= 250000
```

Transformierte Anfrage:

```
SELECT aid
FROM (
  SELECT aid, length
  FROM tracks
  WHERE aid IN (
    SELECT aid
    FROM tracks
    GROUP BY aid
    HAVING avg(length) >= 250000
  )
) AS X
GROUP BY aid
HAVING min(length) >= 250000
```

A09: Minimum-größer zu Durchschnitt-größer (Attribut-Konstante)

Regel: $\sigma_{min(X)>c}(r) \sqsubseteq_K \sigma_{avg(X)>c}(r)$
Invarianten: —
Vorbedingungen:

- Untere Ebene L_i: $\{min\} \not\subseteq O_i, \{>_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: $\{avg, >_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: $\{min, >_c\} \subseteq O_{i+1}$

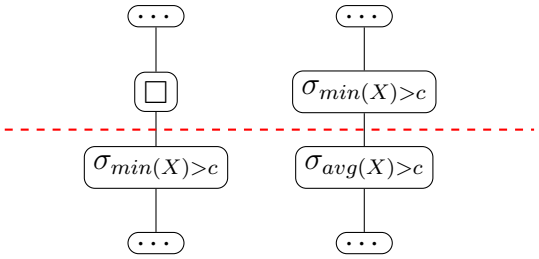


Abbildung C.17: Transformation A09

Originalanfrage:

```
SELECT aid
FROM tracks
GROUP BY aid
HAVING min(length) > 250000
```

Transformierte Anfrage:

```
SELECT aid
FROM (
  SELECT aid, length
  FROM tracks
  WHERE aid IN (
    SELECT aid
    FROM tracks
    GROUP BY aid
    HAVING avg(length) > 250000
  )
) AS X
GROUP BY aid
HAVING min(length) > 250000
```

A10: Maximum-gleich zu Durchschnitt-kleiner-gleich (Attribut-Konstante)

Regel: $\sigma_{max(X)=c}(r) \sqsubseteq_K \sigma_{avg(X)\leq c}(r)$
Invarianten: —
Vorbedingungen:

- Untere Ebene L_i: $\{max, =_c\} \not\subseteq O_i$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: $\{avg, \leq_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: $\{max, =_c\} \subseteq O_{i+1}$

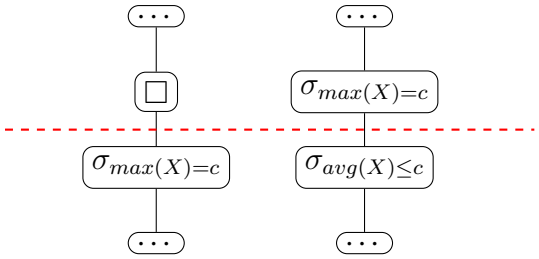


Abbildung C.18: Transformation A10

Originalanfrage:

```
SELECT aid
FROM tracks
GROUP BY aid
HAVING max(length) = 250000
```

Transformierte Anfrage:

```
SELECT aid
FROM (
  SELECT aid, length
  FROM tracks
  WHERE aid IN (
    SELECT aid
    FROM tracks
    GROUP BY aid
    HAVING avg(length) <= 250000
  )
) AS X
GROUP BY aid
HAVING max(length) = 250000
```

A11: Maximum-kleiner-gleich zu Durchschnitt-kleiner-gleich (Attribut-Konstante)

Regel: $\sigma_{max(X)\leq c}(r) \sqsubseteq_K \sigma_{avg(X)\leq c}(r)$
Invarianten: —
Vorbedingungen:

- Untere Ebene L_i: $\{max\} \not\subseteq O_i, \{\leq_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: $\{avg, \leq_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: $\{max, \leq_c\} \subseteq O_{i+1}$

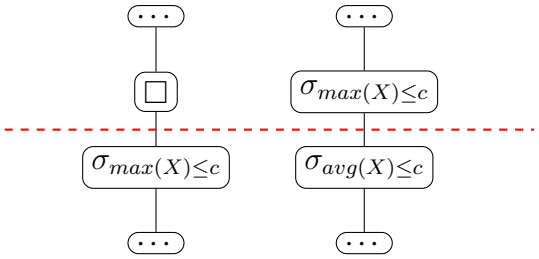


Abbildung C.19: Transformation A11

Originalanfrage:

```
SELECT aid
FROM tracks
GROUP BY aid
HAVING max(length) <= 250000
```

Transformierte Anfrage:

```
SELECT aid
FROM (
  SELECT aid, length
  FROM tracks
  WHERE aid IN (
    SELECT aid
    FROM tracks
    GROUP BY aid
    HAVING avg(length) <= 250000
  )
) AS X
GROUP BY aid
HAVING max(length) <= 250000
```

A12: Maximum-kleiner zu Durchschnitt-kleiner (Attribut-Konstante)

Regel: $\sigma_{max(X)<c}(r) \sqsubseteq_K \sigma_{avg(X)<c}(r)$
Invarianten: —
Vorbedingungen:

- Untere Ebene L_i: $\{max\} \not\subseteq O_i, \{<_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: $\{avg, <_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: $\{max, <_c\} \not\subseteq O_{i+1}$

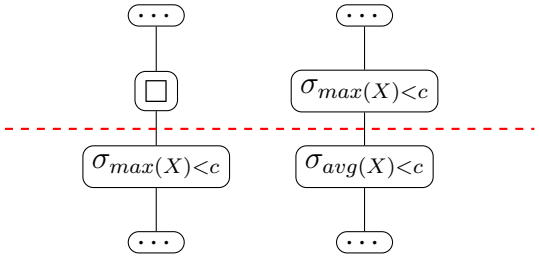


Abbildung C.20: Transformation A12

Originalanfrage:

```
SELECT aid
FROM tracks
GROUP BY aid
HAVING max(length) < 250000
```

Transformierte Anfrage:

```
SELECT aid
FROM (
  SELECT aid, length
  FROM tracks
  WHERE aid IN (
    SELECT aid
    FROM tracks
    GROUP BY aid
    HAVING avg(length) < 250000
  )
) AS X
GROUP BY aid
HAVING max(length) < 250000
```

A13: Minimum-gleich zu Summe-größer-gleich (Attribut-Konstante)

Regel: $\sigma_{min(X)=c}(r) \sqsubseteq_K \sigma_{sum(X)\geq c}(r)$
Invarianten: $c \geq 0$
Vorbedingungen:

- Untere Ebene L_i: $\{min, =_c\} \not\subseteq O_i$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: $\{sum, \geq_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: $\{min, =_c\} \subseteq O_{i+1}$

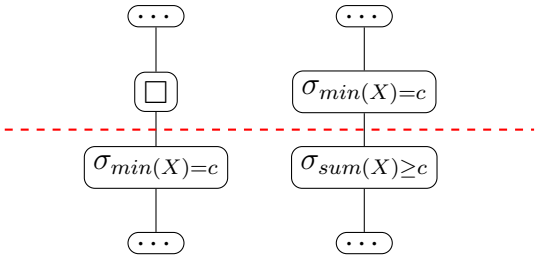


Abbildung C.21: Transformation A13

Originalanfrage:

```
SELECT aid
FROM tracks
GROUP BY aid
HAVING min(length) = 250000
```

Transformierte Anfrage:

```
SELECT aid
FROM (
  SELECT aid, length
  FROM tracks
  WHERE aid IN (
    SELECT aid
    FROM tracks
    GROUP BY aid
    HAVING sum(length) >= 250000
  )
) AS X
GROUP BY aid
HAVING min(length) = 250000
```

A14: Minimum-größer-gleich zu Summe-größer-gleich (Attribut-Konstante)

Regel: $\sigma_{min(X)\geq c}(r) \sqsubseteq_K \sigma_{sum(X)\geq c}(r)$
Invarianten: $c \geq 0$
Vorbedingungen:

- Untere Ebene L_i: $\{min\} \not\subseteq O_i$, $\{\geq_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: $\{sum, \geq_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: $\{min, \geq_c\} \subseteq O_{i+1}$

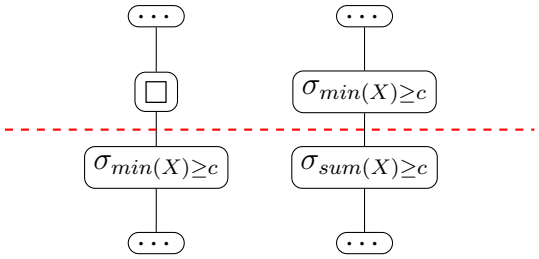


Abbildung C.22: Transformation A14

Originalanfrage:

```
SELECT aid
FROM tracks
GROUP BY aid
HAVING min(length) >= 250000
```

Transformierte Anfrage:

```
SELECT aid
FROM (
  SELECT aid, length
  FROM tracks
  WHERE aid IN (
    SELECT aid
    FROM tracks
    GROUP BY aid
    HAVING sum(length) >= 250000
  )
) AS X
GROUP BY aid
HAVING min(length) >= 250000
```

A15: Minimum-größer zu Summe-größer (Attribut-Konstante)

Regel: $\sigma_{min(X)>c}(r) \sqsubseteq_K \sigma_{sum(X)>c}(r)$
Invarianten: $c \geq 0$
Vorbedingungen:

- Untere Ebene L_i: $\{min\} \not\subseteq O_i, \{>_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: $\{sum, >_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: $\{min, >_c\} \subseteq O_{i+1}$

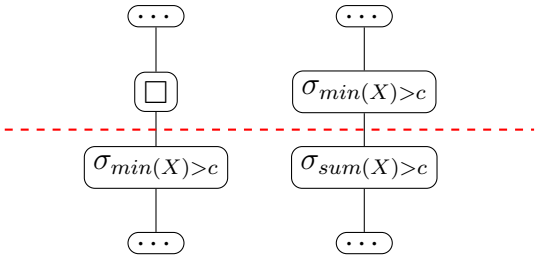


Abbildung C.23: Transformation A15

Originalanfrage:

```
SELECT aid
FROM tracks
GROUP BY aid
HAVING min(length) > 250000
```

Transformierte Anfrage:

```
SELECT aid
FROM (
  SELECT aid, length
  FROM tracks
  WHERE aid IN (
    SELECT aid
    FROM tracks
    GROUP BY aid
    HAVING sum(length) > 250000
  )
) AS X
GROUP BY aid
HAVING min(length) > 250000
```

A16: Maximum-gleich zu Summe-kleiner-gleich (Attribut-Konstante)

Regel: $\sigma_{max(X)=c}(r) \sqsubseteq_K \sigma_{sum(X)\leq c}(r)$
Invarianten: $c \leq 0$
Vorbedingungen:

- Untere Ebene L_i: $\{max, =_c\} \not\subseteq O_i$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: $\{sum, \leq_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: $\{max, =_c\} \subseteq O_{i+1}$

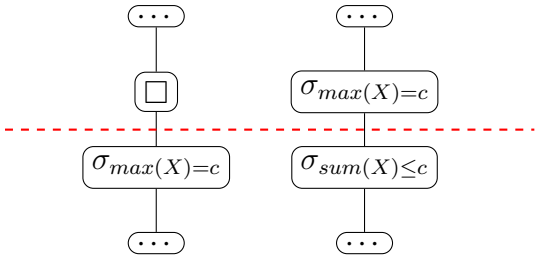


Abbildung C.24: Transformation A16

Originalanfrage:

```
SELECT aid
FROM tracks
GROUP BY aid
HAVING max(length*-1) = -250000
```

Transformierte Anfrage:

```
SELECT aid
FROM (
  SELECT aid, length*-1 AS L
  FROM tracks
  WHERE aid IN (
    SELECT aid
    FROM tracks
    GROUP BY aid
    HAVING sum(length*-1) <= -250000
  )
) AS X
GROUP BY aid
HAVING max(L) = -250000
```

A17: Maximum-kleiner-gleich zu Summe-kleiner-gleich (Attribut-Konstante)

Regel: $\sigma_{max(X)\leq c}(r) \sqsubseteq_K \sigma_{sum(X)\leq c}(r)$
Invarianten: $c \leq 0$
Vorbedingungen:

- Untere Ebene L_i: $\{max, \leq_c\} \not\subseteq O_i$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: $\{sum, \leq_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: $\{max, \leq_c\} \subseteq O_{i+1}$

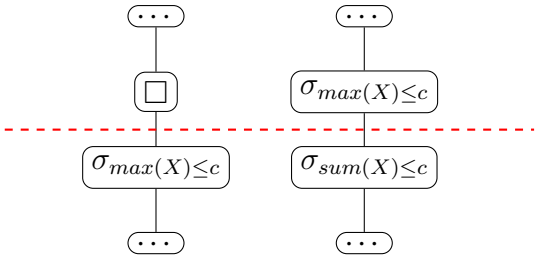


Abbildung C.25: Transformation A17

Originalanfrage:

```
SELECT aid
FROM tracks
GROUP BY aid
HAVING max(length*-1) <= -250000
```

Transformierte Anfrage:

```
SELECT aid
FROM (
  SELECT aid, length*-1 AS L
  FROM tracks
  WHERE aid IN (
    SELECT aid
    FROM tracks
    GROUP BY aid
    HAVING sum(length*-1) <= -250000
  )
) AS X
GROUP BY aid
HAVING max(L) <= -250000
```

A18: Maximum-kleiner zu Summe-kleiner (Attribut-Konstante)

Regel: $\sigma_{max(X)<c}(r) \sqsubseteq_K \sigma_{sum(X)<c}(r)$
Invarianten: $c \leq 0$
Vorbedingungen:

- Untere Ebene L_i: $\{max, <_c\} \not\subseteq O_i$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: $\{sum, <_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: $\{max, <_c\} \subseteq O_{i+1}$

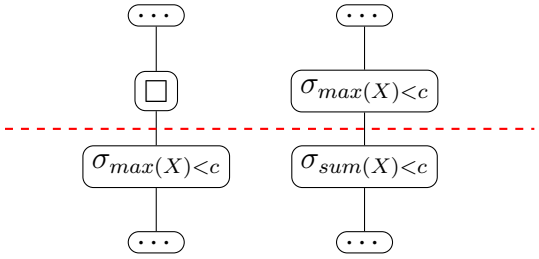


Abbildung C.26: Transformation A18

Originalanfrage:

```
SELECT aid
FROM tracks
GROUP BY aid
HAVING max(length*-1) < -250000
```

Transformierte Anfrage:

```
SELECT aid
FROM (
  SELECT aid, length*-1 AS L
  FROM tracks
  WHERE aid IN (
    SELECT aid
    FROM tracks
    GROUP BY aid
    HAVING sum(length*-1) < -250000
  )
) AS X
GROUP BY aid
HAVING max(L) < -250000
```

A19: Durchschnitt-größer-gleich zu Summe-größer-gleich (Attribut-Konstante)

Regel: $\sigma_{avg(X)\geq c}(r) \sqsubseteq_K \sigma_{sum(X)\geq c}(r)$
Invarianten: $c \geq 0$
Vorbedingungen:

- Untere Ebene L_i: $\{avg\} \not\subseteq O_i, \{\geq_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: $\{sum, \geq_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: $\{avg, \geq_c\} \subseteq O_{i+1}$

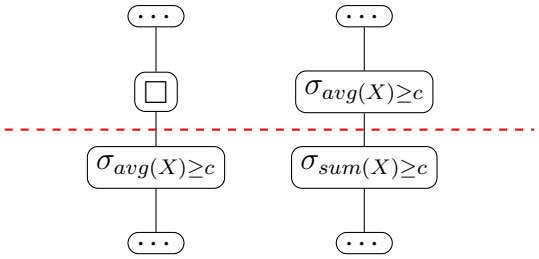


Abbildung C.27: Transformation A19

Originalanfrage:

```
SELECT aid
FROM tracks
GROUP BY aid
HAVING avg(length) >= 250000
```

Transformierte Anfrage:

```
SELECT aid
FROM (
  SELECT aid, length
  FROM tracks
  WHERE aid IN (
    SELECT aid
    FROM tracks
    GROUP BY aid
    HAVING sum(length) >= 250000
  )
) AS X
GROUP BY aid
HAVING avg(length) >= 250000
```

A20: Durchschnitt-größer zu Summe-größer (Attribut-Konstante)

Regel: $\sigma_{avg(X)>c}(r) \sqsubseteq_K \sigma_{sum(X)>c}(r)$
Invarianten: $c \geq 0$
Vorbedingungen:

- Untere Ebene L_i: $\{avg\} \not\subseteq O_i, \{>_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: $\{sum, >_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: $\{avg, >_c\} \subseteq O_{i+1}$

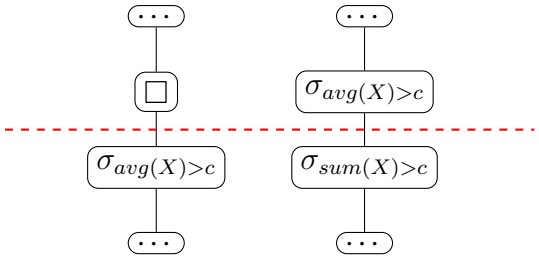


Abbildung C.28: Transformation A20

Originalanfrage:

```
SELECT aid
FROM tracks
GROUP BY aid
HAVING avg(length) > 250000
```

Transformierte Anfrage:

```
SELECT aid
FROM (
  SELECT aid, length
  FROM tracks
  WHERE aid IN (
    SELECT aid
    FROM tracks
    GROUP BY aid
    HAVING sum(length) > 250000
  )
) AS X
GROUP BY aid
HAVING avg(length) > 250000
```

A21: Durchschnitt-kleiner-gleich zu Summe-kleiner-gleich (Attribut-Konstante)

Regel: $\sigma_{avg(X)\leq c}(r) \sqsubseteq_K \sigma_{sum(X)\leq c}(r)$
Invarianten: $c \leq 0$
Vorbedingungen:

- Untere Ebene L_i: $\{avg\} \not\subseteq O_i, \{\leq_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: $\{sum, \leq_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: $\{avg, \leq_c\} \subseteq O_{i+1}$

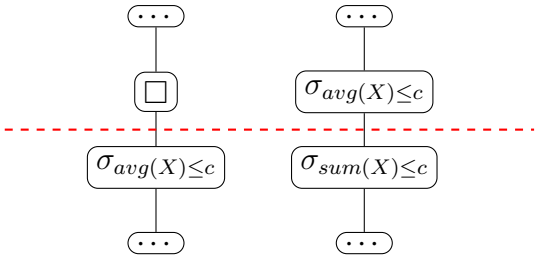


Abbildung C.29: Transformation A21

Originalanfrage:

```
SELECT aid
FROM tracks
GROUP BY aid
HAVING avg(length*-1) <= -250000
```

Transformierte Anfrage:

```
SELECT aid
FROM (
  SELECT aid, length*-1 AS L
  FROM tracks
  WHERE aid IN (
    SELECT aid
    FROM tracks
    GROUP BY aid
    HAVING sum(length*-1) <= -250000
  )
) AS X
GROUP BY aid
HAVING avg(L) <= -250000
```

A22: Durchschnitt-kleiner zu Summe-kleiner (Attribut-Konstante)

Regel: $\sigma_{avg(X)<c}(r) \sqsubseteq_K \sigma_{sum(X)<c}(r)$
Invarianten: $c \leq 0$
Vorbedingungen:

- Untere Ebene L_i: $\{avg\} \not\subseteq O_i, \{<_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: $\{sum, <_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: $\{avg, <_c\} \subseteq O_{i+1}$

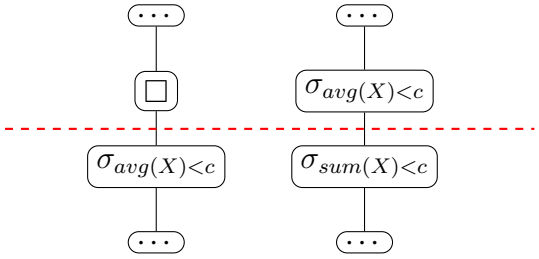


Abbildung C.30: Transformation A22

Originalanfrage:

```
SELECT aid
FROM tracks
GROUP BY aid
HAVING avg(length*-1) < -250000
```

Transformierte Anfrage:

```
SELECT aid
FROM (
  SELECT aid, length*-1 AS L
  FROM tracks
  WHERE aid IN (
    SELECT aid
    FROM tracks
    GROUP BY aid
    HAVING sum(length*-1) < -250000
  )
) AS X
GROUP BY aid
HAVING avg(L) < -250000
```

A23: Summe-größer-gleich zu Durchschnitt-größer-gleich-0 (Attribut-Konstante)

Regel: $\sigma_{sum(X)\geq c}(r) \sqsubseteq_K \sigma_{avg(X)\geq 0}(r)$
Invarianten: $c \geq 0$
Vorbedingungen:

- Untere Ebene L_i: $\{sum\} \not\subseteq O_i, \{\geq_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: $\{avg, \geq_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: $\{sum, \geq_c\} \subseteq O_{i+1}$

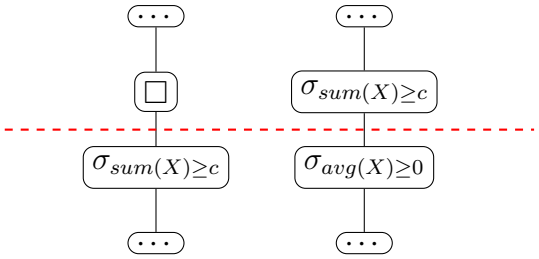


Abbildung C.31: Transformation A23

Originalanfrage:

```
SELECT aid
FROM tracks
GROUP BY aid
HAVING sum(length) >= 250000
```

Transformierte Anfrage:

```
SELECT aid
FROM (
  SELECT aid, length
  FROM tracks
  WHERE aid IN (
    SELECT aid
    FROM tracks
    GROUP BY aid
    HAVING avg(length) >= 0
  )
) AS X
GROUP BY aid
HAVING sum(length) >= 250000
```

A24: Summe-größer zu Durchschnitt-größer-0 (Attribut-Konstante)

Regel: $\sigma_{sum(X)>c}(r) \sqsubseteq_K \sigma_{avg(X)>0}(r)$
Invarianten: $c \geq 0$
Vorbedingungen:

- Untere Ebene L_i: $\{sum\} \not\subseteq O_i$, $\{>_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: $\{avg, >_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: $\{sum, >_c\} \not\subseteq O_{i+1}$

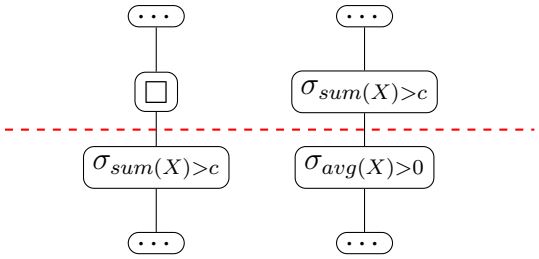


Abbildung C.32: Transformation A24

Originalanfrage:

```
SELECT aid
FROM tracks
GROUP BY aid
HAVING sum(length) > 250000
```

Transformierte Anfrage:

```
SELECT aid
FROM (
  SELECT aid, length
  FROM tracks
  WHERE aid IN (
    SELECT aid
    FROM tracks
    GROUP BY aid
    HAVING avg(length) > 0
  )
) AS X
GROUP BY aid
HAVING sum(length) > 250000
```

A25: Summe-kleiner-gleich zu Durchschnitt-kleiner-gleich-0 (Attribut-Konstante)

Regel: $\sigma_{sum(X)\leq c}(r) \sqsubseteq_K \sigma_{avg(X)\leq 0}(r)$
Invarianten: $c \leq 0$
Vorbedingungen:

- Untere Ebene L_i: $\{sum\} \notin O_i, \{\leq_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: $\{avg, \leq_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: $\{sum, \leq_c\} \subseteq O_{i+1}$

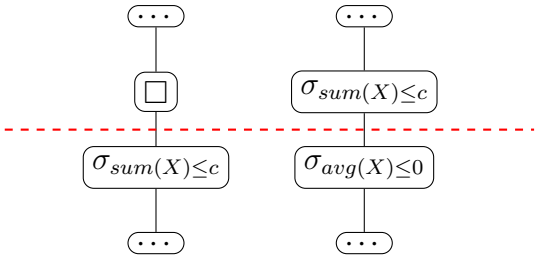


Abbildung C.33: Transformation A25

Originalanfrage:

```
SELECT aid
FROM tracks
GROUP BY aid
HAVING sum(length*-1) <= -250000
```

Transformierte Anfrage:

```
SELECT aid
FROM (
  SELECT aid, length*-1 AS L
  FROM tracks
  WHERE aid IN (
    SELECT aid
    FROM tracks
    GROUP BY aid
    HAVING avg(length*-1) <= 0
  )
) AS X
GROUP BY aid
HAVING sum(L) <= -250000
```

A26: Summe-kleiner zu Durchschnitt-kleiner-0 (Attribut-Konstante)

Regel: $\sigma_{sum(X)<c}(r) \sqsubseteq_K \sigma_{avg(X)<0}(r)$
Invarianten: $c \leq 0$
Vorbedingungen:

- Untere Ebene L_i: $\{sum\} \not\subseteq O_i, \{<_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: $\{avg, <_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: $\{sum, <_c\} \subseteq O_{i+1}$

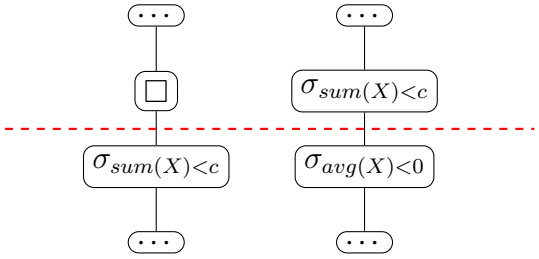


Abbildung C.34: Transformation A26

Originalanfrage:

```
SELECT aid
FROM tracks
GROUP BY aid
HAVING sum(length*-1) < -250000
```

Transformierte Anfrage:

```
SELECT aid
FROM (
  SELECT aid, length*-1 AS L
  FROM tracks
  WHERE aid IN (
    SELECT aid
    FROM tracks
    GROUP BY aid
    HAVING avg(length*-1) < 0
  )
) AS X
GROUP BY aid
HAVING sum(L) < -250000
```

A27: Alle-kleiner zu Maximum-kleiner (Attribut-Konstante)

Regel: $\sigma_{\forall x \in X : x < c}(r) \sqsubseteq_K \sigma_{max(X) < c}(r)$
Invarianten: —
Vorbedingungen:

- Untere Ebene L_i: $\{\forall\} \not\subseteq O_i, \{<_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: $\{max, <_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: —

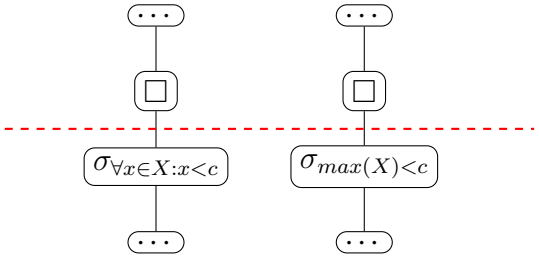


Abbildung C.35: Transformation A27

Originalanfrage:

```
SELECT DISTINCT aid
FROM tracks
WHERE aid NOT IN (
  SELECT aid
  FROM tracks
  WHERE NOT(length < 250000)
)
```

Transformierte Anfrage:

```
SELECT DISTINCT aid
FROM tracks
GROUP BY aid
HAVING max(length) < 250000
OR max(length) IS NULL
```

A28: Alle-kleiner-gleich zu Maximum-kleiner-gleich (Attribut-Konstante)

Regel: $\sigma_{\forall x \in X : x \leq c}(r) \sqsubseteq_K \sigma_{max(X) \leq c}(r)$
Invarianten: —
Vorbedingungen:

- Untere Ebene L_i: $\{\forall\} \not\subseteq O_i, \{\leq_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: $\{max, \leq_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: —

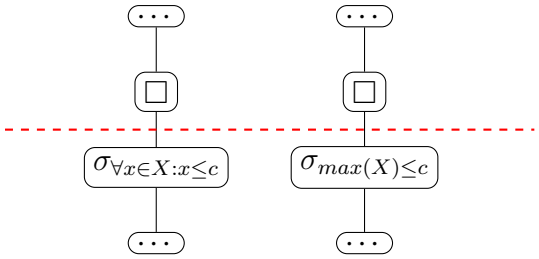


Abbildung C.36: Transformation A28

Originalanfrage:

```
SELECT DISTINCT aid
FROM tracks
WHERE aid NOT IN (
  SELECT aid
  FROM tracks
  WHERE NOT(length <= 250000)
)
```

Transformierte Anfrage:

```
SELECT DISTINCT aid
FROM tracks
GROUP BY aid
HAVING max(length) <= 250000
OR max(length) IS NULL
```

A29: Alle-gleich zu Minimum-gleich (Attribut-Konstante)

Regel: $\sigma_{\forall x \in X : x=c}(r) \sqsubseteq_K \sigma_{min(X)=c}(r)$
Invarianten: —
Vorbedingungen:

- Untere Ebene L_i: $\{\forall\} \not\subseteq O_i, \{=_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: $\{min, =_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: $\{\forall, =_c\} \subseteq O_{i+1}$

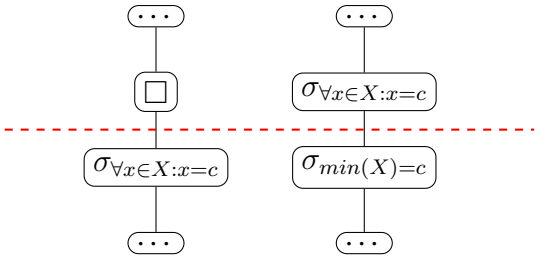


Abbildung C.37: Transformation A29

Originalanfrage:

```
SELECT DISTINCT aid
FROM tracks
WHERE aid NOT IN (
  SELECT aid
  FROM tracks
  WHERE NOT(length = 250000)
)
```

Transformierte Anfrage:

```
SELECT DISTINCT aid
FROM tracks
WHERE aid IN (
  SELECT aid
  FROM tracks
  GROUP BY aid
  HAVING min(length) = 250000
  OR min(length) IS NULL
)
AND aid NOT IN (
  SELECT aid
  FROM tracks
  WHERE NOT(length = 250000)
)
```

A30: Alle-gleich zu Maximum-gleich (Attribut-Konstante)

Regel: $\sigma_{\forall x \in X : x = c}(r) \sqsubseteq_K \sigma_{max(X)=c}(r)$
Invarianten: —
Vorbedingungen:

- Untere Ebene L_i: $\{\forall\} \not\subseteq O_i, \{=_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: $\{max, =_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: $\{\forall, =_c\} \subseteq O_{i+1}$

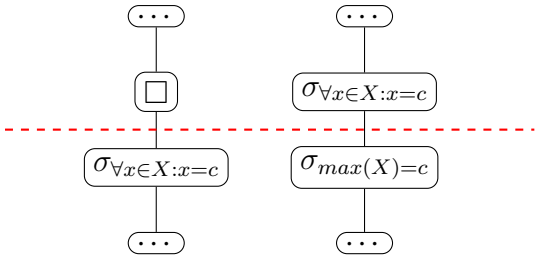


Abbildung C.38: Transformation A30

Originalanfrage:

```
SELECT DISTINCT aid
FROM tracks
WHERE aid NOT IN (
  SELECT aid
  FROM tracks
  WHERE NOT(length = 250000)
)
```

Transformierte Anfrage:

```
SELECT DISTINCT aid
FROM tracks
WHERE aid IN (
  SELECT aid
  FROM tracks
  GROUP BY aid
  HAVING max(length) = 250000
  OR max(length) IS NULL
)
AND aid NOT IN (
  SELECT aid
  FROM tracks
  WHERE NOT(length = 250000)
)
```

A31: Alle-ungleich zu Minimum-ungleich (Attribut-Konstante)

Regel: $\sigma_{\forall x \in X : x \neq c}(r) \sqsubseteq_K \sigma_{min(X) \neq c}(r)$
Invarianten: —
Vorbedingungen:

- Untere Ebene L_i: $\{\forall\} \not\subseteq O_i$, $\{\neq_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: $\{min, \neq_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: $\{\forall, \neq_c\} \subseteq O_{i+1}$

min wird unterstützt

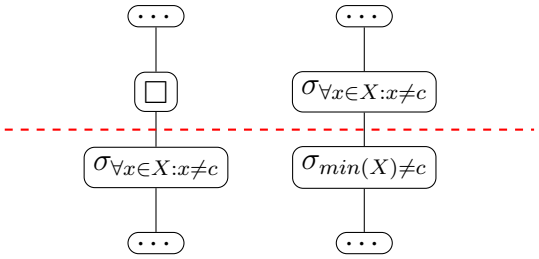


Abbildung C.39: Transformation A31

Originalanfrage:

```
SELECT DISTINCT aid
FROM tracks
WHERE aid NOT IN (
  SELECT aid
  FROM tracks
  WHERE NOT(length != 250000)
)
```

Transformierte Anfrage:

```
SELECT DISTINCT aid
FROM tracks
GROUP BY aid
HAVING min(length) != 250000
AND aid NOT IN (
  SELECT aid
  FROM tracks
  WHERE NOT(length != 250000)
)
OR max(length) IS NULL
```

A32: Alle-ungleich zu Maximum-ungleich (Attribut-Konstante)

Regel: $\sigma_{\forall x \in X : x \neq c}(r) \sqsubseteq_K \sigma_{max(X) \neq c}(r)$
Invarianten: —
Vorbedingungen:

- Untere Ebene L_i: $\{\forall\} \not\subseteq O_i, \{\neq_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: $\{max, \neq_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: $\{\forall, \neq_c\} \subseteq O_{i+1}$

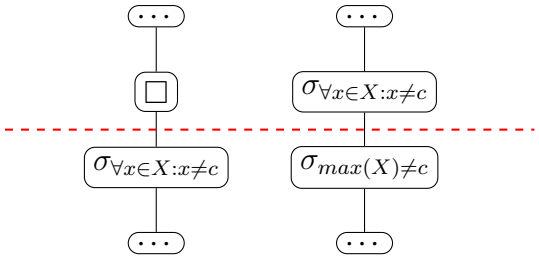


Abbildung C.40: Transformation A32

Originalanfrage:

```
SELECT DISTINCT aid
FROM tracks
WHERE aid NOT IN (
  SELECT aid
  FROM tracks
  WHERE NOT(length != 250000)
)
```

Transformierte Anfrage:

```
SELECT DISTINCT aid
FROM tracks
GROUP BY aid
HAVING max(length) != 250000
AND aid NOT IN (
  SELECT aid
  FROM tracks
  WHERE NOT(length != 250000)
)
OR max(length) IS NULL
```

A33: Alle-größer-gleich zu Minimum-größer-gleich (Attribut-Konstante)

Regel: $\sigma_{\forall x \in X : x \geq c}(r) \sqsubseteq_K \sigma_{min(X) \geq c}(r)$
Invarianten: —
Vorbedingungen:

- Untere Ebene L_i: $\{\forall\} \not\subseteq O_i, \{\geq_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: $\{min, \geq_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: —

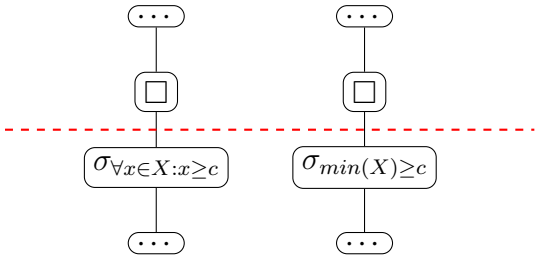


Abbildung C.41: Transformation A33

Originalanfrage:

```
SELECT DISTINCT aid
FROM tracks
WHERE aid NOT IN (
  SELECT aid
  FROM tracks
  WHERE NOT(length >= 250000)
)
```

Transformierte Anfrage:

```
SELECT DISTINCT aid
FROM tracks
GROUP BY aid
HAVING min(length) >= 250000
OR max(length) IS NULL
```

A34: Alle-größer zu Minimum-größer (Attribut-Konstante)

Regel: $\sigma_{\forall x \in X : x > c}(r) \sqsubseteq_K \sigma_{min(X) > c}(r)$
Invarianten: —
Vorbedingungen:

- Untere Ebene L_i: $\{\forall\} \not\subseteq O_i, \{>_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: $\{min, >_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: —

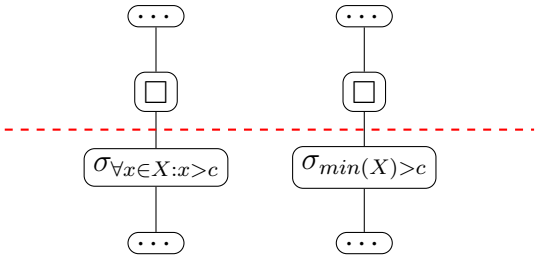


Abbildung C.42: Transformation A34

Originalanfrage:

```
SELECT DISTINCT aid
FROM tracks
WHERE aid NOT IN (
  SELECT aid
  FROM tracks
  WHERE NOT(length > 250000)
)
```

Transformierte Anfrage:

```
SELECT DISTINCT aid
FROM tracks
GROUP BY aid
HAVING min(length) > 250000
OR max(length) IS NULL
```

A35: Alle-kleiner zu Durchschnitt-kleiner (Attribut-Konstante)

Regel: $\sigma_{\forall x \in X : x < c}(r) \sqsubseteq_K \sigma_{avg(X) < c}(r)$
Invarianten: —
Vorbedingungen:

- Untere Ebene L_i: $\{\forall\} \not\subseteq O_i, \{<_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: $\{avg, <_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: $\{\forall, <_c\} \subseteq O_{i+1}$

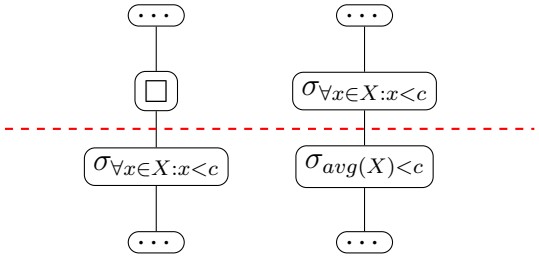


Abbildung C.43: Transformation A35

Originalanfrage:

```
SELECT DISTINCT aid
FROM tracks
WHERE aid NOT IN (
  SELECT aid
  FROM tracks
  WHERE NOT(length < 250000)
)
```

Transformierte Anfrage:

```
SELECT DISTINCT aid
FROM tracks
WHERE aid NOT IN (
  SELECT aid
  FROM tracks
  GROUP BY aid
  HAVING NOT(avg(length) < 250000)
)
AND aid NOT IN (
  SELECT aid
  FROM tracks
  WHERE NOT(length < 250000)
)
```

A36: Alle-kleiner-gleich zu Durchschnitt-kleiner-gleich (Attribut-Konstante)

Regel: $\sigma_{\forall x \in X : x \leq c}(r) \sqsubseteq_K \sigma_{avg(X) \leq c}(r)$
Invarianten: —
Vorbedingungen:

- Untere Ebene L_i: $\{\forall\} \not\subseteq O_i, \{\leq_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: $\{avg, \leq_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: $\{\forall, \leq_c\} \subseteq O_{i+1}$

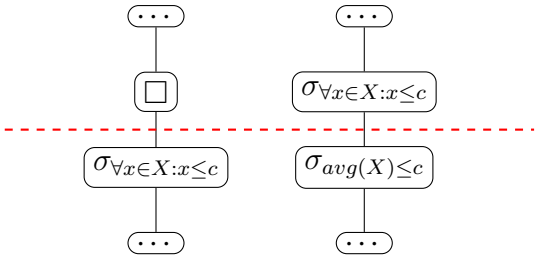


Abbildung C.44: Transformation A36

Originalanfrage:

```
SELECT DISTINCT aid
FROM tracks
WHERE aid NOT IN (
  SELECT aid
  FROM tracks
  WHERE NOT(length <= 250000)
)
```

Transformierte Anfrage:

```
SELECT DISTINCT aid
FROM tracks
WHERE aid NOT IN (
  SELECT aid
  FROM tracks
  GROUP BY aid
  HAVING NOT(avg(length) <= 250000)
)
AND aid NOT IN (
  SELECT aid
  FROM tracks
  WHERE NOT(length <= 250000)
)
```

A37: Alle-gleich zu Durchschnitt-gleich (Attribut-Konstante)

Regel: $\sigma_{\forall x \in X : x = c}(r) \sqsubseteq_K \sigma_{avg(X)=c}(r)$
Invarianten: —
Vorbedingungen:

- Untere Ebene L_i: $\{\forall\} \not\subseteq O_i, \{=_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: $\{avg, =_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: $\{\forall, =_c\} \subseteq O_{i+1}$

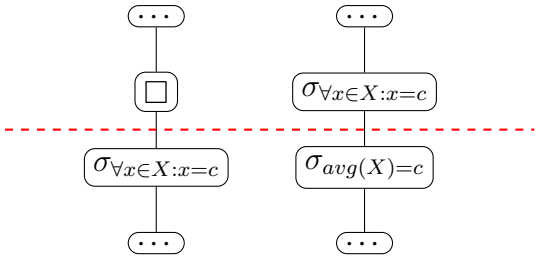


Abbildung C.45: Transformation A37

Originalanfrage:

```
SELECT DISTINCT aid
FROM tracks
WHERE aid NOT IN (
  SELECT aid
  FROM tracks
  WHERE NOT(length = 250000)
)
```

Transformierte Anfrage:

```
SELECT DISTINCT aid
FROM tracks
WHERE aid NOT IN (
  SELECT aid
  FROM tracks
  GROUP BY aid
  HAVING NOT(avg(length) = 250000)
)
AND aid NOT IN (
  SELECT aid
  FROM tracks
  WHERE NOT(length = 250000)
)
```

A38: Alle-größer-gleich zu Durchschnitt-größer-gleich (Attribut-Konstante)

Regel: $\sigma_{\forall x \in X : x \geq c}(r) \sqsubseteq_K \sigma_{avg(X) \geq c}(r)$
Invarianten: —
Vorbedingungen:

- Untere Ebene L_i: $\{\forall\} \not\subseteq O_i, \{\geq_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: $\{avg, \geq_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: $\{\forall, \geq_c\} \subseteq O_{i+1}$

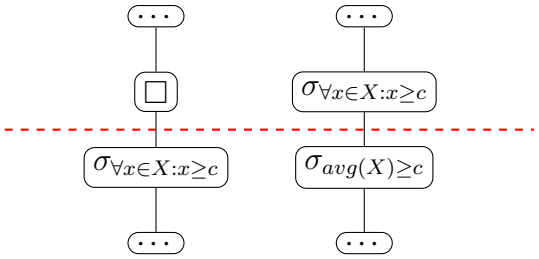


Abbildung C.46: Transformation A38

Originalanfrage:

```
SELECT DISTINCT aid
FROM tracks
WHERE aid NOT IN (
  SELECT aid
  FROM tracks
  WHERE NOT(length >= 250000)
)
```

Transformierte Anfrage:

```
SELECT DISTINCT aid
FROM tracks
WHERE aid NOT IN (
  SELECT aid
  FROM tracks
  GROUP BY aid
  HAVING NOT(avg(length) >= 250000)
)
AND aid NOT IN (
  SELECT aid
  FROM tracks
  WHERE NOT(length >= 250000)
)
```

A39: Alle-größer zu Durchschnitt-größer (Attribut-Konstante)

Regel: $\sigma_{\forall x \in X : x > c}(r) \sqsubseteq_K \sigma_{avg(X) > c}(r)$
Invarianten: —
Vorbedingungen:

- Untere Ebene L_i: $\{\forall\} \not\subseteq O_i, \{>_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: $\{avg, >_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: $\{\forall, >_c\} \subseteq O_{i+1}$

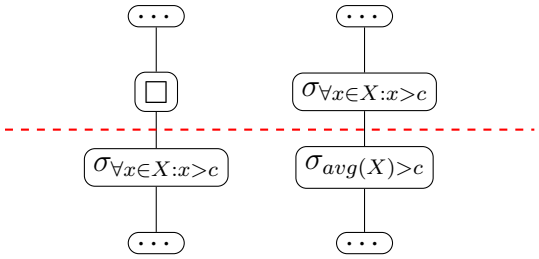


Abbildung C.47: Transformation A39

Originalanfrage:

```
SELECT DISTINCT aid
FROM tracks
WHERE aid NOT IN (
  SELECT aid
  FROM tracks
  WHERE NOT(length > 250000)
)
```

Transformierte Anfrage:

```
SELECT DISTINCT aid
FROM tracks
WHERE aid NOT IN (
  SELECT aid
  FROM tracks
  GROUP BY aid
  HAVING NOT(avg(length) > 250000)
)
AND aid NOT IN (
  SELECT aid
  FROM tracks
  WHERE NOT(length > 250000)
)
```

A40: Alle-kleiner-gleich zu Summe-kleiner-gleich (Attribut-Konstante)

Regel: $\sigma_{\forall x \in X : x \leq c}(r) \sqsubseteq_K \sigma_{sum(X) \leq c}(r)$
Invarianten: $c \leq 0$
Vorbedingungen:

- Untere Ebene L_i: $\{\forall\} \not\subseteq O_i, \{\leq_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: $\{sum, \leq_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: $\{\forall, \leq_c\} \subseteq O_{i+1}$

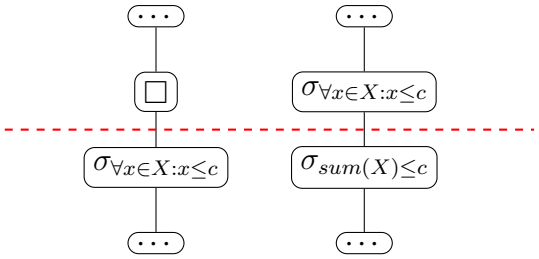


Abbildung C.48: Transformation A40

Originalanfrage:

```
SELECT DISTINCT aid
FROM tracks
WHERE aid NOT IN (
  SELECT aid
  FROM tracks
  WHERE NOT(length*-1 <= -250000)
)
```

Transformierte Anfrage:

```
SELECT DISTINCT aid
FROM tracks
WHERE aid NOT IN (
  SELECT aid
  FROM tracks
  GROUP BY aid
  HAVING NOT(
    sum(length*-1) <= -250000
  )
)
AND aid NOT IN (
  SELECT aid
  FROM tracks
  WHERE NOT(length*-1 <= -250000)
)
```

A41: Alle-kleiner zu Summe-kleiner (Attribut-Konstante)

Regel: $\sigma_{\forall x \in X : x < c}(r) \sqsubseteq_K \sigma_{sum(X) < c}(r)$
Invarianten: $c \leq 0$
Vorbedingungen:

- Untere Ebene L_i: $\{\forall\} \not\subseteq O_i, \{<_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: $\{sum, <_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: $\{\forall, <_c\} \subseteq O_{i+1}$

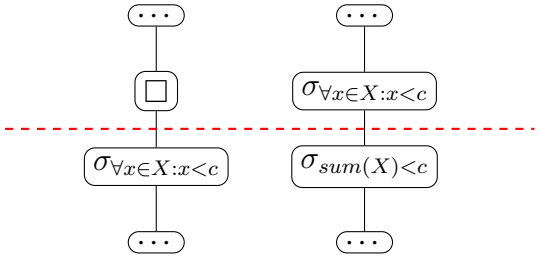


Abbildung C.49: Transformation A41

Originalanfrage:

```
SELECT DISTINCT aid
FROM tracks
WHERE aid NOT IN (
  SELECT aid
  FROM tracks
  WHERE NOT(length*-1 < -250000)
)
```

Transformierte Anfrage:

```
SELECT DISTINCT aid
FROM tracks
WHERE aid NOT IN (
  SELECT aid
  FROM tracks
  GROUP BY aid
  HAVING NOT(
    sum(length*-1) < -250000
  )
)
AND aid NOT IN (
  SELECT aid
  FROM tracks
  WHERE NOT(length*-1 < -250000)
)
```

A42: Alle-gleich zu Summe-kleiner-gleich (Attribut-Konstante)

Regel: $\sigma_{\forall x \in X : x=c}(r) \sqsubseteq_K \sigma_{sum(X) \leq c}(r)$
Invarianten: $c \leq 0$
Vorbedingungen:

- Untere Ebene L_i: $\{\forall, =_c\} \not\subseteq O_i$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: $\{sum, \leq_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: $\{\forall, =_c\} \subseteq O_{i+1}$

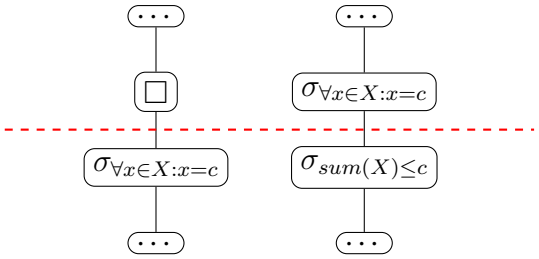


Abbildung C.50: Transformation A42

Originalanfrage:

```
SELECT DISTINCT aid
FROM tracks
WHERE aid NOT IN (
  SELECT aid
  FROM tracks
  WHERE NOT(length*-1 = -250000)
)
```

Transformierte Anfrage:

```
SELECT DISTINCT aid
FROM tracks
WHERE aid NOT IN (
  SELECT aid
  FROM tracks
  GROUP BY aid
  HAVING NOT(
    sum(length*-1) <= -250000
  )
)
AND aid NOT IN (
  SELECT aid
  FROM tracks
  WHERE NOT(length*-1 = -250000)
)
```

A43: Alle-größer zu Summe-größer (Attribut-Konstante)

Regel: $\sigma_{\forall x \in X : x > c}(r) \sqsubseteq_K \sigma_{sum(X) > c}(r)$
Invarianten: $c \geq 0$
Vorbedingungen:

- Untere Ebene L_i: $\{\forall\} \not\subseteq O_i, \{>_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: $\{sum, >_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: $\{\forall, >_c\} \subseteq O_{i+1}$

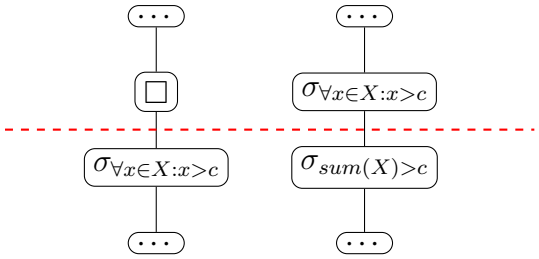


Abbildung C.51: Transformation A43

Originalanfrage:

```
SELECT DISTINCT aid
FROM tracks
WHERE aid NOT IN (
  SELECT aid
  FROM tracks
  WHERE NOT(length = 250000)
)
```

Transformierte Anfrage:

```
SELECT DISTINCT aid
FROM tracks
WHERE aid NOT IN (
  SELECT aid
  FROM tracks
  GROUP BY aid
  HAVING NOT(sum(length) <= 250000)
)
AND aid NOT IN (
  SELECT aid
  FROM tracks
  WHERE NOT(length = 250000)
)
```

A44: Alle-größer-gleich zu Summe-größer-gleich (Attribut-Konstante)

Regel: $\sigma_{\forall x \in X : x \geq c}(r) \sqsubseteq_K \sigma_{sum(X) \geq c}(r)$
Invarianten: $c \geq 0$
Vorbedingungen:

- Untere Ebene L_i: $\{\forall\} \not\subseteq O_i, \{\geq_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: $\{sum, \geq_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: $\{\forall, \geq_c\} \subseteq O_{i+1}$

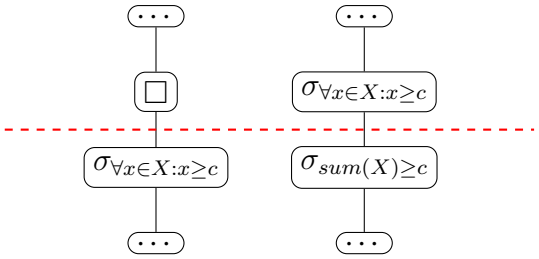


Abbildung C.52: Transformation A44

Originalanfrage:

```
SELECT DISTINCT aid
FROM tracks
WHERE aid NOT IN (
  SELECT aid
  FROM tracks
  WHERE NOT(length >= 250000)
)
```

Transformierte Anfrage:

```
SELECT DISTINCT aid
FROM tracks
WHERE aid NOT IN (
  SELECT aid
  FROM tracks
  GROUP BY aid
  HAVING NOT(sum(length) >= 250000)
)
AND aid NOT IN (
  SELECT aid
  FROM tracks
  WHERE NOT(length >= 250000)
)
```

A45: Alle-gleich zu Summe-größer-gleich (Attribut-Konstante)

Regel: $\sigma_{\forall x \in X : x=c}(r) \sqsubseteq_K \sigma_{sum(X) \geq c}(r)$
Invarianten: $c \geq 0$
Vorbedingungen:

- Untere Ebene L_i: $\{\forall, =_c\} \not\subseteq O_i$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: $\{sum, \geq_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: $\{\forall, =_c\} \subseteq O_{i+1}$

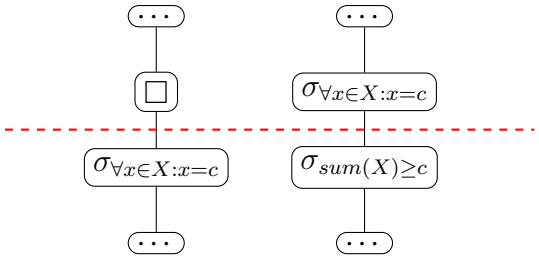


Abbildung C.53: Transformation A45

Originalanfrage:

```
SELECT DISTINCT aid
FROM tracks
WHERE aid NOT IN (
  SELECT aid
  FROM tracks
  WHERE NOT(length = 250000)
)
```

Transformierte Anfrage:

```
SELECT DISTINCT aid
FROM tracks
WHERE aid NOT IN (
  SELECT aid
  FROM tracks
  GROUP BY aid
  HAVING NOT(sum(length) >= 250000)
)
AND aid NOT IN (
  SELECT aid
  FROM tracks
  WHERE NOT(length = 250000)
)
```

A46: Minimum-größer zu Alle-größer (Attribut-Konstante)

Regel: $\sigma_{min(X)>c}(r) \sqsubseteq_K \sigma_{\forall x \in X: x>c}(r)$
Invarianten: —
Vorbedingungen:

- Untere Ebene L_i: $\{min\} \not\subseteq O_i$, $\{>_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: $\{\forall, >_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: —

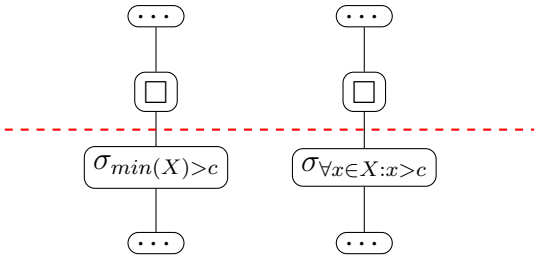


Abbildung C.54: Transformation A46

Originalanfrage:

```
SELECT DISTINCT aid
FROM tracks
GROUP BY aid
HAVING min(length) > 250000
OR min(length) IS NULL
```

Transformierte Anfrage:

```
SELECT DISTINCT aid
FROM tracks
WHERE aid NOT IN (
  SELECT aid
  FROM tracks
  WHERE NOT(length > 250000)
)
```

A47: Minimum-größer-gleich zu Alle-größer-gleich (Attribut-Konstante)

Regel: $\sigma_{min(X)\geq c}(r) \sqsubseteq_K \sigma_{\forall x \in X: x\geq c}(r)$
Invarianten: —
Vorbedingungen:

- Untere Ebene L_i: $\{min\} \not\subseteq O_i$, $\{\geq\} \subseteq O_i$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: $\{\forall, \geq_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: —

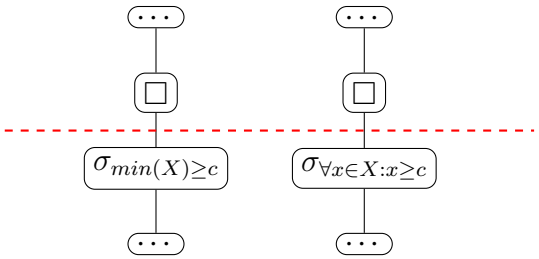


Abbildung C.55: Transformation A47

Originalanfrage:

```
SELECT DISTINCT aid
FROM tracks
GROUP BY aid
HAVING min(length) >= 250000
OR max(length) IS NULL
```

Transformierte Anfrage:

```
SELECT DISTINCT aid
FROM tracks
WHERE aid NOT IN (
  SELECT aid
  FROM tracks
  WHERE NOT(length >= 250000)
)
```

A48: Minimum-gleich zu Alle-größer-gleich (Attribut-Konstante)

Regel: $\sigma_{min(X)=c}(r) \sqsubseteq_K \sigma_{\forall x \in X: x \geq c}(r)$
Invarianten: —
Vorbedingungen:

- Untere Ebene L_i: $\{min, =_c\} \not\subseteq O_i$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: $\{\forall, \geq_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: $\{min, =_c\} \subseteq O_{i+1}$

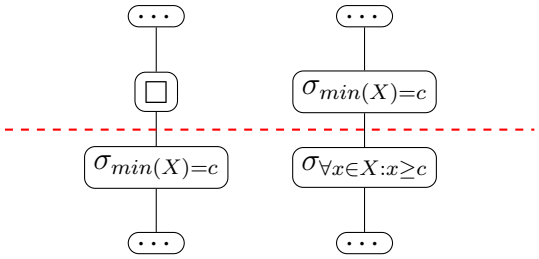


Abbildung C.56: Transformation A48

Originalanfrage:

```
SELECT DISTINCT aid
FROM tracks
GROUP BY aid
HAVING min(length) = 250000
```

Transformierte Anfrage:

```
SELECT DISTINCT aid
FROM tracks
WHERE aid NOT IN (
  SELECT aid
  FROM tracks
  WHERE NOT(length >= 250000)
)
AND aid IN (
  SELECT aid
  FROM tracks
  GROUP BY aid
  HAVING min(length) = 250000
)
```

A49: Maximum-kleiner zu Alle-kleiner (Attribut-Konstante)

Regel: $\sigma_{max(X)<c}(r) \sqsubseteq_K \sigma_{\forall x \in X: x<c}(r)$
Invarianten: —
Vorbedingungen:

- Untere Ebene L_i: $\{max\} \not\subseteq O_i, \{<_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: $\{\forall, <_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: —

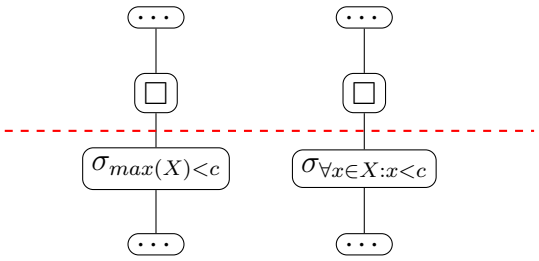


Abbildung C.57: Transformation A49

Originalanfrage:

```
SELECT DISTINCT aid
FROM tracks
GROUP BY aid
HAVING max(length) < 250000
OR max(length) IS NULL
```

Transformierte Anfrage:

```
SELECT DISTINCT aid
FROM tracks
WHERE aid NOT IN (
  SELECT aid
  FROM tracks
  WHERE NOT(length < 250000)
)
```

A50: Maximum-kleiner-gleich zu Alle-kleiner-gleich (Attribut-Konstante)

Regel: $\sigma_{max(X)\leq c}(r) \sqsubseteq_K \sigma_{\forall x \in X: x\leq c}(r)$
Invarianten: —
Vorbedingungen:

- Untere Ebene L_i: $\{max\} \not\subseteq O_i, \{\leq_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: $\{\forall, \leq_c\} \subseteq O_i$
- Obere Ebene L_{i+1}: —

$\forall$ wird unterstützt

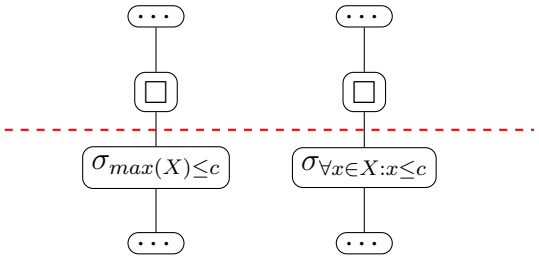


Abbildung C.58: Transformation A50

Originalanfrage:

```
SELECT DISTINCT aid
FROM tracks
GROUP BY aid
HAVING max(length) <= 250000
OR max(length) IS NULL
```

Transformierte Anfrage:

```
SELECT DISTINCT aid
FROM tracks
WHERE aid NOT IN (
  SELECT aid
  FROM tracks
  WHERE NOT(length <= 250000)
)
```

A51: Maximum-gleich zu Alle-kleiner-gleich (Attribut-Konstante)

Regel: $\sigma_{max(X)=c}(r) \sqsubseteq_K \sigma_{\forall x \in X: x \leq c}(r)$
Invarianten: —
Vorbedingungen:

- Untere Ebene L_i: $\{max, =_c\} \not\subseteq O_i$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: $\{\forall, \leq\} \subseteq O_i$
- Obere Ebene L_{i+1}: $\{max, =_c\} \subseteq O_{i+1}$

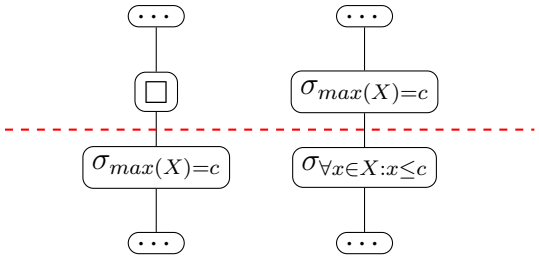


Abbildung C.59: Transformation A51

Originalanfrage:

```
SELECT DISTINCT aid
FROM tracks
GROUP BY aid
HAVING max(length) = 250000
```

Transformierte Anfrage:

```
SELECT DISTINCT aid
FROM tracks
WHERE aid NOT IN (
  SELECT aid
  FROM tracks
  WHERE NOT(length <= 250000)
)
AND aid IN (
  SELECT aid
  FROM tracks
  GROUP BY aid
  HAVING max(length) = 250000
)
```

C.3 Klassische Optimierungsregeln mit Query Rewriting by Contract

In diesem Abschnitt werden Anfragetransformationen vorgestellt, die zu den Standardoptimierungsregeln jedes relationalen Datenbanksystems gehören. Details zu den Optimierungsregeln lassen sich in vielen Lehrbüchern, beispielsweise in [SSH11], finden.

Der Vorteil dieser Regelklasse ist, dass sie auf einfachen Operatoren, die insbesondere auf den unteren Ebenen zum Einsatz kommen, arbeitet. Dadurch können die ersten Vorselektionen bereits sehr früh in Richtung der Datenquellen auf der untersten Ebene verschoben werden.

Ein weiterer Vorteil ist, dass diese Regeln stets eine äquivalente Anfrage erzeugen. Dadurch entstehen, lokal auf zwei Ebenen beschränkt, keine zusätzlichen Tupel, wodurch die Rewriting Supremum-Eigenschaft ohne die Einführung zusätzlicher Operatoren sichergestellt wird. Dennoch ist Vorsicht bei der Ausführung geboten: Zwar liefert die reine Anwendung der Regeln eine äquivalente Anfrage, jedoch können die Kontrakt-basierten Regeln aufgrund der unterschiedlichen Vor- und Nachbedingungen sowie Invarianten nicht in umgekehrter Reihenfolge verwendet werden.

K0-1: Kommutativität des Verbunds

Regel: $r_1 \bowtie r_2 \equiv r_2 \bowtie r_1$
Invarianten: —
Vorbedingungen:

- Untere Ebene L_i: $\{\bowtie\} \subseteq O_i$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: $\{\bowtie\} \subseteq O_i$
- Obere Ebene L_{i+1}: —

Anmerkung: Die Regel wird nur für interne Vertauschungen benutzt und hat keinen direkten Einfluss auf die Anfrageverteilung.

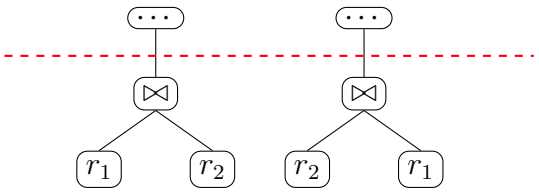


Abbildung C.60: Transformation K0-1

Originalanfrage:

```
SELECT *
FROM tracks JOIN genres
  ON(tracks.gid = genres.gid)
```

Transformierte Anfrage:

```
SELECT *
FROM genres JOIN tracks
  ON(genres.gid = tracks.gid)
```

K0-2: Assoziativität des Verbunds

Regel: $(r_1 \bowtie r_2) \bowtie r_3 \equiv r_1 \bowtie (r_2 \bowtie r_3)$
Invarianten: —
Vorbedingungen:

- Untere Ebene L_i: $\{\bowtie\} \subseteq O_i$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: $\{\bowtie\} \subseteq O_i$
- Obere Ebene L_{i+1}: —

Anmerkung: Die Regel wird nur für interne Vertauschungen benutzt und hat keinen direkten Einfluss auf die Anfrageverteilung.

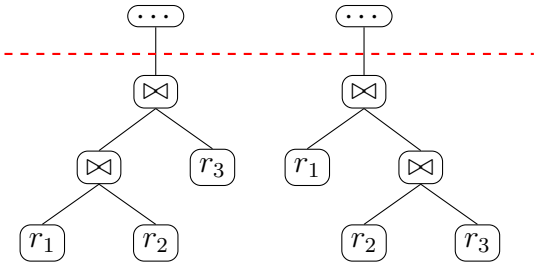


Abbildung C.61: Transformation K0-2

Originalanfrage:

```
SELECT *
FROM (
  genres JOIN tracks
    ON(genres.gid = tracks.gid)
) JOIN artists
    ON(tracks.artid = artists.artid)
```

Transformierte Anfrage:

```
SELECT *
FROM genres JOIN (
 tracks JOIN artists
  ON(tracks.artid = artists.artid
  )
) ON(genres.gid = tracks.gid)
```

K01: Entfernen redundanter Operationen

Regel: $r \bowtie r \equiv r$
Invarianten: —
Vorbedingungen:

- Untere Ebene L_i: $\{\bowtie\} \not\subseteq O_i$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: —
- Obere Ebene L_{i+1}: —

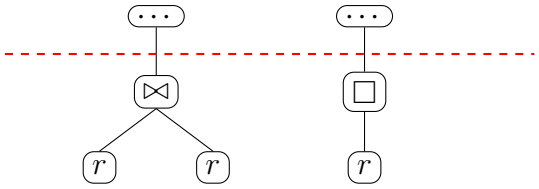


Abbildung C.62: Transformation K01

Originalanfrage:

```
SELECT t1.tid, t1.length
FROM tracks t1 JOIN tracks t2
  ON(t1.tid=t2.tid)
```

Transformierte Anfrage:

```
SELECT t1.tid, t1.length
FROM tracks t1
```

K02: Kommutativität von Selektion und Verbund

Regel: $\sigma_Y(\sigma_X(r_1) \bowtie r_2) \sqsubseteq_K \sigma_{X \wedge Y}(r_1 \bowtie r_2)$
Invarianten: $attr(X) \subseteq r_1$
Vorbedingungen:

- Untere Ebene L_i: $\{\sigma_X\} \not\subseteq O_i, \{\bowtie\} \subseteq O_i$
- Obere Ebene L_{i+1}: $\{\sigma_Y\} \subseteq O_{i+1}$

Nachbedingungen:

- Untere Ebene L_i: $\{\bowtie\} \subseteq O_i$
- Obere Ebene L_{i+1}: $\{\sigma_X, \sigma_X, \wedge\} \subseteq O_{i+1}$

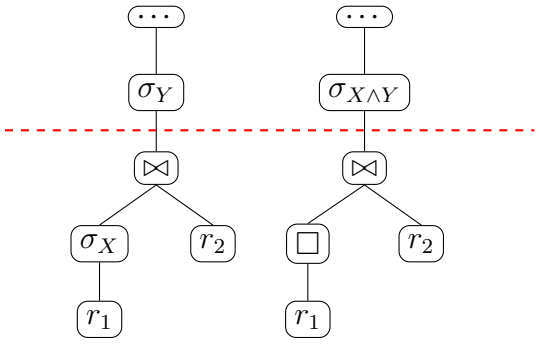


Abbildung C.63: Transformation K02

Originalanfrage:

```
WITH X AS(
  SELECT *
  FROM tracks
  WHERE length > 250000
),
Y AS(
  SELECT *
  FROM X NATURAL JOIN albums
)
SELECT *
FROM Y
WHERE name LIKE 'Distant Worlds%'
```

Transformierte Anfrage:

```
WITH X AS(
  SELECT *
  FROM tracks NATURAL JOIN albums
)
SELECT *
FROM X
WHERE name LIKE 'Distant Worlds%'
AND length > 250000
```

K03: Kommutativität von Projektion und Verbund – V1

Regel: $\pi_X(\pi_X(r_1) \bowtie r_2) \sqsubseteq_K \pi_X(r_1 \bowtie r_2)$
Invarianten: $attr(X) \subseteq r_1, attr(Y) \subseteq r_1 \bowtie r_2$
Vorbedingungen:

- Untere Ebene L_i: $\{\pi_X\} \not\subseteq O_i, \{\bowtie\} \subseteq O_i$
- Obere Ebene L_{i+1}: $\{\pi_X\} \subseteq O_{i+1}$

Nachbedingungen:

- Untere Ebene L_i: $\{\bowtie\} \subseteq O_i$
- Obere Ebene L_{i+1}: $\{\pi_X\} \subseteq O_{i+1}$

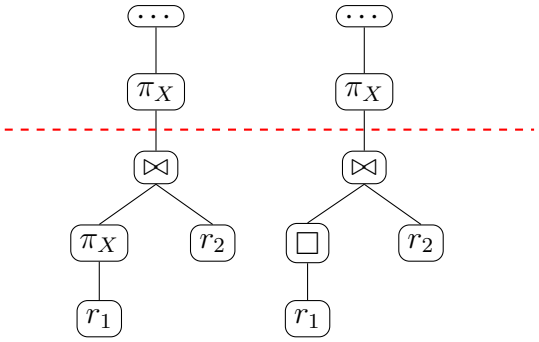


Abbildung C.64: Transformation K03

Originalanfrage:

```
WITH X AS (
  SELECT tid, aid, title
  FROM tracks
),
Y AS (
  SELECT *
  FROM X NATURAL JOIN albums
)
SELECT tid, aid, title
FROM Y
```

Transformierte Anfrage:

```
WITH X AS (
  SELECT *
  FROM tracks
), Y AS (
  SELECT *
  FROM X NATURAL JOIN albums
)
SELECT tid, aid, title
FROM Y
```

K04: Kommutativität von Projektion und Verbund – V2

Regel: $\pi_Y(r_1) \bowtie \pi_Z(r_2) \sqsubseteq_K \pi_X(r_1 \bowtie \pi_Z(r_2))$
Invarianten: $X := Y \cup Z$
Vorbedingungen:

- Untere Ebene L_i: $\{\pi_Y\} \not\subseteq O_i$, $\{\bowtie, \pi_Z\} \subseteq O_i$, $Y \cap Z \neq \emptyset \vee R(r_1) \cap R(r_2) = \emptyset$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: $\{\bowtie, \pi_Z\} \subseteq O_i$
- Obere Ebene L_{i+1}: $\{\pi_X\} \subseteq O_{i+1}$,

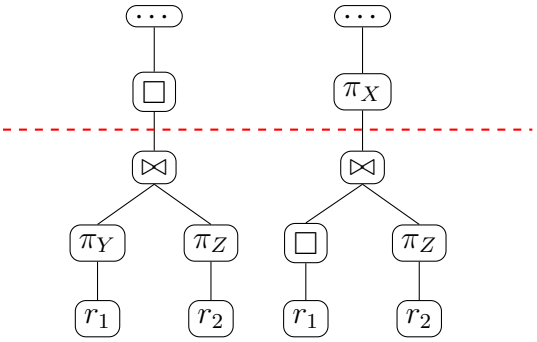


Abbildung C.65: Transformation K04

Originalanfrage:

```
WITH X AS (
  SELECT tid, aid, title
  FROM tracks
), Y AS (
  SELECT aid, name
  FROM albums
), Z AS (
  SELECT *
  FROM X NATURAL JOIN Y
)
SELECT *
FROM Z
```

Transformierte Anfrage:

```
WITH X AS (
  SELECT *
  FROM tracks
), Y AS (
  SELECT aid, name
  FROM albums
), Z AS (
  SELECT *
  FROM X NATURAL JOIN Y
)
SELECT aid, tid, title, name
FROM Z
```

K05: Dominanz der äußeren Projektion

Regel: $\pi_X(\pi_Y(r)) \sqsubseteq_K \pi_X(r)$
Invarianten: $X \subseteq Y$
Vorbedingungen:

- Untere Ebene L_i: $\{\pi_Y\} \not\subseteq O_i$
- Obere Ebene L_{i+1}: $\{\pi_X\} \subseteq O_{i+1}$

Nachbedingungen:

- Untere Ebene L_i: —
- Obere Ebene L_{i+1}: $\{\pi_X\} \subseteq O_{i+1}$

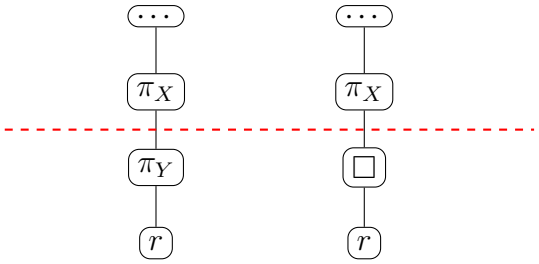


Abbildung C.66: Transformation K05

Originalanfrage:

```
WITH X AS (
  SELECT tid, aid, title
  FROM tracks
)
SELECT aid, title
FROM X
```

Transformierte Anfrage:

```
WITH X AS (
  SELECT *
  FROM tracks
)
SELECT aid, title
FROM X
```

K06: Kommutativität von zwei Selektionen – V1

Regel: $\sigma_X(\sigma_Y(r)) \sqsubseteq_K \sigma_{X \wedge Y}(r)$
Invarianten: —
Vorbedingungen:

- Untere Ebene L_i: $\{\sigma_Y\} \not\subseteq O_i$
- Obere Ebene L_{i+1}: $\{\sigma_X\} \subseteq O_{i+1}$

Nachbedingungen:

- Untere Ebene L_i: —
- Obere Ebene L_{i+1}: $\{\sigma_X, \sigma_Y, \wedge\} \subseteq O_{i+1}$

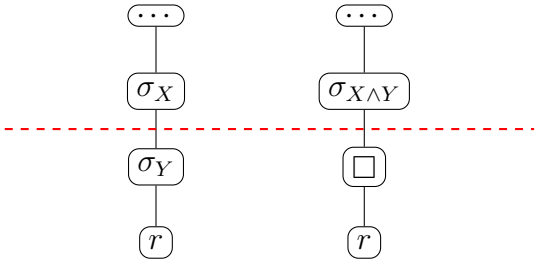


Abbildung C.67: Transformation K06

Originalanfrage:

```
WITH X AS (
  SELECT *
  FROM tracks
  WHERE title LIKE '%song%'
)
SELECT *
FROM X
WHERE length > 250000
```

Transformierte Anfrage:

```
WITH X AS (
  SELECT *
  FROM tracks
)
SELECT *
FROM X
WHERE title LIKE '%song%'
AND length > 250000
```

K07: Kommutativität von zwei Selektionen – V2

Regel: $\sigma_{X \wedge Y}(r) \sqsubseteq_K \sigma_Y(\sigma_X(r))$
Invarianten: —
Vorbedingungen:

- Untere Ebene L_i: $\{\sigma_Y, \wedge\} \not\subseteq O_i$, $\{\sigma_X\} \subseteq O_i$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: $\{\sigma_X\} \subseteq O_i$,
- Obere Ebene L_{i+1}: $\{\sigma_Y\} \subseteq O_{i+1}$,

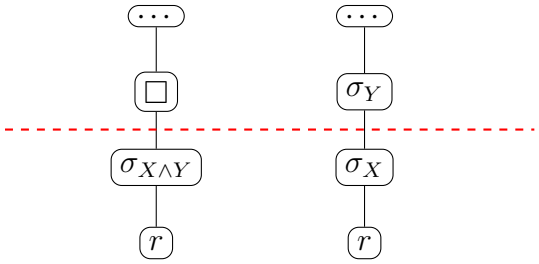


Abbildung C.68: Transformation K07

Originalanfrage:

```
WITH X AS (
  SELECT *
  FROM tracks
  WHERE title LIKE '%song%'
  AND length > 250000
)
SELECT *
FROM X
```

Transformierte Anfrage:

```
WITH X AS (
  SELECT *
  FROM tracks
  WHERE title LIKE '%song%'
)
SELECT *
FROM X
WHERE length > 250000
```

K08: Kommutativität von zwei Selektionen – V3

Regel: $\sigma_X(\sigma_Y(r)) \sqsubseteq_K \sigma_Y(\sigma_X(r))$
Invarianten: —
Vorbedingungen:

- Untere Ebene L_i: $\{\sigma_Y\} \not\subseteq O_i$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: $\{\sigma_X\} \subseteq O_i$
- Obere Ebene L_{i+1}: $\{\sigma_Y\} \subseteq O_{i+1}$

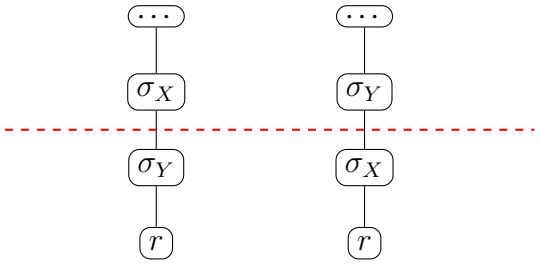


Abbildung C.69: Transformation K08

Originalanfrage:

```
WITH X AS (
  SELECT *
  FROM tracks
  WHERE length > 250000
)
SELECT *
FROM X
WHERE title LIKE '%song%'
```

Transformierte Anfrage:

```
WITH X AS (
  SELECT *
  FROM tracks
  WHERE title LIKE '%song%'
)
SELECT *
FROM X
WHERE length > 250000
```

K09: Kommutativität von Selektion und Projektion – V1

Regel: $\sigma_Y(\pi_X(r)) \sqsubseteq_K \pi_X(\sigma_Y(r))$
Invarianten: —
Vorbedingungen:

- Untere Ebene L_i: $\{\pi_X\} \not\subseteq O_i$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: $\{\sigma_Y\} \subseteq O_i$
- Obere Ebene L_{i+1}: $\{\pi_X\} \subseteq O_{i+1}$

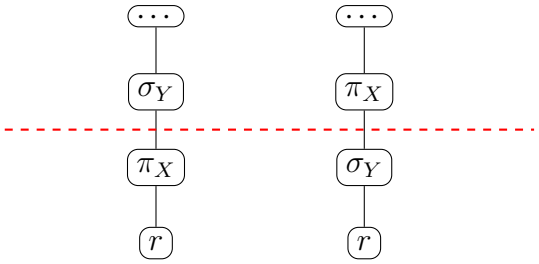


Abbildung C.70: Transformation K09

Originalanfrage:

```
WITH X AS (
  SELECT title, length
  FROM tracks
)
SELECT *
FROM X
WHERE length > 250000
```

Transformierte Anfrage:

```
WITH X AS (
  SELECT *
  FROM tracks
  WHERE length > 250000
)
SELECT title, length
FROM X
```

K10: Kommutativität von Selektion und Projektion – V2

Regel: $\pi_Y(\sigma_X(r)) \sqsubseteq_K \sigma_X(\pi_Y(r))$
Invarianten: $attr(X) \subseteq Y$
Vorbedingungen:

- Untere Ebene L_i: $\{\sigma_X\} \not\subseteq O_i$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: $\{\pi_Y\} \subseteq O_i$
- Obere Ebene L_{i+1}: $\{\sigma_X\} \subseteq O_{i+1}$

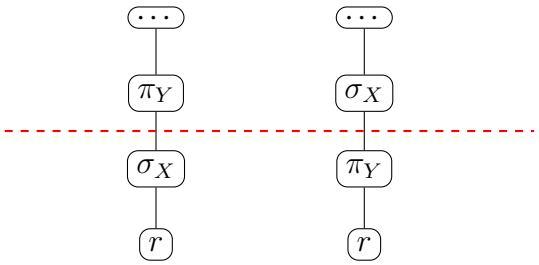


Abbildung C.71: Transformation K10

Originalanfrage:

```
WITH X AS (
  SELECT *
  FROM tracks
  WHERE title LIKE '%song%'
)
SELECT title, length
FROM X
```

Transformierte Anfrage:

```
WITH X AS (
  SELECT title, length
  FROM tracks
)
SELECT *
FROM X
WHERE title LIKE '%song%'
```

K11: Kommutativität von Selektion und Projektion – V3

Regel: $\pi_X(\sigma_Z(\pi_{X,Y}(r))) \sqsubseteq_K \pi_X(\sigma_Z(r))$
Invarianten: $attr(Z) \subseteq X \cup Y$
Vorbedingungen:

- Untere Ebene L_i: $\{\pi_{X,Y}\} \not\subseteq O_i, \{\sigma_Z\} \subseteq O_i$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: $\{\sigma_Z\} \subseteq O_i$
- Obere Ebene L_{i+1}: $\{\pi_X\} \subseteq O_{i+1}$

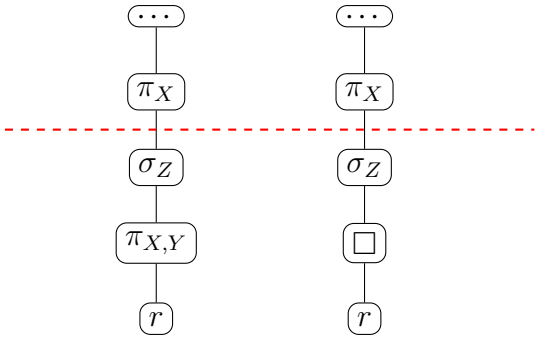


Abbildung C.72: Transformation K11

Originalanfrage:

```
WITH X AS (
  SELECT title, length
  FROM tracks
), Y AS (
  SELECT *
  FROM X
  WHERE length < 250000
)
SELECT title
FROM Y
```

Transformierte Anfrage:

```
WITH X AS (
  SELECT *
  FROM tracks
  WHERE length < 250000
)
SELECT title
FROM X
```

K12: Kommutativität von Selektion und Projektion – V4

Regel: $\pi_X(\sigma_Z(r)) \sqsubseteq_K \pi_X(\sigma_Z(\pi_{X,Y}(r)))$
Invarianten: $Y := attr(Z) \backslash X$
Vorbedingungen:

- Untere Ebene L_i: $\{\sigma_Z\} \not\subseteq O_i$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: $\{\pi_{X,Y}\} \subseteq O_i$
- Obere Ebene L_{i+1}: $\{\pi_X, \sigma_Z\} \subseteq O_{i+1}$

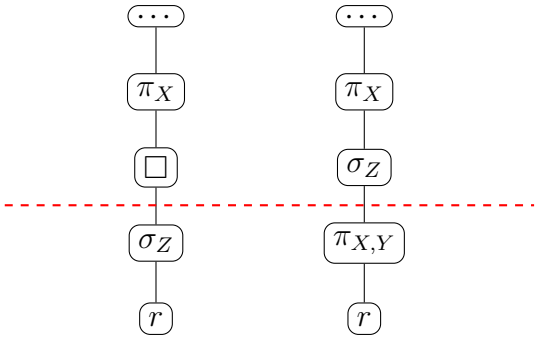


Abbildung C.73: Transformation K12

Originalanfrage:

```
WITH X AS (
  SELECT *
  FROM tracks
  WHERE title LIKE '%song%'
)
SELECT title
FROM X
```

Transformierte Anfrage:

```
WITH X AS (
  SELECT title, length
  FROM tracks
), Y AS (
  SELECT *
  FROM X
  WHERE title LIKE '%song%'
)
SELECT title
FROM Y
```

K13: Kommutativität von Selektion und Mengenvereinigung – V1

Regel: $\sigma_X(r_1 \cup r_2) \sqsubseteq_K \sigma_X(r_1) \cup \sigma_X(r_2)$
Invarianten: $R(r_1) \equiv R(r_2)$
Vorbedingungen:

- Untere Ebene L_i: $\{\cup\} \not\subseteq O_i$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: $\{\sigma_X\} \subseteq O_i$
- Obere Ebene L_{i+1}: $\{\cup\} \subseteq O_{i+1}$

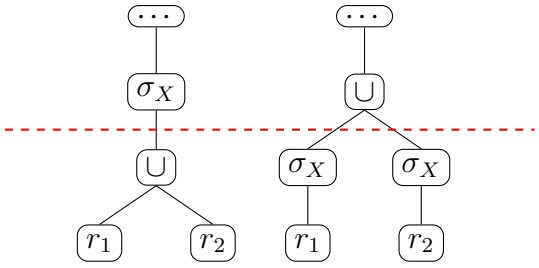


Abbildung C.74: Transformation K13

Originalanfrage:

```
WITH X AS (
  (SELECT name
  FROM composers) UNION
  (SELECT name
  FROM artists)
)
SELECT name
FROM X
WHERE name LIKE '%&%'
```

Transformierte Anfrage:

```
WITH X AS (
  SELECT name
  FROM composers
  WHERE name LIKE '%&%'
), Y AS (
  SELECT name
  FROM artists
  WHERE name LIKE '%&%'
)
SELECT *
FROM X
UNION
SELECT *
FROM Y
```

K14: Kommutativität von Selektion und Mengenvereinigung – V2

Regel: $\sigma_X(r_1) \cup \sigma_X(r_2) \sqsubseteq_K \sigma_X(r_1 \cup r_2)$
Invarianten: $R(r_1) \equiv R(r_2)$
Vorbedingungen:

- Untere Ebene L_i: $\{\sigma_X\} \not\subseteq O_i$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: $\{\cup\} \subseteq O_i$
- Obere Ebene L_{i+1}: $\{\sigma_X\} \subseteq O_{i+1}$

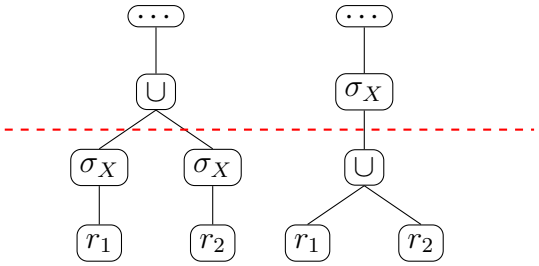


Abbildung C.75: Transformation K14

Originalanfrage:

```
WITH X AS (
  SELECT name
  FROM composers
  WHERE cid < 2500
), Y AS (
  SELECT name
  FROM artists
  WHERE artid < 2500
)
SELECT *
FROM X
UNION
SELECT *
FROM Y
```

Transformierte Anfrage:

```
WITH X AS (
  (SELECT cid, name
  FROM composers)
  UNION
  (SELECT artid, name
  FROM artists)
)
SELECT name
FROM X
WHERE cid < 2500
```

K15: Kommutativität von Selektion und Mengendifferenz – V1

Regel: $\sigma_X(r_1 - r_2) \sqsubseteq_K \sigma_X(r_1) - r_2$
Invarianten: $R(r_1) \equiv R(r_2)$
Vorbedingungen:

- Untere Ebene L_i: $\{-\} \not\subseteq O_i$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: $\{\sigma_X\} \subseteq O_i$
- Obere Ebene L_{i+1}: $\{-\} \subseteq O_{i+1}$

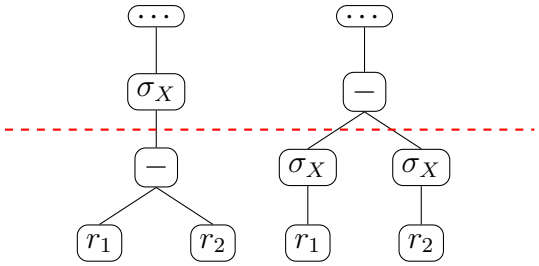


Abbildung C.76: Transformation K15

Originalanfrage:

```
WITH X AS (
   (SELECT name
   FROM composers) EXCEPT
   (SELECT name
   FROM artists)
)
SELECT *
FROM X
WHERE name LIKE '%&%'
```

Transformierte Anfrage:

```
WITH X AS (
   SELECT name
   FROM composers
   WHERE name LIKE '%&%'
), Y AS (
   SELECT name
   FROM artists
   WHERE name LIKE '%&%'
)

SELECT *
FROM X
EXCEPT
SELECT *
FROM Y
```

K16: Kommutativität von Selektion und Mengendifferenz – V2

Regel: $\sigma_X(r_1) - \sigma_X(r_2) \sqsubseteq_K \sigma_X(r_1) - r_2$
Invarianten: $R(r_1) \equiv R(r_2)$
Vorbedingungen:

- Untere Ebene L_i: $\{\sigma_Y\} \not\subseteq O_i$, $\{\sigma_X\} \subseteq O_i$
- Obere Ebene L_{i+1}: $\{-\} \subseteq O_{i+1}$

Nachbedingungen:

- Untere Ebene L_i: $\{\sigma_X\} \subseteq O_i$
- Obere Ebene L_{i+1}: $\{-\} \subseteq O_{i+1}$

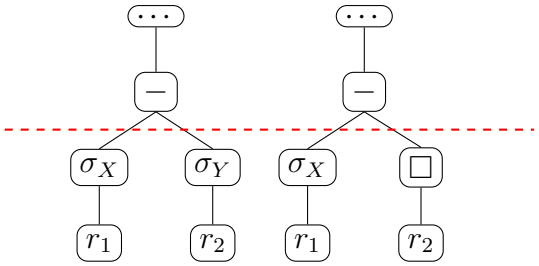


Abbildung C.77: Transformation K16

Originalanfrage:

```
WITH X AS (
  SELECT name
  FROM composers
  WHERE name LIKE '%&%'
), Y AS (
  SELECT name
  FROM artists
  WHERE name LIKE '%&%'
)
SELECT *
FROM X
EXCEPT
SELECT *
FROM Y
```

Transformierte Anfrage:

```
WITH X AS (
  SELECT name
  FROM composers
  WHERE name LIKE '%&%'
), Y AS (
  SELECT name
  FROM artists
)

SELECT *
FROM X
EXCEPT
SELECT *
FROM Y
```

K17: Kommutativität von Selektion und Mengendifferenz – V3

Regel: $\sigma_X(r_1) - r_2 \sqsubseteq_K \sigma_X(r_1 - r_2)$
Invarianten: $R(r_1) \equiv R(r_2)$
Vorbedingungen:

- Untere Ebene L_i: $\{\sigma_X\} \not\subseteq O_i$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: $\{-\} \subseteq O_i$
- Obere Ebene L_{i+1}: $\{\sigma_X\} \subseteq O_{i+1}$

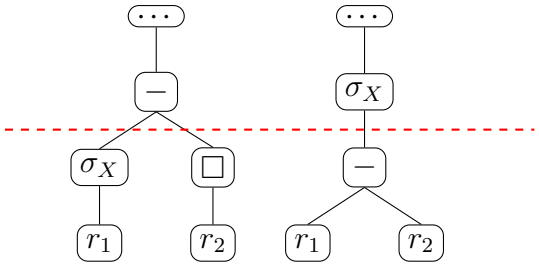


Abbildung C.78: Transformation K17

Originalanfrage:

```
WITH X AS (
  SELECT name
  FROM composers
  WHERE name LIKE '%&%'
), Y AS (
  SELECT name
  FROM artists
)
SELECT *
FROM X
EXCEPT
SELECT *
FROM Y
```

Transformierte Anfrage:

```
WITH X AS (
  SELECT name
  FROM composers
  EXCEPT
  SELECT name
  FROM artists
)

SELECT *
FROM X
WHERE name LIKE '%&%'
```

K18: Kommutativität von Projektion und Mengenvereinigung – V1

Regel: $\pi_X(r_1 \cup r_2) \sqsubseteq_K \pi_X(r_1) \cup \pi_X(r_2)$
Invarianten: $R(r_1) \equiv R(r_2)$
Vorbedingungen:

- Untere Ebene L_i: $\{\cup\} \not\subseteq O_i$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: $\{\pi_X\} \subseteq O_i$
- Obere Ebene L_{i+1}: $\{\cup\} \subseteq O_{i+1}$

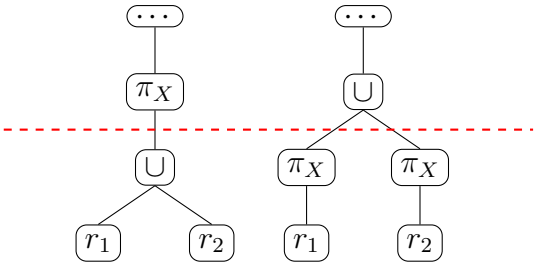


Abbildung C.79: Transformation K18

Originalanfrage:

```
WITH X AS (
  SELECT *
  FROM composers
  UNION
  SELECT *
  FROM artists
)
SELECT DISTINCT name
FROM X
```

Transformierte Anfrage:

```
WITH X AS (
  SELECT name
  FROM composers
), Y AS (
  SELECT name
  FROM artists
)

SELECT *
FROM X
UNION
SELECT *
FROM Y
```

K19: Kommutativität von Projektion und Mengenvereinigung – V2

Regel: $\pi_X(r_1) \cup \pi_X(r_2) \sqsubseteq_K \pi_X(r_1 \cup r_2)$
Invarianten: $R(r_1) \equiv R(r_2)$
Vorbedingungen:

- Untere Ebene L_i: $\{\pi_X\} \not\subseteq O_i$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: $\{\cup\} \subseteq O_i$
- Obere Ebene L_{i+1}: $\{\pi_X\} \subseteq O_{i+1}$

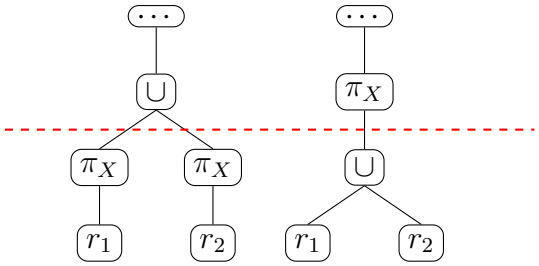


Abbildung C.80: Transformation K19

Originalanfrage:

```
WITH X AS (
  SELECT name
  FROM composers
), Y AS (
  SELECT name
  FROM artists
)
SELECT *
FROM X
UNION
SELECT *
FROM Y
```

Transformierte Anfrage:

```
WITH X AS (
  SELECT *
  FROM composers
  UNION
  SELECT *
  FROM artists
)

SELECT DISTINCT name
FROM X
```

K20: Distributivität von Mengenvereinigung und Verbund – V1

Regel: $(r_1 \cup r_2) \bowtie (s_1 \cup s_2) \sqsubseteq_K (r_1 \bowtie s_1) \cup (r_1 \bowtie s_2) \cup (r_2 \bowtie s_1) \cup (r_2 \bowtie s_2)$
Invarianten: $R(r_1) \equiv R(r_2)$, $R(s_1) \equiv R(s_2)$
Vorbedingungen:

- Untere Ebene L_i: $\{\cup\} \not\subseteq O_i$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: $\{\bowtie\} \subseteq O_i$
- Obere Ebene L_{i+1}: $\{\cup\} \subseteq O_{i+1}$

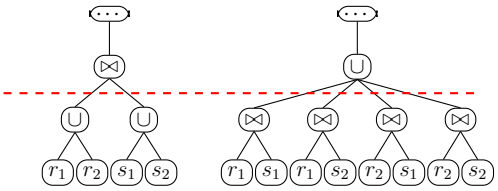


Abbildung C.81: Transformation K20

Originalanfrage:

```
WITH X AS (
  (SELECT artid, name
  FROM artists
  WHERE name LIKE '%&%')
  UNION
  (SELECT artid, name
  FROM artists
  WHERE name LIKE '%Nob%')
), Y AS (
  (SELECT tid, title, artid
  FROM tracks
  WHERE length > 250000)
  UNION
  (SELECT tid, title, artid
  FROM tracks
  WHERE title LIKE '%song%')
)
SELECT *
FROM X NATURAL JOIN Y
```

Transformierte Anfrage:

```
WITH F1 AS (
  SELECT *
  FROM
  (SELECT artid, name
  FROM artists
  WHERE name LIKE '%&%') AS F1X
  NATURAL JOIN
  (SELECT tid, title, artid
  FROM tracks
  WHERE length > 250000) AS F1Y
), F2 AS (
  SELECT *
  FROM
  (SELECT artid, name
  FROM artists
  WHERE name LIKE '%&%') AS F2X
  NATURAL JOIN
  (SELECT tid, title, artid
  FROM tracks
  WHERE title LIKE '%song%') AS F2Y
), F3 AS (
  SELECT *
  FROM
  (SELECT artid, name
  FROM artists
  WHERE name LIKE '%Nob%') AS F3X
  NATURAL JOIN
  (SELECT tid, title, artid
  FROM tracks
  WHERE length > 250000) AS F3Y
), F4 AS (
  SELECT *
  FROM
  (SELECT artid, name
  FROM artists
  WHERE name LIKE '%Nob%') AS F4X
  NATURAL JOIN
  (SELECT tid, title, artid
  FROM tracks
  WHERE title LIKE '%song%') AS F4Y
)
SELECT * FROM F1 UNION SELECT * FROM F2
UNION
SELECT * FROM F3 UNION SELECT * FROM F4
```

K21: Distributivität von Mengenvereinigung und Verbund – V2

Regel: $(r_1 \bowtie s_1) \cup (r_1 \bowtie s_2) \cup (r_2 \bowtie s_1) \cup (r_2 \bowtie s_2) \sqsubseteq_K (r_1 \cup r_2) \bowtie (s_1 \cup s_2)$
Invarianten: $R(r_1) \equiv R(r_2)$, $R(s_1) \equiv R(s_2)$
Vorbedingungen:

- Untere Ebene L_i: $\{\bowtie\} \not\subseteq O_i$
- Obere Ebene L_{i+1}: —

Nachbedingungen:

- Untere Ebene L_i: $\{\cup\} \subseteq O_i$
- Obere Ebene L_{i+1}: $\{\bowtie\} \subseteq O_{i+1}$

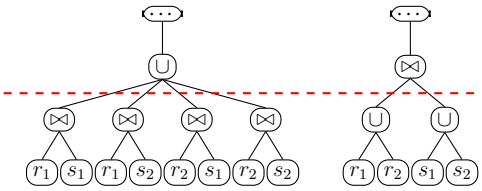


Abbildung C.82: Transformation K21

Transformierte Anfrage:

```
WITH X AS (
   (SELECT artid, name
   FROM artists
   WHERE name LIKE '%&%')
   UNION
   (SELECT artid, name
   FROM artists
   WHERE name LIKE '%Nob%')
), Y AS (
   (SELECT tid, title, artid
   FROM tracks
   WHERE length > 250000)
   UNION
   (SELECT tid, title, artid
   FROM tracks
   WHERE title LIKE '%song%')
)
SELECT *
FROM X NATURAL JOIN Y
```

Originalanfrage:

```
WITH F1 AS (
   SELECT *
   FROM
   (SELECT artid, name
   FROM artists
   WHERE name LIKE '%&%') AS F1X
   NATURAL JOIN
   (SELECT tid, title, artid
   FROM tracks
   WHERE length > 250000) AS F1Y
), F2 AS (
   SELECT *
   FROM
   (SELECT artid, name
   FROM artists
   WHERE name LIKE '%&%') AS F2X
   NATURAL JOIN
   (SELECT tid, title, artid
   FROM tracks
   WHERE title LIKE '%song%') AS F2Y
), F3 AS (
   SELECT *
   FROM
   (SELECT artid, name
   FROM artists
   WHERE name LIKE '%Nob%') AS F3X
   NATURAL JOIN
   (SELECT tid, title, artid
   FROM tracks
   WHERE length > 250000) AS F3Y
), F4 AS (
   SELECT *
   FROM
   (SELECT artid, name
   FROM artists
   WHERE name LIKE '%Nob%') AS F4X
   NATURAL JOIN
   (SELECT tid, title, artid
   FROM tracks
   WHERE title LIKE '%song%') AS F4Y
)
SELECT * FROM F1 UNION SELECT * FROM F2
UNION
SELECT * FROM F3 UNION SELECT * FROM F4
```

Anhang D

Komplexere Beispiele

In diesem Anhang werden Beispiele aufgeführt, welche aufgrund ihres Umfangs den Lesefluss der Konzeptkapitel gestört hätten. Abschnitt D.1 geht auf weiterführende Zerlegungen komplexer Aggregatfunktionen am Beispiel der Korrelation und linearen Regression ein. Anschließend wird in Abschnitt D.2 gezeigt, wie eine komplexe Anfrage schrittweise durch die Anwendung mehrerer Regeln transformiert wird.

D.1 Korrelation und Regression als Anfragebäume

Das folgende Beispiel zeigt die Zerlegung der Aggregatfunktionen *Korrelation* und *Regression* als Anfragebäume und in der Meta-Algebra, welche in Abschnitt 6.1.3 eingeführt wurde. In dem Beispiel wird zudem auf Probleme bei Berechnungen mit *shared aggregation* (siehe auch Abschnitt 5.5) eingegangen.

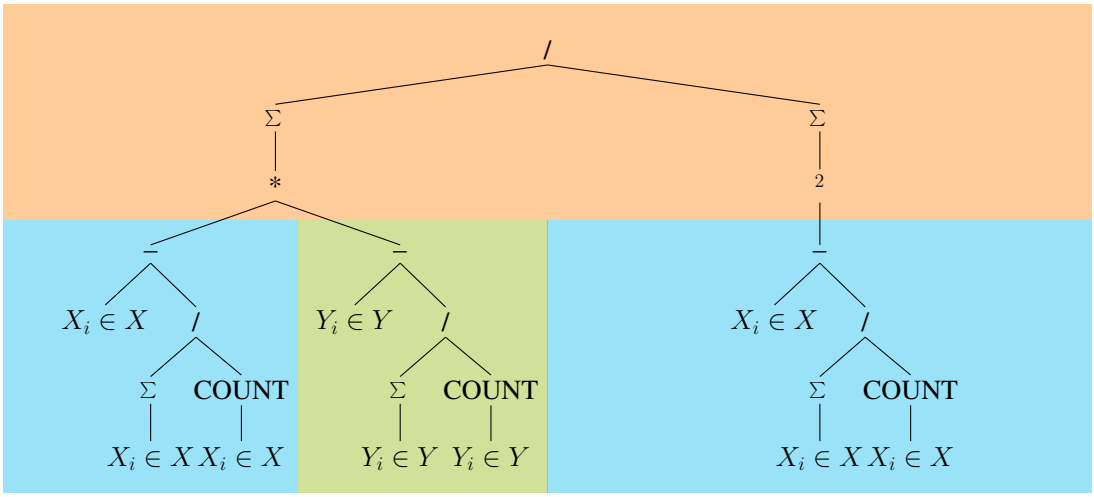
Beispiel: Zerlegung komplexerer Aggregatfunktionen

Statistische Auswertungen waren bis vor wenigen Jahren ein herausstellendes Merkmal von Programmiersprachen wie R[a] oder statistischer Softwareprodukte wie der SPSS-Suite[b] von IBM. Durch die Erweiterungen des SQL-Standards um weitere Aggregatfunktionen sowie dem *Row Pattern Matching*, welches beispielsweise zur Erkennung von Ausreißern genutzt werden kann [Sch18], haben relationale Datenbanksysteme diese Lücke geschlossen.

Zu den gängigsten bi- bzw. multivariaten Statistikverfahren gehören die Korrelations- und Regressionsanalyse. Bei der Korrelationsanalyse werden Abhängigkeiten zwischen zwei oder mehreren Attributen anhand ihrer Attributwerte untersucht. Mit Hilfe der Regressionsanalyse wird anschließend die Form der Abhängigkeit zwischen einem abhängigen und mehreren unabhängigen Attributen bestimmt. Die Regressionsgerade lässt sich wie folgt berechnen:

$$reg\_slope(X,Y) := \frac{\sum_{i=1}^{n}\left(X_i - \overline{X}\right) * \left(Y_i - \overline{Y}\right)}{\sum_{i=1}^{n}\left(X_i - \overline{X}\right)^2}. \tag{D.1}$$

In der folgenden Abbildung wird die Berechnung des Anstiegs einer linearen Regression, $reg\_slope(X,Y)$, über zwei numerischen Attributen, X und Y, als Anfragebaum visualisiert.



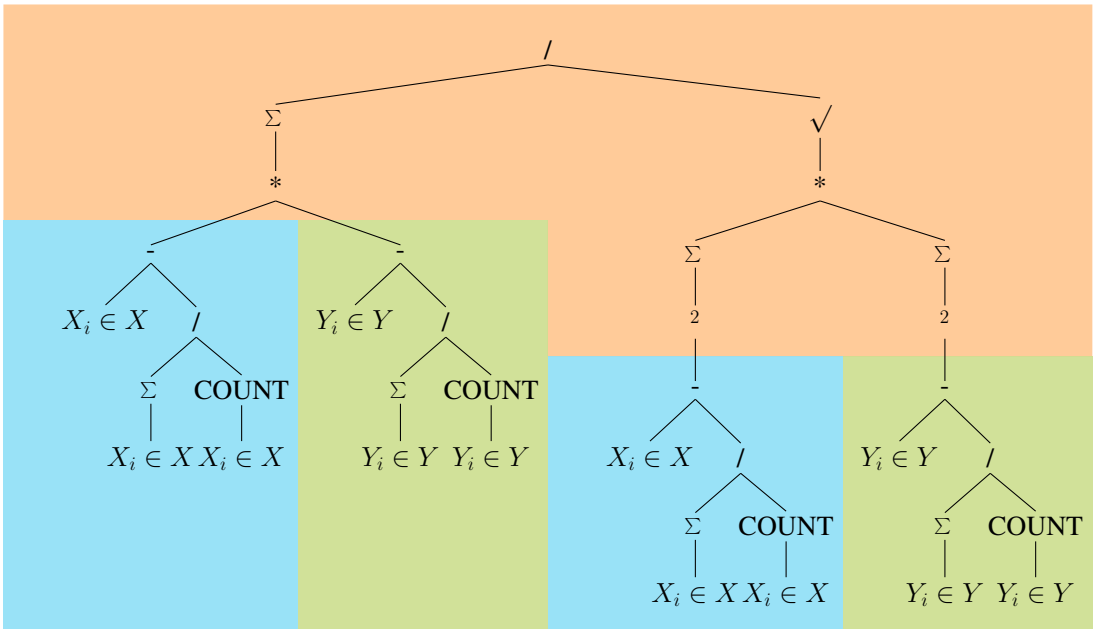
Für die initiale Verteilung wird die Anfrage in zwei Anfragefragmente aufgeteilt: Die ersten Teilanfragen berechnen die Differenzen zwischen den einzelnen X-Werten und deren Durchschnitt $\overline{X}$ (in der obigen Abbildung blau hinterlegt). Gleiches gilt für die Y-Werte (in der obigen Abbildung grün hinterlegt). Da $\left(X_i - \overline{X}\right)$ an zwei Stellen innerhalb der Berechnung benötigt wird, erscheint es doppelt im Anfragebaum[c]. Der Rest der Anfrage (paarweise Multiplikation, Summenbildung und Division) wird in der Cloud (in der obigen Abbildung orange hinterlegt) ausgeführt. In der verwendeten Meta-Algebra wird dies folgendermaßen dargestellt:

- $L_{Sensor}^{Regression} := (\{\sum, COUNT, -\}, \{X, Y\})$
- $L_{Cloud}^{Regression} := (\{\sum, *, ^2, /\}, \{\left(X_i - \overline{X}\right), \left(Y_i - \overline{Y}\right)\})$

Die Korrelation wird wie folgt berechnet:

$$corr(X,Y) := \frac{\sum_{i=1}^{n}\left(X_i - \overline{X}\right) * \left(Y_i - \overline{Y}\right)}{\sqrt{\sum_{i=1}^{n}\left(X_i - \overline{X}\right)^2 * \sum_{i=1}^{n}\left(Y_i - \overline{Y}\right)^2}} \tag{D.2}$$

Die folgende Abbildung zeigt, wie eine Korrelationsanalyse, $corr(X,Y)$, über zwei nummerische Attributen, X und Y, als Anfragebaum dargestellt wird. Die Verteilung der Anfrage erfolgt dabei analog zu der Verteilung der Regressionsanalyse.



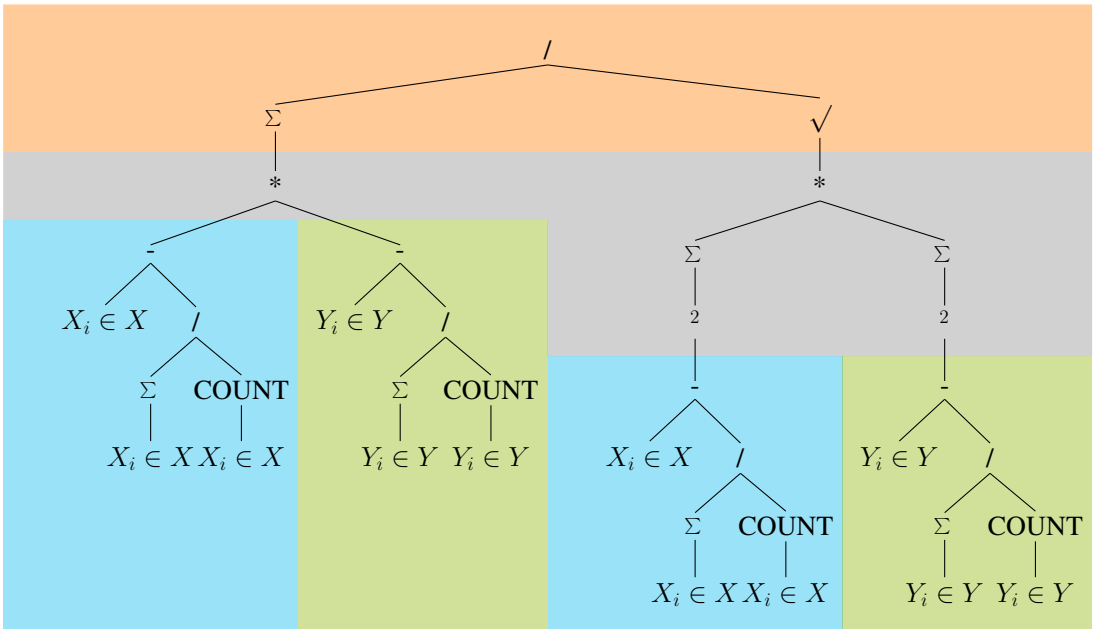
Für die Korrelation werden zwei Datenmengen benötigt: alle X- und alle Y-Werte. Die benötigten Operatoren sind die Division bzw. die Multiplikation, das Quadrieren bzw. das Wurzelziehen, die Addition und die Subtraktion sowie als Aggregatfunktionen die Summe, der Durchschnitt und die Anzahl der Werte. In der verwendeten Meta-Algebra wird dies folgendermaßen dargestellt:

- $L_{Sensor}^{Korrelation} := (\{\sum, COUNT, -\}, \{X, Y\})$
- $L_{Cloud}^{Korrelation} := (\{\sum, *, ^2, /, \sqrt{}\}, \{\left(X_i - \overline{X}\right), \left(Y_i - \overline{Y}\right)\})$

Für die Korrelationsanalyse ist es nicht ausreichend, dass die Anfragefragmente so gewählt werden, dass die Differenzen zwischen den X-Werten und dem arithmetischem Durchschnitt der X-Werte (in den Abbildungen als blau hinterlegter Anteil zu erkennen) durch die Sensoren vorberechnet werden und der Rest der Auswertung in der Cloud erfolgt. Gleiches gilt für die Y-Werte (grüner Hintergrund). Grund hierfür sind die unzureichenden Vorberechnungen der Aggregate, da

$$D_{Cloud}^{Korrelation} \equiv D_{Cloud}^{Regression} \tag{D.3}$$

gilt. Die Zwischenergebnisse können somit dazu genutzt werden, um eine zusätzliche Regressionsanalyse durchzuführen. Widerspricht dies dem zuvor festgelegten Zweck des Assistenzsystems, so besteht die Gefahr einer Datenschutzverletzung durch Datenmissbrauch. Die folgende Abbildung zeigt eine Modifizierung des Anfragebaums:



Um zu verhindern, dass die Daten in dieser voraggregierten Form in die Cloud geschickt werden, wird eine zusätzliche Ebene, $Edge$, zwischen den Sensoren und der Cloud eingefügt, die beispielsweise aus lokalen Rechnern und Routern als Gateway-Knoten besteht. Für die neue Ebene nehmen wir an, dass sie über ausreichend Anfragekapazitäten verfügt, um die (x,y)-Paare im Zähler der Korrelationsfunktion miteinander zu multiplizieren und um das Produkt der aufsummierten Quadrate der (x,y)-Paare im Nenner zu berechnen (siehe grau hinterlegter Anteil in der obigen Abbildung).

Die verbleibende Analyse, d. h., die Summe im Zähler der Korrelationsfunktion, die Berechnung der Wurzel im Nenner und die abschließende Division, wird in der Cloud durchgeführt. In der verwendeten Meta-Algebra wird dies folgendermaßen realisiert:

- $L_{Sensor}^{Korrelation2} := (\{\sum, COUNT, -\}, \{X, Y\})$
- $L_{Edge}^{Korrelation2} := (\{\sum, *, ^2\}, \{X_i^{diff} := \left(X_i - \overline{X}\right), Y_i^{diff} := \left(Y_i - \overline{Y}\right)\})$
- $L_{Cloud}^{Korrelation2} := (\{\sum, *, ^2, /, \sqrt{}\}, \{\left(X_i^{diff} * Y_i^{diff}\right), \left(\sum (X_i^{diff})^2 * \sum (Y_i^{diff})^2\right)\})$

[a] `https://www.r-project.org/`, zuletzt aufgerufen am 05.01.2022
[b] `https://www.ibm.com/de-de/products/spss-modeler`, zuletzt aufgerufen am 05.01.2022
[c] Sofern durch den lokalen Optimierer die Äquivalenz der Teilausdrücke erkannt wird, braucht das Zwischenergebnis nicht doppelt berechnet zu werden.

D.2 Relationale Anfrage auf dem Amarok-Datensatz

In diesem Abschnitt wird gezeigt, wie anhand des entwickelten Regelsystems eine komplexere Anfrage schrittweise transformiert wird. Für das Beispiel gehen wir von drei aufeinander aufbauenden Ebenen, L_1 bis L_3, aus, die wie folgt definiert sind:

- $L_1 := (\{\pi, \sigma, \gamma, \bowtie, =_c, MIN, AVG\}, \{r(Albums), r(Composers), r(Tracks)\})$
- $L_2 := (O_1 \cup \{SUM, MAX <, <, \leq, =, \geq, >\}, \emptyset)$
- $L_3 := (O_2 \cup \{\neq\}, \emptyset)$

Beispiel: Komplexe Beispielanfrage

Als komplexe Beispielanfrage gehen wir von folgender Fragestellung aus:

> *Gebe den Namen des Komponisten und des dazugehörigen Albums aus, dessen kürzester Track auf dem jeweiligen Album genau zwölf Sekunden lang ist und dessen längster Track mindestens zehn Minuten dauert.*

Mittels SQL lässt sich die obige Problemstellung wie folgt lösen:

```
1 SELECT CName, AName
2 FROM Composers JOIN Tracks USING (cid) JOIN Albums USING (aid)
3 WHERE CName != ''
4 GROUP BY aid, CName, AName
5 HAVING min(length) = 12*1000
6 AND max(length) >= 10*60*1000
```

Die Anfrage enthält somit zwei Verbund-Operatoren, eine Gruppierung sowie zwei Selektionsbedingungen auf dieser Gruppierung. Zusätzlich wurde eine weitere Selektion auf den Namen der Komponisten eingefügt, damit keine leeren Zeichenketten zurückgegeben werden. Abweichend vom laufenden Beispiel (siehe Abbildung 2.2) wurden die Attributnamen für Komponisten und Alben mit `CName` und `AName` zur Verbesserung der Lesbarkeit und Vereinfachung der Anfrage umbenannt.

Ziel dieses Abschnittes ist es, zu zeigen, wie die Anfrage auf die Ebenen L_1 bis L_3 aufgeteilt werden kann, sodass möglichst viele Daten bereits auf L_1 vorselektiert werden. Dazu wird insbesondere der Nutzen von Anfragetransformationen mit funktionalen und Inklusionsabhängigkeiten gezeigt. Als Ausdruck der Relationenalgebra sieht die Anfrage ohne vorherige Optimierung wie folgt aus:

$$
\begin{array}{l}
1\ \pi_{CName,AName}\ (\\
2\quad \sigma_{min(length)=12*1000 \wedge max(length) \geq 10*60*1000}\ (\\
3\quad\quad \gamma_{min(length),max(length);aid,CName,AName}\ (\\
4\quad\quad\quad \sigma_{CName \neq ''}\ (\\
5\quad\quad\quad\quad r(Composers) \bowtie r(Tracks) \bowtie r(Albums) \\
6\quad)\ )\ )\ )
\end{array}
$$

Grafisch wird dieser Ausdruck als Anfragebaum in Abbildung D.1 visualisiert.

D.2.1 Anwendung klassischer Optimierungsregeln

Durch die Anwendung klassischer Optimierungsregeln lässt sich die Anfrage wie folgt optimieren: Wird die Kommutativität von Selektion und Verbund ausgenutzt, lässt sich die Selektion auf den Namen des Komponisten (in Abbildung D.2 grün markiert) mit dem darunter liegenden Verbund-Operator tauschen. Zusätzlich werden auf den Basisrelationen weitere Projektionen eingefügt (in Abbildung D.2 bernsteinfarben markiert), damit nur die zwingend notwendigen Attribute für die Ausführung der nachfolgenden Operatoren genutzt werden.

D.2.2 Optimierung unter Ausnutzung funktionaler Abhängigkeiten und Inklusionsabhängigkeiten

Weiterhin lassen sich auf Basis der Arbeiten von Türker [Tür99] und Kaseler [Kas21] weitere Anfragetransformation erstellen, die auf der Existenz von funktionalen Abhängigkeiten und Inklusionsabhängigkeiten in der Datenbank beruhen. Zwischen den Attributen `cid` und `CName` in der Relation `Composers` besteht jeweils eine funktionale Abhängigkeit in beide Richtungen:

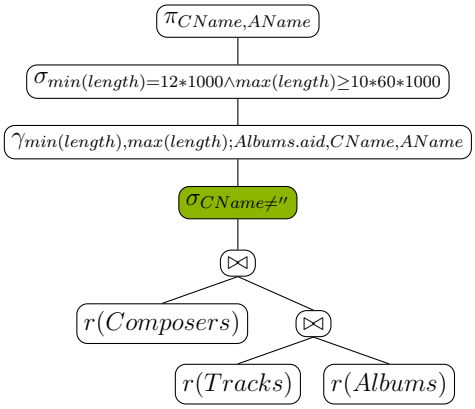


Abbildung D.1: Unoptimierte Anfrage als Algebrabaum. Grün hervorgehoben wurde die Selektionsbedingung $\sigma_{CName \neq ''}$, welche im ersten Schritt durch die Anwendung klassischer Optimierungsregeln tiefer im Baum platziert wird.

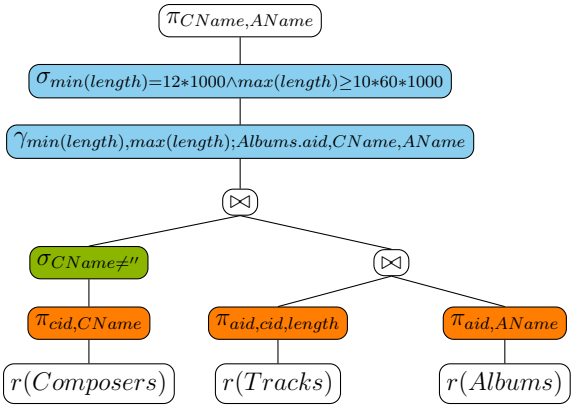


Abbildung D.2: Anfragebaum nach Anwendung der Regeln K2 und K12. Die bernsteinfarben markierten Knoten stellen zusätzliche Projektionen dar, die zusätzlich zum Verschieben der Selektionsbedingung (grün) eingefügt wurden. Für den nächsten Schritt werden die blau hervorgehobenen Operatoren, Gruppierung und Selektion, verschoben.

$$Composers.cid \rightarrow Composers.CName$$
$$\texttt{bzw.} \tag{D.4}$$
$$Composers.CName \rightarrow Composers.cid;$$

Beide Attribute bestimmen sich gegenseitig (jeder Komponist in der Beispieldatenbank hat seinen Namen als nicht-ausgezeichneten alternativen Schlüssel), wodurch die Gruppierung γ wie folgt ersetzt werden kann:

$$\begin{gathered}\gamma_{min(length),max(length);Albums.aid,Composers.CName,Albums.AName}(\dots) \\ \downarrow \\ \gamma_{min(length),max(length);Albums.aid,Composers.cid,Albums.AName}(\dots)\end{gathered} \tag{D.5}$$

Für die Attribute `aid` und `AName` der Relation `Albums` gilt lediglich die funktionale Abhängigkeit

$$Albums.aid \rightarrow Albums.AName, \tag{D.6}$$

nicht jedoch

$$Albums.AName \rightarrow Albums.aid, \tag{D.7}$$

da der gleiche Albumname für einige Alben mehrfach vergeben ist. Dadurch ist hier keine Ersetzung in der Gruppierung möglich. Zusätzlich besteht durch die vorherige Transformation eine Inklusionsabhängigkeit zwischen dem Attribut `cid` aus der Relation `Composers` und dem Attribut `cid` aus der Relation `Tracks`:

$$Tracks.cid \subseteq Composers.cid. \tag{D.8}$$

Da durch den natürlichen Verbund zwischen `Tracks` und `Composers` die unbeteiligten Komponisten entfernt werden, ist die Menge der Komponisten identisch:

$$\pi_{cid}(Tracks \bowtie Composers) \equiv Tracks.cid, \tag{D.9}$$

wodurch die Gruppierung auf dem Attribut `Tracks.cid` statt auf `Composers.cid` durchgeführt werden kann:

$$\begin{gathered}\gamma_{min(length),max(length);Albums.aid,Composers.cid,Albums.AName}(\dots) \\ \downarrow \\ \gamma_{min(length),max(length);Albums.aid,Tracks.cid,Albums.AName}(\dots)\end{gathered} \tag{D.10}$$

Da die Relation `Composers` an dieser Stelle nicht mehr für die Gruppierung benötigt wird, kann die Gruppierung, inklusive der nachfolgenden Selektion, mit dem ersten Verbund vertauscht werden (blaue Knoten in den Abbildungen D.2 und D.3). Zusätzlich wird nach der Selektion eine weitere Projektion auf den Attributen `tracks.cid` und `AName` eingefügt, da die weiteren Attribute für den Rest der Anfrage nicht mehr benötigt werden (bernsteinfarbener Knoten in Abbildung D.3).

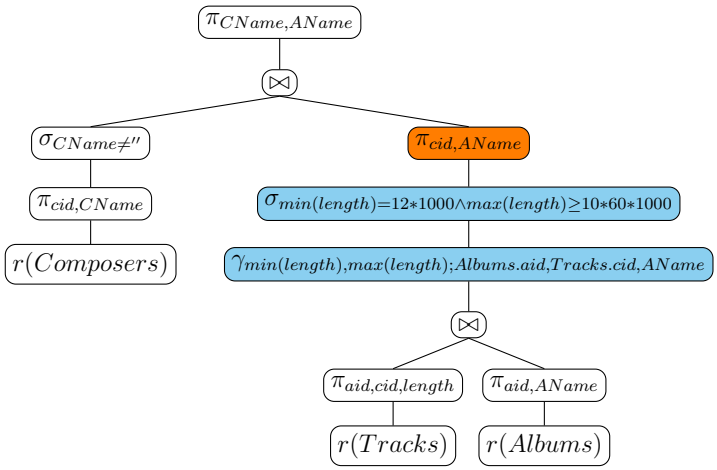


Abbildung D.3: Anfragebaum nach Anwendung der funktionalen Abhängigkeit $Composers.CName \rightarrow Composers.cid$ sowie der Inklusionsabhängigkeit $Tracks.cid \subseteq Composers.cid$. Die betroffenen Knoten sind blau markiert.

Da eine funktionale Abhängigkeit zwischen der ID des Albums und dessen Titel besteht ($Albums.aid \rightarrow Albums.AName$), ist der Name des Albums für die Gruppierung vernachlässigbar. Jedoch muss der Titel nach

dem Verbund wieder hinzugefügt werden, da der Titel für die abschließende Projektion benötigt wird. Dies führt somit zu einer Reduzierung der Gruppierung:

$$\gamma_{min(length),max(length);Albums.aid,Tracks.cid,AName}(\dots) \\ \downarrow \\ \gamma_{min(length),max(length);Albums.aid,Tracks.cid}(\dots) \tag{D.11}$$

Durch die bestehende Inklusionsabhängigkeit aufgrund der Fremdschlüsselbeziehung zwischen der ID des Albums in den Relationen `Albums` und `Tracks` lässt sich `Albums.aid` durch `Tracks.aid` in der Gruppierung ersetzen:

$$\gamma_{min(length),max(length);Albums.aid,Tracks.cid}(\dots) \\ \downarrow \\ \gamma_{min(length),max(length);Tracks.aid,Tracks.cid}(\dots) \tag{D.12}$$

Anschließend lassen sich die Gruppierung und die darauf folgende Selektion mit dem zweiten Verbund vertauschen. Dieser Transformationsschritt ist in Abbildung D.4 dargestellt.

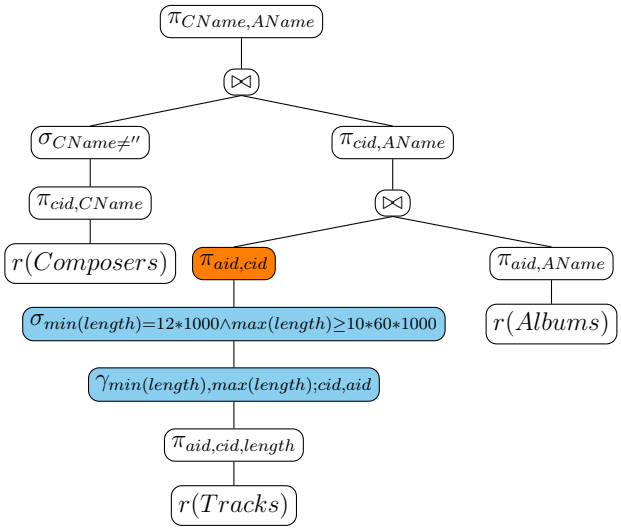


Abbildung D.4: Anfragebaum nach Anwendung der funktionalen Abhängigkeit $Albums.aid \rightarrow Albums.AName$ sowie der Inklusionsabhängigkeit $Tracks.aid \subseteq Albums.aid$. Die betroffenen Knoten sind blau markiert.

Für die optimale Verteilung der Anfrage muss im nächsten Schritt die Gruppenselektion in zwei einzelne Selektionsbedingungen aufgespalten werden. Dies wird durch die Transformationsregel `K7` (Kommutativität von zwei Selektionen – V2) realisiert. Das Ergebnis der Regelanwendung ist in Abbildung D.5 visualisiert.

Durch die Aufteilung der Selektionsprädikate kann das Prädikat $min(length) = 12 * 1000$ direkt auf der Ebene L_1 ausgeführt werden, da das Vergleichsprädikat $=_c$ zusammen mit der verwendeten Aggregatfunktion min in O_1 enthalten ist (siehe Abbildung D.6; kräftigeres blau). Die zweite Selektionsbedingung, $max(length)$ $>=$ $10 * 60 * 1000$, kann aufgrund der fehlenden Unterstützung von $\geq$ und max nicht auf der Ebene L_1 ausgeführt werden und wird somit weiter oben im Anfragebaum ausgeführt. Um dies zu realisieren, muss nach der Selektion auf dem Minimum das Attribut `length` mit dem Zwischenergebnis verknüpft werden (siehe Abbildung D.6; moccafarbene Knoten). Nach dem Verbund kann die erneute Gruppierung zusammen mit der fehlenden Selektion (grau-blauer Knoten in Abbildung D.6) ausgeführt werden.

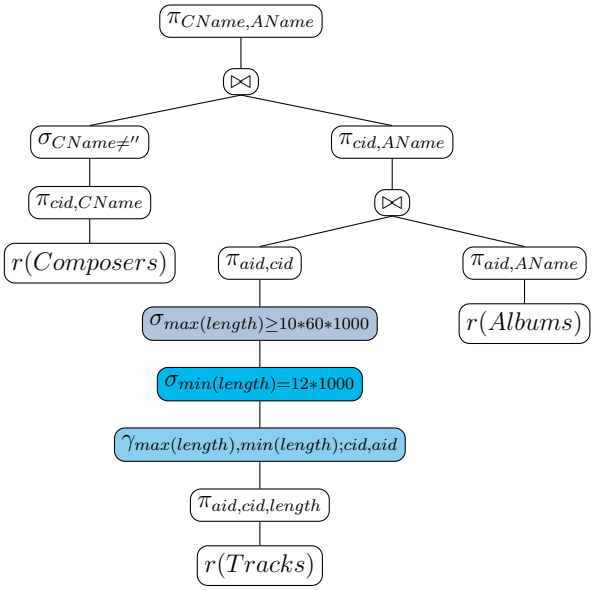


Abbildung D.5: Anfragebaum nach Zerlegung der Gruppenselektion. Die beiden Selektionsbedingungen sind zur besseren Unterscheidbarkeit in zwei verschiedenen Blautönen dargestellt.

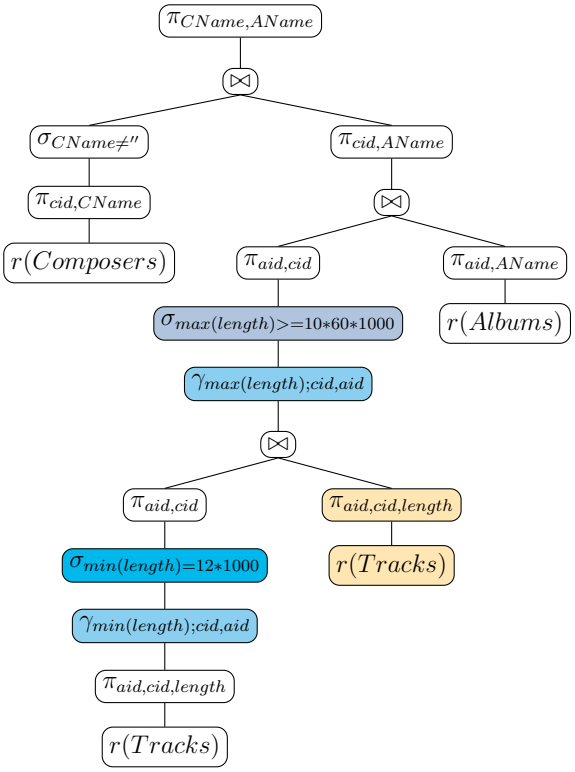


Abbildung D.6: Anfragebaum nach Auftrennung und Verteilung der Gruppenselektion.

Der folgende Quellcodeausschnitt zeigt die SQL-Anfrage mit den entstandenen Anfragefragmenten:

```
WITH Q1_1 AS (
  SELECT *
  FROM (
    SELECT aid, cid
    FROM tracks
    GROUP BY aid, cid
    HAVING min(length) = 12*1000
  ) AS sub1 NATURAL JOIN (
    SELECT aid, cid, length
    FROM tracks
  ) AS sub2
), Q1_2 AS (
  SELECT cid, name AS CName
  FROM composers
), Q1_3 AS (
  SELECT aid, name AS AName
  FROM albums
), Q2 AS (
  SELECT cid, AName
  FROM Q1_3 NATURAL JOIN (
    SELECT aid, cid
    FROM Q1_1
    GROUP BY cid, aid
    HAVING max(length) >= 10*60*1000
  ) AS sub3
), Q3 AS (
  SELECT CName, AName
  FROM Q1_2 NATURAL JOIN Q2
  WHERE CName != ''
)
SELECT *
FROM Q3
```

Zur besseren Visualisierung wurden die Ebenen L_1, L_2 und L_3 in Abbildung D.7 farbig markiert. Die einzelnen Fragmente (Q_{1_1} bis Q_3) aus der obigen SQL-Anfrage wurden zusätzlich an deren Wurzelknoten gekennzeichnet.

Bezüglich der Ebenen L_i ergibt sich somit folgende Aufteilung:

- $L_1 := (O_1, \{r(Albums), r(Composers), r(Tracks)\})$
- $L_2 := (O_2, \{$
  - $D_{1\_1} = \pi_{aid,cid}(\sigma_{min(length)=12*1000}(\gamma_{min(length);aid,cid}(r(Tracks)))) \bowtie \pi_{aid,cid,length}(r(Tracks))$,
  - $D_{1\_2} = \pi_{cid,CName}(r(Composers))$,
  - $D_{1\_3} = \pi_{aid,AName}(r(Albums))$

  $\})$
- $L_3 := (O_3, \{$
  - D_{1_2},
  - $D_2 = \pi_{cid,AName}((\pi_{aid,cid}(\sigma_{max(length)\geq 10*60*1000}(\gamma_{max(length);cid,aid}(D_2)))) \bowtie (D_{1\_2}))$

  $\})$

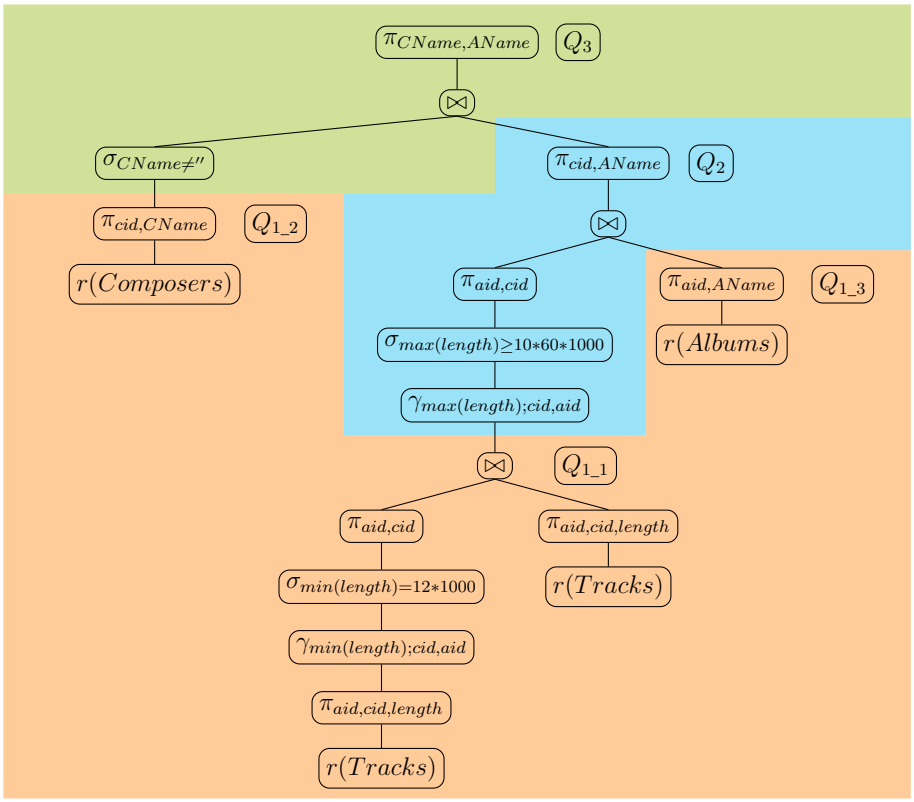


Abbildung D.7: Vollständig transformierte Anfrage mit eingezeichneter Anfrageverteilung. Der orangefarbene Anteil enthält die Basisrelationen sowie alle unterstützten Operatoren der untersten Ebene. Im blauen Teilbaum wird zusätzlich die Gruppenselektion mit der Aggregatfunktion max unterstützt. Auf der obersten Ebene (grün hinterlegt) werden alle Operatoren, einschließlich dem verbleibenden Test auf Ungleichheit, unterstützt.

Die einzelnen Anfragefragmente Q_i können direkt auf der dazugehörigen Ebene L_i ausgeführt werden. Sofern vom DBMS unterstützt, kann jedes einzelne Fragment zuvor lokal optimiert werden. Dies beeinflusst jedoch nicht den kompletten Ausführungsplan und die dahinterstehende Verteilung der Anfragefragmente.

D.2.3 Ausblick: Nesten und Entnesten

Als weitere Optimierung kann der Verbund in Fragment Q_{1_1} durch eine Folge von Nest- und Entnest-Operatoren ersetzt werden. Dabei werden pro Gruppe von `(aid, cid)`-Belegungen die dazugehörigen `length`-Werte in einem Array während der Gruppierung gespeichert. Diese denormalisierte Relation wird anschließend wieder entnestet, wodurch weitere Gruppierungen sowie Aggregatfunktionen angewendet werden können. Eine ausführliche Beschreibung dieses Verfahrens am Beispiel von PostgreSQL finden Sie im Begleitmaterial zu dieser Arbeit (siehe Anhang G).

Anhang E

Detaillierte Evaluation der Laufzeit

Die folgenden Tabellen E.1 bis E.11 zeigen die Auswirkungen der in Anhang C aufgeführten Transformationsregeln am Beispiel des Amarok-Testszenarios. Jede Tabelle zeigt bis zu acht Transformationsregeln in jeweils beiden Varianten: Das $-$ steht für die Auswertungen mit den Originalanfragen, das $+$ für die transformierten Anfragen, die jeweils zehn Mal an die PostgreSQL-Datenbank gestellt. Zu jeder Messreihe wurde die durchschnittliche Ausführungszeit (ohne Ergebnisrückgabe) und die Standardabweichung berechnet. Auf Basis der Durchschnittswerte wurde die prozentuale Änderung der Laufzeit erfasst. Die Messwerte für die weiteren Datenbanksysteme, Db2 und Oracle sowie für die anderen Anwendungsszenarien (TPC-H und MuSAMA) befinden sich im Begleitmaterial zu dieser Arbeit.

Regel	**A01**		**A02**		**A03**		**A04**		**A05**		**A06**		**A07**		**A08**	
Variante	**-**	**+**	**-**	**+**	**-**	**+**	**-**	**+**	**-**	**+**	**-**	**+**	**-**	**+**	**-**	**+**
Messung #1 in ms	41	23	12	37	11	23	11	30	12	27	12	27	12	26	11	25
Messung #2 in ms	13	23	12	30	11	23	11	35	11	27	11	27	12	29	11	33
Messung #3 in ms	12	22	12	26	11	30	11	29	11	27	11	30	11	27	11	26
Messung #4 in ms	13	22	17	26	11	29	11	28	11	27	11	28	11	26	11	28
Messung #5 in ms	12	37	14	23	11	28	11	28	14	27	11	27	11	25	11	26
Messung #6 in ms	12	27	13	23	11	28	11	28	17	27	11	27	11	26	11	26
Messung #7 in ms	12	23	12	23	11	38	11	34	14	27	11	27	11	30	11	24
Messung #8 in ms	11	23	13	23	11	30	11	28	13	27	11	27	11	26	11	25
Messung #9 in ms	11	22	12	22	11	40	11	37	12	27	11	31	11	29	11	24
Messung #10 in ms	11	29	11	22	11	33	11	28	11	27	11	29	11	26	11	25
Durchschnitt in ms	14,8	25,1	12,8	25,5	11,0	30,2	11,0	30,5	12,6	27,0	11,1	28,0	11,2	27,0	11,0	26,2
Standardabweichung in ms	8,76	4,55	1,60	4,50	0,00	5,29	0,00	3,29	1,85	0,00	0,30	1,41	0,40	1,61	0,00	2,52
Laufzeitunterschied in %	69,59		99,22		174,55		177,27		114,29		152,25		141,07		138,18	

Tabelle E.1: Messwerte und Laufzeitunterschiede für die Transformationsregeln A01 bis A08 am Beispiel der Amarok-Datenbank.

Regel	A09		A10		A11		A12		A13		A14		A15		A16	
Variante	-	+	-	+	-	+	-	+	-	+	-	+	-	+	-	+
Messung #1 in ms	11	25	12	22	12	22	12	31	12	28	12	28	12	31	14	34
Messung #2 in ms	11	25	11	22	12	25	12	28	12	28	12	28	15	30	13	32
Messung #3 in ms	11	25	11	22	12	23	12	23	12	28	12	28	15	31	13	33
Messung #4 in ms	11	25	12	22	11	25	12	22	12	28	12	28	13	31	13	32
Messung #5 in ms	11	25	12	22	12	24	12	22	11	29	12	27	12	30	12	33
Messung #6 in ms	11	25	12	22	12	23	12	22	12	29	11	24	12	30	12	32
Messung #7 in ms	11	25	15	22	12	23	12	22	12	29	12	24	11	36	12	33
Messung #8 in ms	12	25	13	22	11	29	12	22	12	30	11	24	11	34	13	34
Messung #9 in ms	12	25	12	22	12	24	12	22	11	29	12	29	12	25	13	32
Messung #10 in ms	12	24	12	22	12	22	12	22	12	29	12	29	11	24	13	31
Durchschnitt in ms	11,3	24,9	12,2	22,0	11,8	24,0	12,0	23,6	11,8	28,7	11,8	26,9	12,4	30,2	12,8	32,6
Standardabweichung in ms	0,46	0,30	1,08	0,00	0,40	1,95	0,00	3,04	0,40	0,64	0,40	1,97	1,43	3,40	0,60	0,92
Laufzeitunterschied in %	120,35		80,33		103,39		96,67		143,22		127,97		143,55		154,69	

Tabelle E.2: Messwerte und Laufzeitunterschiede für die Transformationsregeln A09 bis A16 am Beispiel der Amarok-Datenbank.

Regel	A17		A18		A19		A20		A21		A22		A23		A24	
Variante	-	+	-	+	-	+	-	+	-	+	-	+	-	+	-	+
Messung #1 in ms	13	32	13	27	19	25	14	25	15	29	16	29	12	25	13	25
Messung #2 in ms	13	30	13	27	15	25	14	25	15	29	15	29	12	25	13	25
Messung #3 in ms	13	27	13	27	14	25	14	25	15	29	15	29	12	25	13	25
Messung #4 in ms	12	27	13	27	14	25	14	25	15	29	15	29	13	25	12	26
Messung #5 in ms	12	27	13	27	14	25	14	25	15	29	22	29	14	25	12	25
Messung #6 in ms	13	27	13	27	14	24	14	26	15	28	18	29	14	25	12	26
Messung #7 in ms	13	43	13	27	14	25	14	26	15	28	16	29	13	25	12	26
Messung #8 in ms	13	27	13	27	14	25	14	25	15	28	15	28	13	25	12	26
Messung #9 in ms	14	27	13	27	14	25	14	25	15	29	15	29	13	25	12	25
Messung #10 in ms	13	27	21	27	14	25	14	25	15	29	15	29	12	25	12	25
Durchschnitt in ms	12,9	29,4	13,8	27,0	14,6	24,9	14,0	25,2	15,0	28,7	16,2	28,9	12,8	25,0	12,3	25,4
Standardabweichung in ms	0,54	4,82	2,40	0,00	1,50	0,30	0,00	0,40	0,00	0,46	2,14	0,30	0,75	0,00	0,46	0,49
Laufzeitunterschied in %	127,91		95,65		70,55		80,00		91,33		78,40		95,31		106,50	

Tabelle E.3: Messwerte und Laufzeitunterschiede für die Transformationsregeln A17 bis A24 am Beispiel der Amarok-Datenbank.

Regel	A25		A26		A27		A28		A29		A30		A31		A32	
Variante	-	+	-	+	-	+	-	+	-	+	-	+	-	+	-	+
Messung #1 in ms	14	28	14	28	17	9	17	9	19	26	19	26	19	16	17	15
Messung #2 in ms	13	28	13	29	25	9	17	9	19	26	20	26	18	15	17	19
Messung #3 in ms	13	28	13	29	19	9	17	9	19	25	20	26	18	15	17	17
Messung #4 in ms	14	29	13	29	17	9	17	9	20	26	19	26	22	16	17	17
Messung #5 in ms	13	29	13	30	17	9	17	9	19	26	19	26	20	16	17	16
Messung #6 in ms	13	29	13	30	17	9	17	9	19	26	19	26	18	16	17	16
Messung #7 in ms	13	29	16	29	17	9	17	9	19	26	19	26	17	15	17	16
Messung #8 in ms	13	29	17	29	17	9	17	9	20	26	19	26	17	15	17	16
Messung #9 in ms	16	28	15	30	17	9	17	9	20	26	21	26	17	15	17	16
Messung #10 in ms	14	28	13	29	17	9	17	9	19	26	21	26	17	15	17	16
Durchschnitt in ms	13,6	28,5	14,0	29,2	18,0	9,0	17,0	9,0	19,3	25,9	19,6	26,0	18,3	15,4	17,0	16,4
Standardabweichung in ms	0,92	0,50	1,41	0,60	2,41	0,00	0,00	0,00	0,46	0,30	0,80	0,00	1,55	0,49	0,00	1,02
Laufzeitunterschied in %	109,56		108,57		-50,00		-47,06		34,20		32,65		-15,85		-3,53	

Tabelle E.4: Messwerte und Laufzeitunterschiede für die Transformationsregeln A25 bis A32 am Beispiel der Amarok-Datenbank.

Regel	A33		A34		A35		A36		A37		A38		A39		A40	
Variante	**-**	**+**	**-**	**+**	**-**	**+**	**-**	**+**	**-**	**+**	**-**	**+**	**-**	**+**	**-**	**+**
Messung #1 in ms	17	9	16	10	17	27	16	27	19	28	17	27	18	27	19	28
Messung #2 in ms	16	9	16	10	16	26	18	27	19	27	17	27	17	27	18	28
Messung #3 in ms	16	9	16	10	16	26	17	27	19	27	17	26	17	30	19	27
Messung #4 in ms	16	9	16	10	16	26	17	27	23	27	17	26	17	37	18	28
Messung #5 in ms	16	10	16	10	16	26	17	27	23	27	17	26	18	29	18	28
Messung #6 in ms	17	10	16	10	17	26	17	27	20	28	17	26	18	26	18	28
Messung #7 in ms	16	10	17	10	16	26	19	26	20	45	17	27	17	26	18	28
Messung #8 in ms	16	10	17	10	16	27	18	26	20	30	17	26	17	25	18	28
Messung #9 in ms	16	9	16	10	16	27	16	26	20	28	17	26	17	26	18	28
Messung #10 in ms	16	9	16	10	16	26	16	26	20	28	17	27	17	26	18	28
Durchschnitt in ms	16,2	9,4	16,2	10,0	16,2	26,3	17,1	26,6	20,3	29,5	17,0	26,4	17,3	27,9	18,2	27,9
Standardabweichung in ms	0,40	0,49	0,40	0,00	0,40	0,46	0,94	0,49	1,42	5,24	0,00	0,49	0,46	3,36	0,40	0,30
Laufzeitunterschied in %	-41,98		-38,27		62,35		55,56		45,32		55,29		61,27		53,30	

Tabelle E.5: Messwerte und Laufzeitunterschiede für die Transformationsregeln A33 bis A40 am Beispiel der Amarok-Datenbank.

Regel	A41		A42		A43		A44		A45		A46		A47		A48	
Variante	**-**	**+**	**-**	**+**	**-**	**+**	**-**	**+**	**-**	**+**	**-**	**+**	**-**	**+**	**-**	**+**
Messung #1 in ms	18	29	18	31	17	33	14	33	16	34	9	16	11	16	16	28
Messung #2 in ms	17	28	18	31	16	33	14	31	16	34	9	16	11	16	14	30
Messung #3 in ms	16	28	18	40	16	31	14	30	17	42	9	16	11	16	12	35
Messung #4 in ms	15	28	18	36	16	32	14	30	16	33	9	16	11	16	11	28
Messung #5 in ms	15	28	18	38	16	32	14	35	17	32	9	16	11	16	11	27
Messung #6 in ms	15	28	18	39	17	28	14	39	16	32	9	16	11	16	10	27
Messung #7 in ms	15	28	18	38	17	26	14	31	16	44	9	16	11	19	10	27
Messung #8 in ms	15	28	18	38	16	26	14	30	16	36	9	16	11	17	9	27
Messung #9 in ms	15	28	18	37	16	26	14	32	16	35	12	16	11	17	9	27
Messung #10 in ms	15	28	18	37	16	26	14	31	16	32	11	16	11	17	9	27
Durchschnitt in ms	15,6	28,1	18,0	36,5	16,3	29,3	14,0	32,2	16,2	35,4	9,5	16,0	11,0	16,6	11,1	28,3
Standardabweichung in ms	1,02	0,30	0,00	2,94	0,46	3,00	0,00	2,71	0,40	4,03	1,02	0,00	0,00	0,92	2,21	2,41
Laufzeitunterschied in %	80,13		102,78		79,75		130,00		118,52		68,42		50,91		154,95	

Tabelle E.6: Messwerte und Laufzeitunterschiede für die Transformationsregeln A41 bis A48 am Beispiel der Amarok-Datenbank.

Regel	A49		A50		A51	
Variante	**-**	**+**	**-**	**+**	**-**	**+**
Messung #1 in ms	9	17	9	17	9	27
Messung #2 in ms	9	16	9	16	9	27
Messung #3 in ms	9	16	9	16	9	28
Messung #4 in ms	9	16	9	17	9	30
Messung #5 in ms	9	16	9	16	9	35
Messung #6 in ms	9	17	9	16	9	29
Messung #7 in ms	9	16	9	16	9	37
Messung #8 in ms	9	17	9	17	9	29
Messung #9 in ms	9	16	9	16	9	35
Messung #10 in ms	9	16	9	17	9	30
Durchschnitt in ms	9,0	16,3	9,0	16,4	9,0	30,7
Standardabweichung in ms	0,00	0,46	0,00	0,49	0,00	3,44
Laufzeitunterschied in %	81,11		82,22		241,11	

Tabelle E.7: Messwerte und Laufzeitunterschiede für die Transformationsregeln A49 bis A51 am Beispiel der Amarok-Datenbank.

Regel	K01		K02		K03		K04		K05		K06		K07		K08	
Variante	**-**	**+**	**-**	**+**	**-**	**+**	**-**	**+**	**-**	**+**	**-**	**+**	**-**	**+**	**-**	**+**
Messung #1 in ms	40	24	8	22	68	93	84	89	34	31	7	10	6	10	11	8
Messung #2 in ms	38	23	7	21	59	85	83	82	32	24	6	10	7	10	11	8
Messung #3 in ms	39	22	7	21	60	96	81	81	32	24	6	11	6	10	11	8
Messung #4 in ms	39	20	7	21	59	97	84	81	32	25	6	11	6	10	11	8
Messung #5 in ms	39	20	7	22	59	98	82	81	34	26	6	10	6	10	11	7
Messung #6 in ms	38	21	7	21	60	95	81	83	33	27	6	11	6	10	11	8
Messung #7 in ms	38	21	6	21	61	94	83	82	32	27	6	11	6	11	11	7
Messung #8 in ms	40	21	6	21	59	97	84	78	33	24	6	10	7	11	11	7
Messung #9 in ms	39	21	6	21	62	99	84	80	32	24	7	10	6	14	11	7
Messung #10 in ms	39	20	7	22	59	94	83	78	32	24	6	10	6	13	11	7
Durchschnitt in ms	38,9	21,3	6,8	21,3	60,6	94,8	82,9	81,5	32,6	25,6	6,2	10,4	6,2	10,9	11,0	7,5
Standardabweichung in ms	0,70	1,27	0,60	0,46	2,65	3,74	1,14	2,94	0,80	2,15	0,40	0,49	0,40	1,37	0,00	0,50
Laufzeitunterschied in %	-45,24		213,24		56,44		-1,69		-21,47		67,74		75,81		-31,82	

Tabelle E.8: Messwerte und Laufzeitunterschiede für die Transformationsregeln K01 bis K08 am Beispiel der Amarok-Datenbank.

Regel	K09		K10		K11		K12		K13		K14		K15		K16	
Variante	**-**	**+**	**-**	**+**	**-**	**+**	**-**	**+**	**-**	**+**	**-**	**+**	**-**	**+**	**-**	**+**
Messung #1 in ms	24	21	8	16	22	18	7	16	5	3	3	3	2	1	1	1
Messung #2 in ms	24	21	7	16	21	17	6	16	3	1	2	3	2	2	2	2
Messung #3 in ms	22	22	7	16	21	17	6	16	2	2	2	3	1	1	1	2
Messung #4 in ms	21	22	7	16	23	17	6	16	3	2	3	3	2	2	1	2
Messung #5 in ms	21	22	6	16	21	17	6	16	3	1	2	4	2	1	2	2
Messung #6 in ms	21	22	6	16	21	17	6	16	2	2	3	3	1	2	1	2
Messung #7 in ms	21	22	6	16	21	17	6	16	3	1	2	3	2	1	1	2
Messung #8 in ms	21	34	6	16	21	17	6	16	3	2	2	4	1	1	2	2
Messung #9 in ms	25	27	6	16	21	17	6	16	3	2	3	3	2	2	1	1
Messung #10 in ms	34	23	6	16	21	16	6	16	3	1	2	4	2	1	1	2
Durchschnitt in ms	23,4	23,6	6,5	16,0	21,3	17,0	6,1	16,0	3,0	1,7	2,4	3,3	1,7	1,4	1,3	1,8
Standardabweichung in ms	3,83	3,83	0,67	0,00	0,64	0,45	0,30	0,00	0,77	0,64	0,49	0,46	0,46	0,49	0,46	0,40
Laufzeitunterschied in %	0,85		146,15		-20,19		162,30		-43,33		37,50		-17,65		38,46	

Tabelle E.9: Messwerte und Laufzeitunterschiede für die Transformationsregeln K09 bis K16 am Beispiel der Amarok-Datenbank.

Regel	K17		K18		K19		K20		K21	
Variante	**-**	**+**	**-**	**+**	**-**	**+**	**-**	**+**	**-**	**+**
Messung #1 in ms	1	2	4	4	3	7	25	34	28	32
Messung #2 in ms	2	2	5	3	4	5	24	34	27	31
Messung #3 in ms	2	1	4	4	3	6	25	33	28	30
Messung #4 in ms	3	2	5	3	4	5	24	33	28	30
Messung #5 in ms	1	2	4	4	3	6	24	33	28	38
Messung #6 in ms	2	1	5	3	3	5	24	33	28	33
Messung #7 in ms	2	2	4	4	4	5	24	33	28	28
Messung #8 in ms	2	2	5	5	3	4	24	33	28	28
Messung #9 in ms	2	1	5	5	6	5	24	33	27	27
Messung #10 in ms	1	2	4	5	3	5	24	33	28	27
Durchschnitt in ms	1,8	1,7	4,5	4,0	3,6	5,3	24,2	33,2	27,8	30,4
Standardabweichung in ms	0,60	0,46	0,50	0,77	0,92	0,78	0,40	0,40	0,40	3,20
Laufzeitunterschied in %	-5,56		-11,11		47,22		37,19		9,35	

Tabelle E.10: Messwerte und Laufzeitunterschiede für die Transformationsregeln K17 bis K21 am Beispiel der Amarok-Datenbank.

Regel	**L01**		**L02**		**L03**		**L04**		**L05**		**L06**		**L07**		**L08**	
Variante	**-**	**+**	**-**	**+**	**-**	**+**	**-**	**+**	**-**	**+**	**-**	**+**	**-**	**+**	**-**	**+**
Messung #1 in ms	15	20	7	11	6	18	15	19	10	12	6	9	8	15	23	29
Messung #2 in ms	15	19	6	11	7	13	15	23	10	11	7	8	7	14	23	30
Messung #3 in ms	14	18	7	11	6	12	15	26	11	13	7	9	6	14	22	30
Messung #4 in ms	15	19	7	12	7	12	15	21	10	12	6	8	8	15	23	29
Messung #5 in ms	15	19	6	10	6	11	14	20	10	12	7	8	7	14	22	30
Messung #6 in ms	15	19	7	11	7	11	15	19	10	13	6	8	6	14	23	29
Messung #7 in ms	16	20	6	12	6	11	14	19	10	12	7	9	7	14	23	31
Messung #8 in ms	14	19	7	11	6	11	15	18	11	12	7	8	7	15	23	30
Messung #9 in ms	15	19	6	10	7	11	14	19	12	13	6	8	6	14	23	30
Messung #10 in ms	15	20	6	11	6	10	15	18	10	12	7	9	7	14	23	30
Durchschnitt in ms	14,9	19,2	6,5	11,0	6,4	12,0	14,7	20,2	10,4	12,2	6,6	8,4	6,9	14,3	22,8	29,8
Standardabweichung in ms	0,54	0,60	0,50	0,63	0,49	2,14	0,46	2,40	0,66	0,60	0,49	0,49	0,70	0,46	0,40	0,60
Laufzeitunterschied in %	28,86		69,23		87,50		37,41		17,31		27,27		107,25		30,70	

Tabelle E.11: Messwerte und Laufzeitunterschiede für die Transformationsregeln L01 bis L08 am Beispiel der Amarok-Datenbank.

Anhang F

Demonstrator

In diesem Anhang werden weitere technische Details zum entwickelten Demonstrator dargelegt. Im ersten Abschnitt F.1 werden programmiertechnische Hinweise zur Nutzung der API für potentielle Erweiterungen des Demonstrators gegeben. Der letzte Abschnitt F.2 stellt eine Gebrauchsanleitung zur Einrichtung und Nutzung des Demonstrators in eigenen Anwendungsumgebungen dar.

F.1 Erweiterung der JDBC-Middleware

Der Demonstrator basiert auf der bekannten JDBC-Middleware (Java DataBase Connectivity). In den folgenden Abschnitten werden zunächst die Grundlagen von JDBC kurz erklärt, bevor auf die erfolgten Erweiterungen eingegangen wird.

F.1.1 Grundlagen von JDBC

Java DataBase Connectivity[1] ist eine Schnittstelle (engl.: *Application Programming Interface*; API) für die Programmiersprache Java. Diese API erlaubt den Zugriff auf eine Vielzahl von verschiedenen Datenquellen, wie relationalen Datenbanken, CSV-Dateien oder Textdateien. JDBC nutzt dabei das Java-eigene Konzept *Remote Method Invocation*[2] (engl.: Fernaufruf von Methoden; RMI), um die Methoden zum Ausführen bzw. Verarbeiten von Datenbankanfragen auf einem entfernten Rechner zu verwirklichen.

Für verschiedene Datenbanksysteme (Server) existieren dafür eigene Implementationen der notwendigen Schnittstellen zum Ausführen der Datenbankanfrage. Lokal werden auf dem Client lediglich die von Java bereitgestellten, standardisierten Schnittstellen genutzt. Dadurch wird auf der Anwendungsebene vom konkreten Datenbanksystem abstrahiert. Der Aufbau und die Verwendung der Schnittstellen werden im Rahmen der vorgenommenen Erweiterungen im nächsten Abschnitt beschrieben.

F.1.2 Erweiterungen

Abbildung F.1 zeigt die wichtigsten Klassen zur entwickelten JDBC-Erweiterung. Das Diagramm enthält die wichtigsten Klassen für Client und Server sowie die dazugehörigen Schnittstellen. Nachfolgend werden die einzelnen Klassen und Schnittstellen genauer beschrieben. Beachten Sie, dass der Quellcode nur auszugsweise wiedergegeben ist. Die vollständige Implementierung und Dokumentation kann dem Begleitmaterial zu dieser Arbeit

[1] Nähere Beschreibungen zu JDBC sind unter `https://docs.oracle.com/javase/8/docs/technotes/guides/jdbc/` und `https://docs.oracle.com/javase/8/docs/api/index.html?java/sql/package-summary.html` verfügbar. Beide Websites wurden zuletzt am 05.01.2022 aufgerufen.

[2] Weitere Informationen zu RMI können unter `https://docs.oracle.com/javase/8/docs/technotes/guides/rmi/` sowie `https://docs.oracle.com/javase/8/docs/api/java/rmi/package-summary.html` aufgerufen werden. Der letzte Aufruf der Websites erfolgte am 05.01.2022.

(siehe Anhang G) entnommen werden. Der Großteil der Implementierungsarbeiten erfolgte im Rahmen eines studentischen Projektes [RLRG16].

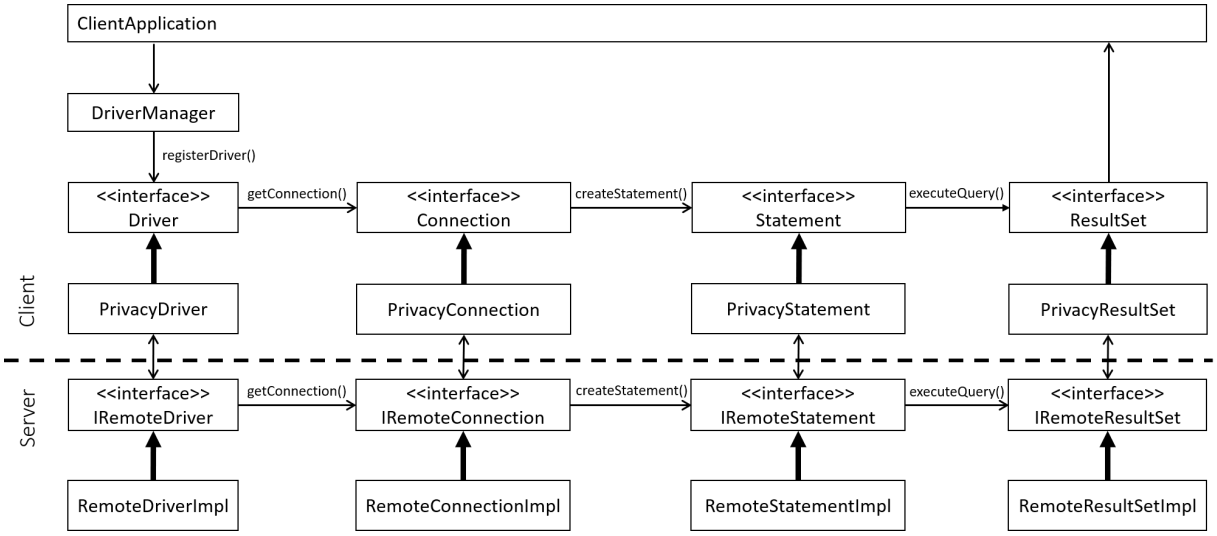


Abbildung F.1: Das Diagramm zeigt die wichtigsten Java-Klassen von JDBC und die dazugehörigen Erweiterungen durch den entwickelten Demonstrators.

Package *client*

Das Package `client` umfasst alle Klassen, die zur Ausführung der Anwendung auf der Seite des Clients notwendig sind. Alle Klassen, mit Ausnahme der `ClientApplication`, implementieren die von JDBC vorgegebenen Interfaces. Die einzelnen Klassen und Methoden können dabei wie gewohnt verwendet werden; lediglich bei der Verwendung von zusätzlich implementierten Methoden ist ein expliziter Cast auf die konkreten Klassen notwendig.

Klasse *ClientApplication*: Die Klasse `ClientApplication` dient als Demoanwendung für den entwickelten Treiber. Der Start des Programms erfolgt über die `main`-Methode, welche nacheinander den Treiber lädt und anschließend eine Verbindung zum Server aufbaut. Danach wird eine Beispielanfrage erzeugt und ausgeführt.

Aus dem Ergebnis der Anfrage werden die Metadaten extrahiert, darunter die Anzahl der zurückgegebenen Attribute. Anschließend wird für jedes Ergebnistupel jeder Attributwert auf der Ausgabekonsole ausgegeben. Abschließend werden die offenen Verbindungen geschlossen. Um auf Fehler zu reagieren, wurde eine rudimentäre Fehlerbehandlung hinzugefügt. In den weiteren Klassen ist die Fehlerbehandlung auch vorhanden, wird jedoch zwecks Lesbarkeit des Quellcodes nicht näher beschrieben.

Klasse *PrivacyDriver*: Die Klasse `PrivacyDriver` (siehe Quellcodeausschnitt F.1) implementiert das Interface `Driver`. Sie dient zum einen zur Bereitstellung der notwendigen Informationen, wie dem URL-Präfix sowie der Versionsnummer, und zum anderen zur Registrierung des Treibers beim Java-internen `DriverManager`. Die Hauptfunktionalität der Klasse besteht in der Bereitstellung einer `Connection` für den Verbindungsaufbau zur Datenbank.

```
1  public class PrivacyDriver implements Driver {
2    ...
3    @Override
4    public Connection connect(String url, Properties info) {
5      PrivacyConnection localConInstance = null;
6      if(acceptsURL(url)) {
7        String server = url.substring(URL_PREFIX.length(), url.length());
8        connectRemote(server);
9        IRemoteConnection remoteConInstance =
10         (IRemoteConnection) remoteDriver.getConnection();
11       PrivacyPolicy objClient = new PrivacyPolicy();
12       objClient.getPrivacyPolicy(new File("policies/Policy.xml"));
13       objClient.compareWithBasics();
14       remoteConInstance.setPrivacy(objClient);
15       localConInstance = new PrivacyConnection(remoteConInstance);
16     }
17     return (Connection) localConInstance;
18   }
19   ...
20 }
```

Quellcodeausschnitt F.1: Ausschnitt aus dem Quellcodes des *PrivacyDriver*s.

Im Gegensatz zu den Standardimplementierungen werden in der Methode `connect` zusätzlich (Zeile 11 bis 14) die Datenschutzeinstellungen (siehe Abschnitt 3.4 und Anhang B.1) aus einer zuvor angelegten Konfigurationsdatei geladen. Diese werden mit den intern festgelegten Mindestanforderungen verglichen und ggf. verschärft. Die geladenen Datenschutzeinstellungen werden von den anderen Klassen bei der Ausführung von Anfragen und der Ergebnisrückgabe aufgerufen, um die Algorithmen zur Anfrageverteilung bzw. Anonymisierung zu parametrisieren.

Klasse *PrivacyConnection*: Die Klasse `PrivacyConnection` implementiert das Interface `Connection`. Im Vergleich zu anderen JDBC-Treibern wurden in dieser Klasse keine Anpassungen hinsichtlich Datenschutzaspekten vorgenommen, da die Klasse nur für die Verbindung zwischen Client und Datenbankmanagementsystem zuständig ist.

Klasse *PrivacyStatement*: Das Interface `Statement` und die Klasse `PrivacyStatement` (siehe Quellcodeausschnitt F.2) dienen zur Ausführung der SQL-Anfragen. Über das `PrivacyStatement` wird sichergestellt, dass die geltende Datenschutzrichtlinie (Zeile 5) eingehalten wird. Zudem wird sichergestellt, dass beim Abfragen des Anfrageergebnisses (Zeile 12) die ggf. anonymisierte Ergebnisrelation in Form eines `PrivacyResultSet`s zurückgegeben wird.

```
1  public class PrivacyStatement implements Statement {
2    ...
3    public PrivacyStatement(IRemoteStatement stmt) {
4      this.remoteStmt = stmt;
5      objClient = stmt.getPrivacyPolicy();
6    }
7
8    @Override
9    public ResultSet executeQuery(String sql) {
10     IRemoteResultSet remoteRsInstance = null;
11     remoteRsInstance = (IRemoteResultSet) remoteStmt.executeQuery(sql);
```

```
12      PrivacyResultSet localRsInstance = new PrivacyResultSet(remoteRsInstance);
13      return localRsInstance;
14    }
15    ...
16  }
```

Quellcodeausschnitt F.2: Ausschnitt für die Klasse *PrivacyStatement*.

Klasse *PrivacyResultSet*: Das `PrivacyResultSet` repräsentiert die Implementierung des Interfaces `ResultSet` auf Client-Seite und empfängt die Ergebnisrelation vom Server entgegen. Da die Implementierung der Datenschutzaspekte auf Seite des Servers erfolgt, wurden hier keine weiteren Anpassungen vorgenommen.

Package *server*

Das Package `server` umfasst alle Klassen, die zur server- bzw. proxyseitigen Ausführung der Anwendung notwendig sind. Alle Klassen implementieren die von JDBC vorgegebenen Interfaces. Der Anwender bzw. Anwendungsentwickler kommt mit diesen Klassen nicht direkt in Kontakt, da durch die Verwendung der Interfaces die Klassen- und Methodenaufrufe durchgeführt werden. Im Folgenden werden auszugsweise die Klassen `RemoteConnectionImpl` und `RemoteStatementImpl` vorgestellt. Diese stellen die serverseitigen Gegenstücke zu den zuvor vorgestellten Klassen aus dem `client`-package dar.

Klasse *RemoteConnectionImpl*: Die Klasse `RemoteConnectionImpl` implementiert das Interface `RemoteConnection` (siehe Quellcodeausschnitt F.3). Beim Erzeugen eines neuen `Statements` werden neben den datenbankspezifischen Einstellungen auch die weitergeleiteten Datenschutzeinstellungen in der Methode `setPrivacy` (siehe Zeile 12) für die weiteren Verarbeitungsschritte der Anfrage festgesetzt und mit den ggf. vorhandenen lokalen Einstellungen verglichen und ggf. verschärft.

```
1  public class RemoteConnectionImpl extends UnicastRemoteObject
2                          implements IRemoteConnection {
3    ...
4    @Override
5    public IRemoteStatement createStatement() {
6      RemoteStatementImpl stmtImplInstance =
7        new RemoteStatementImpl(sqlConnection.createStatement());
8      stmtImplInstance.setPrivacy(objServer);
9      return (IRemoteStatement) stmtImplInstance;
10   }
11
12   public void setPrivacy(PrivacyPolicy obj) {
13     objServer = new PrivacyPolicy();
14     objServer.getPrivacyPolicy(new File(xmlPath));
15     objServer.compareWithBasics();
16     objServer.comparePolicies(obj);
17     objServer.setConnection(this);
18     objServer.fillPrivacy();
19   }
20   ...
21 }
```

Quellcodeausschnitt F.3: Ausschnitt für die Klasse *RemoteConnectionImpl*.

Klasse *RemoteStatementImpl*: Die Klasse `RemoteStatementImpl` implementiert das Interface `RemoteStatement` (siehe Quellcodeausschnitt F.4). Vor Ausführung der Anfrage (Zeile 6) besteht die Möglichkeit, diese zu modifizieren (im Quellcode nicht dargestellt). In Zeile 9 wird das Ergebnis der Anfrage ermittelt. Dabei wird ggf. die Anonymisierung des Ergebnisses vorgenommen, sofern dies durch die im Vorfeld übermittelte Datenschutzrichtlinie vorgesehen ist. Abschließend wird das Ergebnis an den Client übermittelt.

```
1  public class RemoteStatementImpl extends UnicastRemoteObject
2                            implements IRemoteStatement{
3    ...
4    @Override
5    public IRemoteResultSet executeQuery(String query) {
6      ResultSet rs = sqlStatement.executeQuery(query);
7      RemoteResultSetImpl remoteRs = new RemoteResultSetImpl(rs);
8      remoteRs.setPrivacy(objServer);
9      remoteRs = (RemoteResultSetImpl) objServer.getPrivacyResultSet(remoteRs);
10     return (IRemoteResultSet) remoteRs;
11   }
12   ...
13 }
```

Quellcodeausschnitt F.4: Ausschnitt für die Klasse *RemoteStatementImpl*.

Package *privacy*

Innerhalb des Packages `privacy` werden alle Algorithmen zusammengefasst, welche zur Anfrage- und Ergebnistransformation im Prä- bzw. Postprozessor (siehe Abbildung 3.1) zum Einsatz kommen. Dies betrifft die in Anhang B.1.3 vorgestellte Datenschutzauszeichnungssprache PP4SE als auch die Algorithmen zur Erkennung von Quasi-Identifikatoren (siehe Abschnitt 4.4) und zur Transformation von Anfragen (siehe Abschnitt 6.5).

F.2 Einrichtung des Demonstrators in einer eigenen Umgebung

In diesem Abschnitt wird die Einrichtung des Demonstrators auf einem eigenen Rechner erklärt. Dabei wird einerseits auf das Kompilieren und Starten der Anwendung eingegangen, andererseits auf die Konfiguration der einzelnen Komponenten.

F.2.1 Start des Demonstrators

Zum Start des Demonstrators kann entweder das kompilierte Java-Archiv, `Dashboard.jar`, genutzt werden oder aus dem bereitgestellten Quellcode eine lauffähige Version des Programms erstellt werden. Die folgenden Abschnitte erläutern beide Varianten genauer.

Kompilierung des Quellcodes

Der Quellcode zum vollständigen Projekt liegt als Eclipse-Projekt dem Begleitmaterial der Arbeit bei und ist zudem in der aktuellsten Fassung im GitLab des Bereichs Informatik der Universität Rostock[3] verfügbar. Die zur Ausführung notwendigen, externen Bibliotheken liegen im `lib`-Ordner des Projektes bzw. werden via Maven automatisch nachgeladen.

Als zentraler Programmeinstiegspunkt dient die Klasse `dashboard.DashboardGui`, welche sich im `src`-Ordner des Projekts befindet (siehe Abbildung F.2). Innerhalb von Eclipse kann durch einen Rechtsklick auf die

[3] `https://git.informatik.uni-rostock.de/hg/patch-queryrewriter`, zuletzt aufgerufen am 05.01.2022

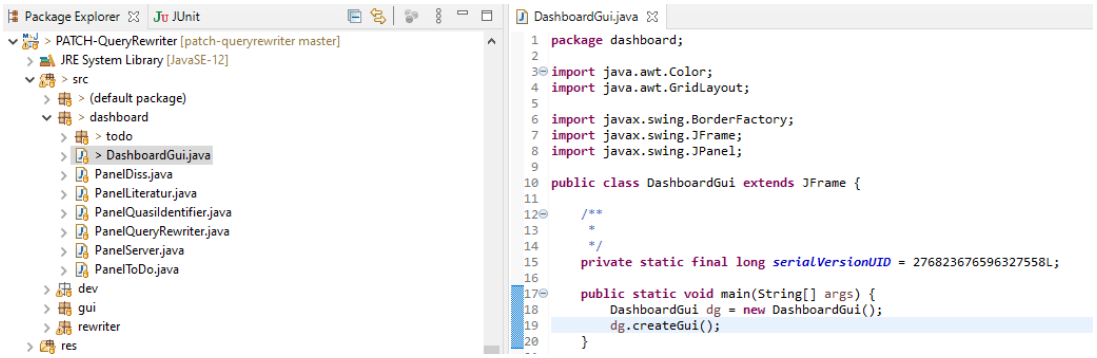


Abbildung F.2: Der Quellcode des Demonstrators liegt als Eclipse-Projekt vor. Die Abbildung zeigt die Klasse $DashboardGui$, welche als Einstiegspunkt in alle Teilaspekte des Demonstrators dient.

Abbildung F.3: Nach erfolgreichem Start des Programms öffnet sich ein Dashboard, über welches die wichtigsten Funktionen erreichbar sind. Dies betrifft die drei vorgestellten Konzepte ($Anfragetransformation$ (links), $Quasi-Identifikatoren$ (mittig) und den $Anfrageprozessor$ (rechts).

Datei, gefolgt von der Auswahl `Run As` → `Java Application`, das Programm gestartet werden. Für andere Programmierumgebungen folgt der Start der Anwendung analog.

Nutzung des vorkompilierten Archivs

Neben dem Quellcode liegt das kompilierte Programm bereits als lauffähiges Java-Archiv vor. Dieses sollte mit einem Doppelklick das in Abbildung F.3 dargestellte Dashboard öffnen. Alternativ dazu lässt sich das Programm mit dem Kommandozeilenbefehl `java -jar dashboard.jar` öffnen.

F.2.2 Konfiguration

Die Konfiguration der einzelnen Teilkomponenten *Anfragetransformation*, *Quasi-Identifikatoren* und *Anfrageprozessor* wird über die jeweiligen Konfigurationsschaltflächen vorgenommen und bleibt auch über das Programmende hinaus bestehen. Im Folgenden werden die möglichen Konfigurationseinstellungen näher erläutert.

Gemeinsame Konfiguration

Als gemeinsame Parameter teilen sich alle Einstellungen die Verbindungsdaten zum verwendeten Datenbanksystem. Als Standardeinstellung wird, abweichend von den Evaluierungen der vorherigen Kapitel, eine SQLite-Datenbank benutzt, welche als Standalone-Variante auf dem lokalen Dateisystem verwendet werden kann. Die

verwendete Beispieldatenbank, *chinook*[4], ist in einem ähnlichen Szenario angesiedelt wie die Amarok-Musik-Datenbank, ergänzt diese jedoch um weitere Relationen wie Kunden und Einkäufe. Dadurch ähnelt die chinook-Datenbank in Teilen auch dem TPC-H-Benchmark.

Es lassen sich in der Konfiguration diejenigen Parameter einstellen, die zum Aufbau einer Datenbankverbindung via JDBC notwendig sind. Dies betrifft dementsprechend folgende Einstellungen (Standardwerte in Klammern):

- Datenbanksystem (SQLite); Auswahl via Drop-Down-Menü
- Host („res" als lokaler Ordner)
- Port (leer)
- Datenbank („chinook.db" als lokale Datei)
- Nutzername (leer)
- Passwort (leer)

Anfragetransformation

Diese Komponente öffnet den in Kapitel 6.5 beschriebenen Demonstrator. Neben der Konfiguration der einzelnen Ebenen bzgl. ihrer Anfragekapazität (siehe Abbildung 6.16) lassen sich dort zudem die voreingestellten Optimierungsregeln (siehe Abbildung F.4) festlegen.

Quasi-Identifikatoren

Der in Kapitel 4.4 eingeführte Algorithmus zur Bestimmung von Quasi-Identifikatoren wird über diese Komponente gesteuert. Als Parameter können die Relationen der zuvor eingestellten Datenbank ausgewählt werden sowie der Grenzwert für die QI-Eigenschaft eingesetzt werden. Über ein Drop-Down-Menü kann der zu verwendende Algorithmus ausgewählt werden. Durch einen Klick auf die Schaltfläche `Ausführen` wird der Algorithmus ausgeführt. Im Anschluss kann über die Schaltfläche `Berechne minimale Überdeckung` die minimale Anzahl an zu anonymisierenden Attributen bestimmt werden. Über die Schaltfläche `Log ansehen` können die Berechnungen beider Schritte nachvollzogen werden.

Anfrageprozessor

Die Beispielimplementierung des Anfrageprozessors ist in drei Ebenen aufgeteilt:

- Server: Als Ausgangsdatenbank wird die Datenbank genutzt, welche in den Einstellungen hinterlegt ist.
- Proxy: Ein Zwischenknoten zwischen Client und Server übernimmt den restlichen Teil der Anfrage.
- Client: Auf dem Client wird das Ergebnis der Anfrage entgegengenommen.

Diese Konfiguration dient als Testumgebung für den in Kapitel 3 vorgestellten JDBC-Treiber. Für jede Ebene lässt sich der verantwortliche Teil des JDBC-Treibers über die dazugehörigen Schaltflächen starten und beenden. Eine Kontrollleuchte gibt Aufschluss über den jeweiligen Systemzustand (grün $\widehat{=}$ alles in Ordnung, rot $\widehat{=}$ nicht gestartet / Fehler).

[4] Weiterführende Informationen zur Beispieldatenbank lassen sich unter `https://www.sqlitetutorial.net/sqlite-sample-database/`, zuletzt aufgerufen am 05.01.2022, finden.

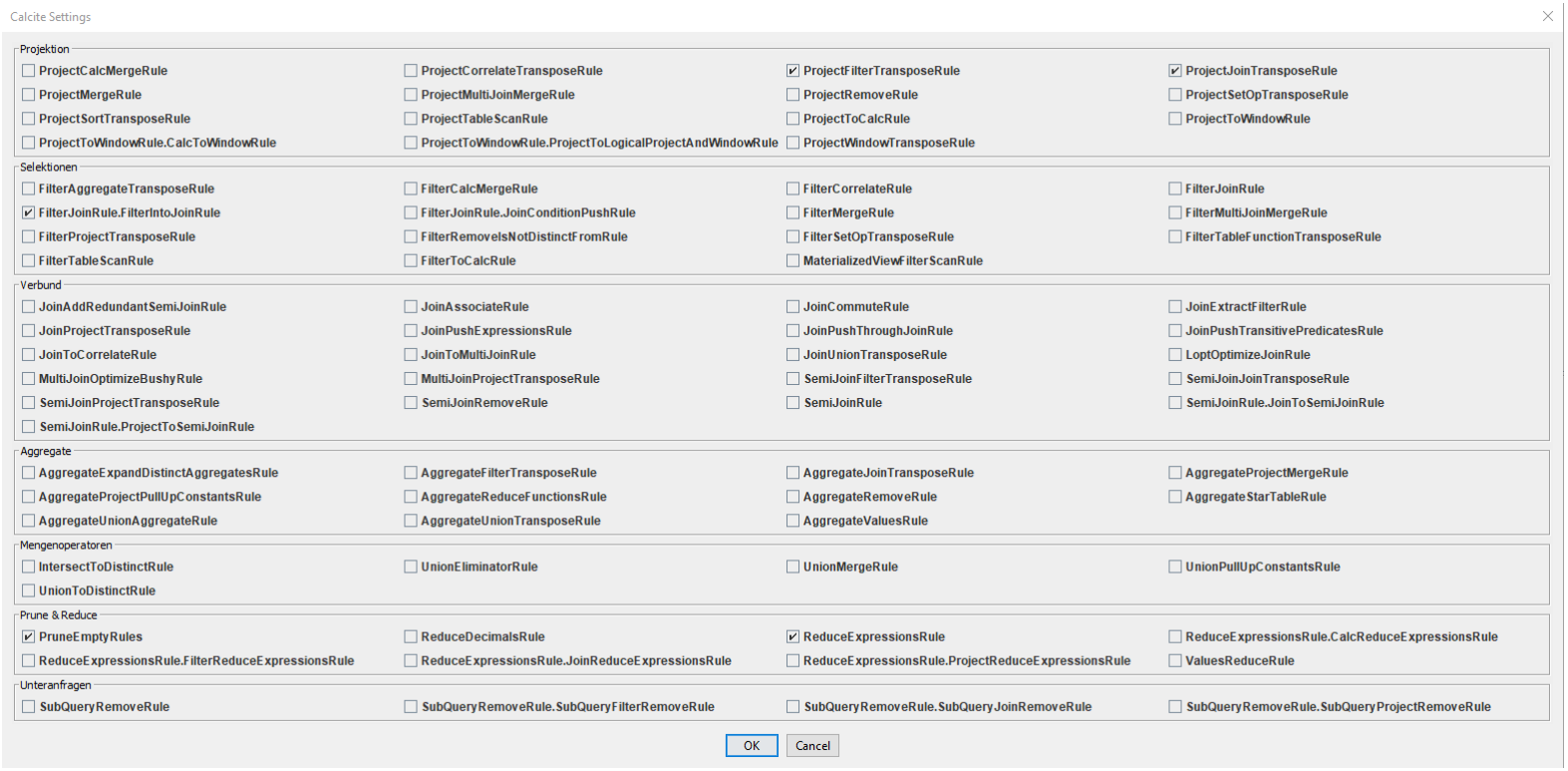


Abbildung F.4: Übersicht über die Optimierungsregeln in Apache Calcite. Standardmäßig sind die Regeln $ProjectFilterTransposeRule$, $ProjectJoinTransposeRule$, $FilterIntoJoinRule$ sowie $PruneEmptyRules$ und $ReduceExpressionRule$ aktiviert.

Anhang G

Begleitmaterial

Im Rahmen dieser Arbeit wurden verschiedene Datensätze, Literaturquellen und Programmteile verwendet bzw. entwickelt, die nicht Bestandteil der gedruckten Abgabefassung dieser Arbeit sind. Wurde diese Arbeit aus einer digitalen Bibliothek bezogen, so können dort einzelne Teile des Materials als Download angeboten sein. Eine vollständige Sammlung des Begleitmaterials befindet sich im GitLab des Bereichs Informatik der Universität Rostock[1].

Die Struktur und der Inhalt des Begleitmaterials werden im Folgenden kurz erläutert. Die Überschriften entsprechen den Ordnern im Wurzelverzeichnis des Begleitmaterials.

Datensätze und Auswertungen

- Ein Technischer Bericht mit
  - einer separaten Beschreibung der Transformationsregeln sowie
  - den dazugehörigen Messwerten.
- Die Amarok-, MuSAMA- und TPC-H-Datensätze mit
  - den gestellten Anfragen und
  - deren Auswertung sowie
  - den dazugehörigen Anfrageplänen in grafischer und textueller Form.

PArADISE

- Der aktuelle Stand des Demonstrators zum Zeitpunkt des Drucks dieser Arbeit[2],
- das Wiki zum Projekt, entnommen dem GitLab der Universität Rostock,
- das generierte Javadoc und
- ein ausführbares Java-Archiv als Client-Anwendung zu einer bestehenden, lokal lauffähigen Testumgebung.
- Die verwendeten Technologien und externe Bibliotheken (siehe Kapitel F),

[1] `https://git.informatik.uni-rostock.de/hg/Dissertation`, zuletzt aufgerufen am 05.01.2022

[2] Die aktuellen Fassungen sind zudem universitätsintern unter `https://git.informatik.uni-rostock.de/dbis/PArADISE/PArADISE` (JDBC-Treiber) sowie `https://git.informatik.uni-rostock.de/hg/patch-queryrewriter` (Transformationswerkzeug) verfügbar.

Dissertation

- Das laufende Beispiel in der kompletten Fassung,
- eigene Publikationen inkl. Bibtex-Einträge,
- sämtliche Tex-Dateien und Abbildungen der Arbeit,
- die digitale Fassung der Arbeit sowie
- ein Video-Tutorial zu dem in Abschnitt 6.5 vorgestellten Demonstrator.

Quellen: Die im Literaturverzeichnis aufgelisteten Quellen, mit Ausnahme digital nicht verfügbarer Werke, werden hier aufgelistet. Es wurden ebenso alle Webseiten, welche innerhalb der Arbeit zitiert wurden, als Portable Dokument Format (*.pdf) exportiert und bereitgestellt. Der Dateiname der Quellen entspricht dem Kürzel im Literaturverzeichnis bzw. der Nummer der Fußnote (für Online-Quellen). Hinweis: Die beigefügte Literatur dient der Nachvollziehbarkeit von Aussagen und darf nicht öffentlich bereitgestellt werden. Dies gilt speziell für Werke, welche durch Lizenzverträge der Universität Rostock durch die Universitätsbibliothek zur Verfügung gestellt wurden.

Software

- das Java Runtime Environment (JRE) und Java Development Kit (JDK) in der zum Demonstrator passenden Version und
- das Zip-Programm 7-Zip in der Version 21.07.

Abbildungsverzeichnis

Tabellenverzeichnis

Index

Literaturverzeichnis

[AAA+21] ALWARAFY, ABDULMALIK, KHALED A. AL-THELAYA, MOHAMED ABDALLAH, JENS SCHNEIDER und MOUNIR HAMDI: *A Survey on Security and Privacy Issues in Edge-Computing-Assisted Internet of Things*. IEEE Internet of Things Journal, 8(6):4004–4022, 2021.

[ABE+13] ARASU, ARVIND, SPYROS BLANAS, KEN EGURO, RAGHAV KAUSHIK, DONALD KOSSMANN, RAVISHANKAR RAMAMURTHY und RAMARATHNAM VENKATESAN: *Orthogonal Security with Cipherbase*. In: *CIDR*. www.cidrdb.org, 2013.

[ABR21] ASGHAR, HIRA, CHRISTOPHE BOBINEAU und MARIE-CHRISTINE ROUSSET: *Compatibility Checking Between Privacy and Utility Policies: A Query-Based Approach*. Research Report, Université Grenoble Alpes ; CNRS ; Grenoble INP ; Laboratoire d'informatique de Grenoble, 2021.

[AC17] AFRATI, FOTO und RADA CHIRKOVA: *Answering Queries Using Views*. Synthesis Lectures on Data Management, 9(2):1–235, 2017.

[ADG14] AFRATI, FOTO N., MATTHEW DAMIGOS und MANOLIS GERGATSOULIS: *On Solving Efficiently the View Selection Problem under Bag and Bag-Set semantics*. Information Systems, 42:153–176, 2014.

[AFB07] AVANES, ARTIN, JOHANN CHRISTOPH FREYTAG und CHRISTOF BORNHÖVD: *Distributed Service Deployment in Mobile Ad-Hoc Networks*. In: *MobiQuitous*, Seiten 1–8. IEEE Computer Society, 2007.

[Agg05] AGGARWAL, CHARU C.: *On k-Anonymity and the Curse of Dimensionality*. In: *VLDB*, Seiten 901–909. ACM, 2005.

[AH18a] AUGE, TANJA und ANDREAS HEUER: *Inverses in Research Data Management: Combining Provenance Management, Schema and Data Evolution (Inverse im Forschungsdatenmanagement)*. In: *Grundlagen von Datenbanken*, Band 2126 der Reihe *CEUR Workshop Proceedings*, Seiten 108–113. CEUR-WS.org, 2018.

[AH18b] AUGE, TANJA und ANDREAS HEUER: *The Theory behind Minimizing Research Data: Result equivalent CHASE-inverse Mappings*. In: GEMULLA, RAINER, SIMONE PAOLO PONZETTO, CHRISTIAN BIZER, MARGRET KEUPER und HEINER STUCKENSCHMIDT (Herausgeber): *Proceedings of the Conference 'Lernen, Wissen, Daten, Analysen', LWDA 2018, Mannheim, Germany, August 22-24, 2018.*, Band 2191 der Reihe *CEUR Workshop Proceedings*, Seiten 1–12. CEUR-WS.org, 2018.

[AHW95] AIKEN, ALEXANDER, JOSEPH M. HELLERSTEIN und JENNIFER WIDOM: *Static Analysis Techniques for Predicting the Behavior of Active Database Rules*. ACM Transactions on Database Systems, 20(1):3–41, 1995.

[AIS93] AGRAWAL, RAKESH, TOMASZ IMIELINSKI und ARUN N. SWAMI: *Mining Association Rules between Sets of Items in Large Databases*. In: BUNEMAN, PETER und SUSHIL JAJODIA (Herausgeber): *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, Seiten 207–216. ACM Press, 1993.

[ALM04] AFRATI, FOTO N., CHEN LI und PRASENJIT MITRA: *On Containment of Conjunctive Queries with Arithmetic Comparisons*. In: BERTINO, ELISA, STAVROS CHRISTODOULAKIS, DIMITRIS PLEXOUSAKIS, VASSILIS CHRISTOPHIDES, MANOLIS KOUBARAKIS, KLEMENS BÖHM und ELENA FERRARI (Herausgeber): *Advances in Database Technology - EDBT 2004, 9th International Conference on Extending Database Technology, Heraklion, Crete, Greece, March 14-18, 2004, Proceedings*, Band 2992 der Reihe *Lecture Notes in Computer Science*, Seiten 459–476. Springer, 2004.

[AMS+15] AL-AZIZY, DALAL, DAVID E. MILLARD, IRAKLIS SYMEONIDIS, KIERON O'HARA und NIGEL SHADBOLT: *A Literature Survey and Classifications on Data Deanonymisation*. In: *CRiSIS*, Band 9572 der Reihe *Lecture Notes in Computer Science*, Seiten 36–51. Springer, 2015.

[BA18] BÄR, DOROTHEE und THORSTEN ALSLEBEN: *Interview mit Staatsministerin Dorothee Bär - Digitales Bürgerkonto wäre ein Gewinn an Datensouveränität*. mittelstandsmagazin, 18(2):17–19, 2018.

[BB06] BYUN, JI-WON und ELISA BERTINO: *Micro-views, or on how to protect privacy while enhancing data usability: concepts and challenges*. SIGMOD Record, 35(1):9–13, 2006.

[BBL10] BEKARA, K., Y. BEN MUSTAPHA und M. LAURENT: *XPACML eXtensible Privacy Access Control Markup Langua*. In: *The Second International Conference on Communications and Networking*, Seiten 1–5, Nov 2010.

[BBtCP15] BENEDIKT, MICHAEL, PIERRE BOURHIS, BALDER TEN CATE und GABRIELE PUPPIS: *Querying Visible and Invisible Tables in the Presence of Integrity Constraints*. CoRR, abs/1509.01683, 2015.

[BC07] BÜNNIG, CHRISTIAN und CLEMENS H CAP: *Five Objectives to Diminish User Concerns About Privacy in Smart Environments*. In: *Proc. Advances in Mobile Computing & Multimedia (MoMM) and Information Integration and Webbased Applications & Services (iiWAS) Workshops*, Seiten 179–188, 2007.

[BCB+13] BARCELLONA, CETTINA, PIETRO CASSARÀ, GIUSEPPE DI BELLA, JOVAN DJ. GOLIC und ILENIA TINNIRELLO: *Multi-party metering: An architecture for privacy-preserving profiling schemes*. In: *SustainIT*, Seiten 1–6. IEEE Computer Society, 2013.

[BCH13] BAHLOUL, SARAH NAIT, EMMANUEL COQUERY und MOHAND-SAID HACID: *Securing Materialized Views: a Rewriting-Based Approach*. In: *29emes Journées BDA*, Seiten 1–25, 2013.

[BCH+18] BEGOLI, EDMON, JESÚS CAMACHO-RODRÍGUEZ, JULIAN HYDE, MICHAEL J. MIOR und DANIEL LEMIRE: *Apache Calcite: A Foundational Framework for Optimized Query Processing Over Heterogeneous Data Sources*. In: *SIGMOD Conference*, Seiten 221–230. ACM, 2018.

[BCMU21] BABUN, LEONARDO, Z. BERKAY CELIK, PATRICK MCDANIEL und A. SELCUK ULUAGAC: *Real-time Analysis of Privacy-(un)aware IoT Applications*. Proceedings on Privacy Enhancing Technologies, 2021(1):145–166, 2021.

[BDH+13] BAULIG, WERNER, REINHARD DANKERT, ROLF HELLWIG, GABRIEL SCHULZ und GESINE NAAB: *Datenschutz in Unternehmen und Behörden*. Blockseminar Datenschutz, Universität Rostock, 2013.

[BE15] BENKAOUZ, YAHYA und MOHAMMED ERRADI: *A distributed protocol for privacy preserving aggregation with non-permanent participants*. Computing, 97(9):893–912, 2015.

[BEH06] BRUDER, ILVIO, CHRISTOPH EIGENSTETTER und ANDREAS HEUER: *Flying Pics*. Wissensmeer - Journal für die Wissenschaftsgemeinschaft Region Rostock, 2006.

[Bel15] BELLMAN, RICHARD: *Adaptive Control Processes - A Guided Tour (Reprint from 1961)*, Band 2045 der Reihe *Princeton Legacy Library*. Princeton University Press, 2015.

[BG16] BUTTERSTEIN, DENNIS und TORSTEN GRUST: *Precision Performance Surgery for PostgreSQL: LLVM-based Expression Compilation, Just in Time*. Proc. VLDB Endowment, 9(13):1517–1520, 2016.

[BGS05] BERTHOLD, OLIVER, OLIVER GÜNTHER und SARAH SPIEKERMANN: *RFID: Verbraucherängste und Verbraucherschutz*. Wirtschaftsinformatik, 47(6):422–430, 2005.

[BHX17] BALASUBRAMANIAN, ADHITHYA, SUMI HELAL und YI XU: *Latency Optimization in Large-Scale Cloud-Sensor Systems*. Open Journal of Internet Of Things (OJIOT), 3(1):18–30, 2017. Special Issue: Proceedings of the International Workshop on Very Large Internet of Things (VLIoT 2017) in conjunction with the VLDB 2017 Conference in Munich, Germany.

[Bib77] BIBA, KENNETH J.: *Integrity considerations for secure computer systems*. Technischer Bericht, DTIC Document, 1977.

[Biz07] BIZER, JOHANN: *Sieben Goldene Regeln des Datenschutzes*. Datenschutz und Datensicherheit - DuD, 31(5):350–356, May 2007.

[BL73] BELL, D. ELLIOTT und LEONARD J. LAPADULA: *Secure Computer Systems: Mathematical Foundations*. Technischer Bericht, MITRE Corp., Bedford, MA, 1973.

[BL08] BYUN, JI-WON und NINGHUI LI: *Purpose based access control for privacy protection in relational database systems*. VLDB Journal, 17(4):603–619, 2008.

[BN12] BADER, SEBASTIAN und MARTIN NYOLT: *A Context-Aware Publish-Subscribe Middleware for Distributed Smart Environments*. In: *PerCom Workshops*, Seiten 100–104. IEEE Computer Society, 2012.

[BN14] BOBERACH, MICHAEL und RAHILD NEUBURGER: *Zukunftspfade Digitales Deutschland 2020*. HMD - Praxis Wirtschaftsinformatik, 51(6):762–772, 2014.

[BPV15] BENEDIKT, MICHAEL, GABRIELE PUPPIS und HUY VU: *The complexity of higher-order queries*. Information and Computation, 244:172–202, 2015.

[Bro84] BRODIE, MICHAEL L: *On the Development of Data Models*. In: *On Conceptual Modelling*, Seiten 19–47. Springer, 1984.

[BSMM06] BRONSTEIN, ILJA N., KONSTANTIN A. SEMENDJAJEW, GERHARD MUSIOL und HEINER MÜHLIG: *Taschenbuch der Mathematik. 6., vollständig überarbeitete und ergänzte Auflage*. Verlag Harri Deutsch, 2006.

[BSS19] BEIER, KATHARINA, MARK SCHWEDA und SILKE SCHICKTANZ: *Taking patient involvement seriously: A critical ethical analysis of participatory approaches in data-intensive medical research*. BMC Medical Informatics and Decision Making, 19(1):1–10, 2019.

[Bun83] BUNDESVERFASSUNGSGERICHT: *Urteil Az. 1 BvR 209/83, 1 BvR 484/83, 1 BvR 440/83, 1 BvR 420/83, 1 BvR 362/83, 1 BvR 269/83**. `https://openjur.de/u/268440.html`, letzter Zugriff am 28. April 2021, 1983.

[Bun05] BUNDESARBEITSGEMEINSCHAFT DER SENIOREN-ORGANISATIONEN E. V.: *Verbraucherforum für Senioren – Ergebnisse einer Befragung zum Thema 'Wohnen im Alter'*, 2005.

[Bun16] BUNDESVERBAND DER ENERGIE- UND WASSERWIRTSCHAFT: *Welche Smart Home-Anwendungen halten Sie im Haushalt für sinnvoll?* Statista - Das Statistik-Portal, `de.statista.com/statistik/daten/studie/576883/umfrage/umfrage-zu-sinnvollen-smart-home-anwendungen-im-haushalt/`, letzter Zugriff am 23. Februar 2018, 2016.

[CCFS21] CUNNINGHAM, TEDDY, GRAHAM CORMODE, HAKAN FERHATOSMANOGLU und DIVESH SRIVASTAVA: *Real-World Trajectory Sharing with Local Differential Privacy*. Proc. VLDB Endow., 14(11):2283–2295, 2021.

[CCS18] CHU, SHUMO, ALVIN CHEUNG und DAN SUCIU: *Axiomatic Foundations and Algorithms for Deciding Semantic Equivalences of SQL Queries*. CoRR, abs/1802.02229, 2018.

[CdVF+07] CIRIANI, VALENTINA, SABRINA DE CAPITANI DI VIMERCATI, SARA FORESTI, SUSHIL JAJODIA, STEFANO PARABOSCHI und PIERANGELA SAMARATI: *Fragmentation and Encryption to Enforce Privacy in Data Storage*. In: *ESORICS*, Band 4734 der Reihe *Lecture Notes in Computer Science*, Seiten 171–186. Springer, 2007.

[CEK16] CIGLIC, MARGARETA, JOHANN EDER und CHRISTIAN KONCILIA: *Anonymization of Data Sets with NULL Values*. Transactions on Large-Scale Data- and Knowledge-Centered Systems, 24:193–220, 2016.

[CIM+11] CHUN, BYUNG-GON, SUNGHWAN IHM, PETROS MANIATIS, MAYUR NAIK und ASHWIN PATTI: *CloneCloud: Elastic Execution between Mobile Device and Cloud*. In: *EuroSys*, Seiten 301–314. ACM, 2011.

[CKPM05] CANTOR, SCOTT, JOHN KEMP, ROB PHILPOTT und EVE MALER: *Assertions and Protocols for the OASIS Security Assertion Markup Language*. OASIS Standard (March 2005), Seiten 1–86, 2005.

[CM77] CHANDRA, ASHOK K. und PHILIP M. MERLIN: *Optimal Implementation of Conjunctive Queries in Relational Data Bases*. In: HOPCROFT, JOHN E., EMILY P. FRIEDMAN und MICHAEL A. HARRISON (Herausgeber): *Proceedings of the 9th Annual ACM Symposium on Theory of Computing, May 4-6, 1977, Boulder, Colorado, USA*, Seiten 77–90. ACM, 1977.

[CNS99] COHEN, SARA, WERNER NUTT und ALEXANDER SEREBRENIK: *Rewriting Aggregate Queries Using Views*. In: VIANU, VICTOR und CHRISTOS H. PAPADIMITRIOU (Herausgeber): *Proceedings of the Eighteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, May 31 - June 2, 1999, Philadelphia, Pennsylvania, USA*, Seiten 155–166. ACM Press, 1999.

[CNS07] COHEN, SARA, WERNER NUTT und YEHOSHUA SAGIV: *Deciding Equivalences among Conjunctive Aggregate Queries*. Journal of the ACM, 54(2):5, 2007.

[Cod82] CODD, E. F.: *Relational Database: A Practical Foundation for Productivity*. Communications of the ACM, 25(2):109–117, 1982.

[Coh06] COHEN, SARA: *Equivalence of Queries Combining Set and Bag-Set Semantics*. In: VANSUMMEREN, STIJN (Herausgeber): *Proceedings of the Twenty-Fifth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 26-28, 2006, Chicago, Illinois, USA*, Seiten 70–79. ACM, 2006.

[CV93] CHAUDHURI, SURAJIT und MOSHE Y. VARDI: *Optimization of* Real *Conjunctive Queries*. In: BEERI, CATRIEL (Herausgeber): *Proceedings of the Twelfth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, May 25-28, 1993, Washington, DC, USA*, Seiten 59–70. ACM Press, 1993.

[CWCS16] CHU, SHUMO, KONSTANTIN WEITZ, ALVIN CHEUNG und DAN SUCIU: *HoTTSQL: Proving Query Rewrites with Univalent SQL Semantics*. CoRR, abs/1607.04822, 2016.

[CY14] CHIRKOVA, RADA und TING YU: *Obtaining Information about Queries behind Views and Dependencies*. CoRR, abs/1403.5199, 2014.

[Dab17] DABELS, RICHARD: *Automatische Generierung von Domänengeneralisierungshierarchien*. Bachelorarbeit, Universität Rostock, 2017. `http://eprints.dbis.informatik.uni-rostock.de/881/`.

[Dal86] DALENIUS, TORE: *Finding a Needle In a Haystack or Identifying Anonymous Census Records*. Journal of Official Statistics, 2(3):329–336, 1986.

[DCLL15] DUFROMENTEL, SÉBASTIEN, SYLVIE CAZALENS, FRANÇOIS LESUEUR und PHILIPPE LAMARRE: *QTor: A Flexible Publish/Subscribe Peer-to-Peer Organization Based on Query Rewriting*. In: *DEXA (2)*, Band 9262 der Reihe *Lecture Notes in Computer Science*, Seiten 507–519. Springer, 2015.

[DDH+15] DANEZIS, GEORGE, JOSEP DOMINGO-FERRER, MARIT HANSEN, JAAP-HENK HOEPMAN, DANIEL LE MÉTAYER, RODICA TIRTEA und STEFAN SCHIFFNER: *Privacy and Data Protection by Design - from policy to engineering*. CoRR, abs/1501.03726, 2015.

[DDK+15] D'ACQUISTO, GIUSEPPE, JOSEP DOMINGO-FERRER, PANAYIOTIS KIKIRAS, VICENÇ TORRA, YVES-ALEXANDRE DE MONTJOYE und ATHENA BOURKA: *Privacy by design in big data: An overview of privacy enhancing technologies in the era of big data analytics*. CoRR, abs/1512.06000, 2015.

[DDS79] DENNING, DOROTHY E., PETER J. DENNING und MAYER D. SCHWARTZ: *The Tracker: A Threat to Statistical Database Security*. ACM Transactions on Database Systems, 4(1):76–96, 1979.

[DEP+15] DHUNGANA, DEEPAK, GERHARD ENGELBRECHT, JOSIANE XAVIER PARREIRA, ANDREAS SCHUSTER und DANILO VALERIO: *Aspern smart ICT: Data analytics and privacy challenges in a smart city*. In: *WF-IoT*, Seiten 447–452. IEEE Computer Society, 2015.

[Deu49] DEUTSCHER BUNDESTAG: *Grundgesetz für die Bundesrepublik Deutschland vom 23. Mai 1949 (BGBl. S. 1), zuletzt geändert durch Artikel 1 und 2 Satz 2 des Gesetzes vom 29. September 2020 (BGBl. I S. 2048)*. `https://www.bundestag.de/gg`, Mai 1949. Zuletzt aufgerufen am 29.04.2021.

[Deu15] DEUTSCHER BUNDESTAG: *Bundesdatenschutzgesetz in der Fassung der Bekanntmachung vom 14. Januar 2003 (BGBl. I S. 66), das zuletzt durch Artikel 1 des Gesetzes vom 25. Februar 2015 (BGBl. I S. 162) geändert worden ist*, 2015. Zuletzt aufgerufen am 28.04.2021.

[Deu17a] DEUTSCHER BUNDESTAG: *Strafgesetzbuch in der Fassung der Bekanntmachung vom 13. November 1998 (BGBl. I S. 3322), das zuletzt durch Artikel 1 des Gesetzes vom 30. Oktober 2017 (BGBl. I S. 3618) geändert worden ist)*. `https://www.gesetze-im-internet.de/stgb/`, 2017. Zuletzt aufgerufen am 28.04.2021.

[Deu17b] DEUTSCHER BUNDESTAG: *Telekommunikationsgesetz vom 22. Juni 2004 (BGBl. I S. 1190), das zuletzt durch Artikel 10 Absatz 12 des Gesetzes vom 30. Oktober 2017 (BGBl. I S. 3618) geändert worden ist)*. `https://www.gesetze-im-internet.de/tkg_2004/`, 2017. Zuletzt aufgerufen am 28.04.2021.

[Deu17c] DEUTSCHER ETHIKRAT: *Big Data und Gesundheit - Datensouveränität als informationelle Freiheitsgestaltung, Stellungnahme*. http://www.ethikrat.org/dateien/pdf/stellungnahme-big-data-und-gesundheit.pdf, 2017. Zuletzt aufgerufen am 16.05.2018.

[Deu18] DEUTSCHER BUNDESTAG: *Bundesdatenschutzgesetz vom 30. Juni 2017 (BGBl. I S. 2097)*. https://www.gesetze-im-internet.de/bdsg_2018/BJNR209710017.html, 2018. Zuletzt aufgerufen am 28.04.2021.

[DFKB13] DANEZIS, GEORGE, CÉDRIC FOURNET, MARKULF KOHLWEISS und SANTIAGO ZANELLA BÉGUELIN: *Smart meter aggregation via secret-sharing*. In: *SEGS@CCS*, Seiten 75–80. ACM, 2013.

[DFSC15] DOMINGO-FERRER, JOSEP und JORDI SORIA-COMAS: *From t-closeness to differential privacy and vice versa in data anonymization*. Knowledge-Based Systems, 74:151–158, 2015.

[DG97] DUSCHKA, OLIVER M. und MICHAEL R. GENESERETH: *Answering Recursive Queries Using Views*. In: *PODS*, Seiten 109–116. ACM Press, 1997.

[DGL00] DUSCHKA, OLIVER M., MICHAEL R. GENESERETH und ALON Y. LEVY: *Recursive Query Plans for Data Integration*. Journal of Logic Programming, 43(1):49–73, 2000.

[DH13] DEUTSCH, ALIN und RICHARD HULL: *Provenance-Directed Chase&Backchase*. In: TANNEN, VAL, LIMSOON WONG, LEONID LIBKIN, WENFEI FAN, WANG-CHIEW TAN und MICHAEL P. FOURMAN (Herausgeber): *In Search of Elegance in the Theory and Practice of Computation - Essays Dedicated to Peter Buneman*, Band 8000 der Reihe *Lecture Notes in Computer Science*, Seiten 227–236. Springer, 2013.

[DHI12] DOAN, ANHAI, ALON Y. HALEVY und ZACHARY G. IVES: *Principles of Data Integration*. Morgan Kaufmann, 2012.

[Dit13] DITTRICH, JENS: *Relationale Algebra: Gruppierung und Aggregation*. https://www.youtube.com/watch?v=BJ7FqbItwuA, 2013. Zuletzt aufgerufen am 07.12.2018.

[DJL79] DOBKIN, DAVID P., ANITA K. JONES und RICHARD J. LIPTON: *Secure Databases: Protection Against User Influence*. ACM Transactions on Database Systems, 4(1):97–106, 1979.

[DK14] DHAIGUDE, AJINKYA A und PREETHAM KUMAR: *Improved Slicing Algorithm For Greater Utility In Privacy Preserving Data Publishing*. International Journal of Data Engineering, 5, 2014.

[DLN05] DEUTSCH, ALIN, BERTRAM LUDÄSCHER und ALAN NASH: *Rewriting Queries Using Views with Access Patterns Under Integrity Constraints*. In: *ICDT*, Band 3363 der Reihe *Lecture Notes in Computer Science*, Seiten 352–367. Springer, 2005.

[DLN07] DEUTSCH, ALIN, BERTRAM LUDÄSCHER und ALAN NASH: *Rewriting queries using views with access patterns under integrity constraints*. Theoretical Computer Science, 371(3):200–226, 2007.

[DMMS06] DOMINGO-FERRER, JOSEP, ANTONI MARTÍNEZ-BALLESTÉ, JOSEP MARIA MATEO-SANZ und FRANCESC SEBÉ: *Efficient multivariate data-oriented microaggregation*. VLDB Journal, 15(4):355–369, 2006.

[DMNS06] DWORK, CYNTHIA, FRANK MCSHERRY, KOBBI NISSIM und ADAM SMITH: *Calibrating Noise to Sensitivity in Private Data Analysis*. In: HALEVI, SHAI und TAL RABIN (Herausgeber): *Theory of Cryptography*, Band 3876 der Reihe *Lecture Notes in Computer Science*, Seiten 265–284. Springer Berlin Heidelberg, 2006.

[Döl16] DÖLLE, LUKAS: *Der Schutz der Privatsphäre bei der Anfragebearbeitung in Datenbanksystemen*. Doktorarbeit, Humboldt Universität zu Berlin, 2016.

[DPT06] DEUTSCH, ALIN, LUCIAN POPA und VAL TANNEN: *Query Reformulation with Constraints*. SIGMOD Record, 35(1):65–73, 2006.

[DR14] DWORK, CYNTHIA und AARON ROTH: *The Algorithmic Foundations of Differential Privacy*. Foundations and Trends in Theoretical Computer Science, 9(3-4):211–407, 2014.

[DWZK19] DING, ZEYU, YUXIN WANG, DANFENG ZHANG und DAN KIFER: *Free Gap Information from the Differentially Private Sparse Vector and Noisy Max Mechanisms*. Proceedings of the VLDB Endowment, 13(3):293–306, 2019.

[DZC08] DAN ZURAS (CHAIR), MIKE COWLISHAW (EDITOR), ET AL.: *IEEE Standard for Floating-Point Arithmetic*. Technischer Bericht, IEEE Computer Society, 2008.

[DZT13] DENNL, CHRISTOPHER, DANIEL ZIENER und JÜRGEN TEICH: *Acceleration of SQL Restrictions and Aggregations through FPGA-Based Dynamic Partial Reconfiguration*. In: *21st IEEE Annual International Symposium on Field-Programmable Custom Computing Machines, FCCM 2013, Seattle, WA, USA, April 28-30, 2013*, Seiten 25–28. IEEE Computer Society, 2013.

[EGKP11] ENEV, MIRO, SIDHANT GUPTA, TADAYOSHI KOHNO und SHWETAK N. PATEL: *Televisions, Video Privacy, and Powerline Electromagnetic Interference*. In: CHEN, YAN, GEORGE DANEZIS und VITALY SHMATIKOV (Herausgeber): *Proceedings of the 18th ACM Conference on Computer and Communications Security, CCS 2011, Chicago, Illinois, USA, October 17-21, 2011*, Seiten 537–550. ACM, 2011.

[Eic18] EICHER, CLAUS CHRISTOPH: *Datenspeicher Auto: Nichts bleibt geheim!* ADAC motorwelt, 2018 (Sommerspezial 07/08):38–39, 2018.

[ENSS15] EICHNER, CHRISTIAN, THOMAS NOCKE, HANS-JÖRG SCHULZ und HEIDRUN SCHUMANN: *Interactive Presentation of Geo-Spatial Climate Data in Multi-Display Environments*. ISPRS International Journal of Geo-Information, 4(2):493–514, 2015.

[Ess09] ESSOH, ALEX DIDIER: *Cloud Computing und Sicherheit - Geht denn das?* Bundesamt für Sicherheit in der Informationstechnik, 4. Grundschutz-Tag, 2009.

[Eur95] EUROPÄISCHES PARLAMENT: *Richtlinie 95/46/EG des europäischen Parlamentes und des Rates vom 24. Oktober 1995 zum Schutz natürlicher Personen bei der Verarbeitung personenbezogener Daten und zum freien Datenverkehr*, 1995.

[Eur11] EUROPÄISCHE KOMMISSION: *Special Eurobarometer 359: Attitudes on Data Protection and Electronic Identity in the European Union*. Special Eurobarometer, 2011.

[Eur16] EUROPÄISCHE UNION: *Verordnung (EU) 2016/679 des Europäischen Parlaments und des Rates vom 27. April 2016 zum Schutz natürlicher Personen bei der Verarbeitung personenbezogener Daten, zum freien Datenverkehr und zur Aufhebung der Richtlinie 95/46/EG (Datenschutz-Grundverordnung)*. EG (Datenschutz-Grundverordnung) Amtsblatt der Europäischen Union, 2016.

[FBD14] FREYTAG, JOHANN-CHRISTOPH, RICO BERGMANN und LUKAS DÖLLE: *Query Adaptation and Privacy for Real-Time Business Intelligence - Extended Abstract*. In: *BIRTE*, Band 206 der Reihe *Lecture Notes in Business Information Processing*, Seiten 1–8. Springer, 2014.

[FKMP03] FAGIN, RONALD, PHOKION G KOLAITIS, RENÉE J MILLER und LUCIAN POPA: *Data Exchange: Semantics and Query Answering*. In: *International Conference on Database Theory*, Seiten 207–224. Springer, 2003.

[FML+15] FELDHEGE, FRANK, ANETT MAU-MOELLER, TOBIAS LINDNER, ALBERT HEIN, ANDREAS MARKSCHIES, UWE KLAUS ZETTL und RAINER BADER: *Accuracy of a Custom Physical Activity and Knee Angle Measurement Sensor System for Patients with Neuromuscular Disorders and Gait Abnormalities*. Sensors, 15(5):10734–10752, 2015.

[FMLJ16] FU, SHUAI, JIANFENG MA, HONGTAO LI und QI JIANG: *A robust and privacy-preserving aggregation scheme for secure smart grid communications in digital communities*. Security and Communication Networks, 9(15):2779–2788, 2016.

[Fre87] FREYTAG, JOHANN CHRISTOPH: *A Rule-Based View of Query Optimization*. In: DAYAL, UMESHWAR und IRVING L. TRAIGER (Herausgeber): *Proceedings of the Association for Computing Machinery Special Interest Group on Management of Data 1987 Annual Conference, San Francisco, CA, USA, May 27-29, 1987*, Seiten 173–180. ACM Press, 1987.

[GAZ+14] GKOUNTOUNA, OLGA, SOTIRIS ANGELI, ATHANASIOS ZIGOMITROS, MANOLIS TERROVITIS und YANNIS VASSILIOU: *k m -Anonymity for Continuous Data Using Dynamic Hierarchies*. In: *Privacy in Statistical Databases*, Band 8744 der Reihe *Lecture Notes in Computer Science*, Seiten 156–169. Springer, 2014.

[GCN+19] GHEORGHE, ALIN-GABRIEL, CONSTANTIN-COSMIN CRECANA, CATALIN NEGRU, FLORIN POP und CIPRIAN DOBRE: *Decentralized Storage System for Edge Computing*. In: IOSUP, ALEXANDRU, RADU PRODAN, ALEXANDRU UTA und FLORIN POP (Herausgeber): *18th International Symposium on Parallel and Distributed Computing, ISPDC 2019, Amsterdam, The Netherlands, June 3-7, 2019*, Seiten 41–49. IEEE, 2019.

[Ges18] GESELLSCHAFT FÜR INFORMATIK E. V. (GI): *Unsere ethischen Leitlinien*. `https://gi.de/fileadmin/GI/Allgemein/PDF/GI_Ethische_Leitlinien_2018.pdf`, 2018. Zuletzt aufgerufen am 13.02.2019.

[GGH17] GOLTZ, JOHANNES, HANNES GRUNERT und ANDREAS HEUER: *De-Anonymisierungsverfahren: Kategorisierung und Anwendung für Datenbankanfragen (De-Anonymization: Categorization and Use-Cases for Database Queries)*. In: *LWDA*, Band 1917 der Reihe *CEUR Workshop Proceedings*, Seite 126. CEUR-WS.org, 2017.

[GH14] GRUNERT, HANNES und ANDREAS HEUER: *Big Data und der Fluch der Dimensionalität: Die effiziente Suche nach Quasi-Identifikatoren in hochdimensionalen Daten*. In: *Grundlagen von Datenbanken*, Band 1313 der Reihe *CEUR Workshop Proceedings*, Seiten 29–34. `http://ceur-ws.org`, 2014.

[GH15a] GRUNERT, HANNES und ANDREAS HEUER: *Generating Privacy Constraints for Assistive Environments*. In: MAGLOGIANNIS, ILIAS, VANGELIS METSIS, GIAN-LUCA MARIOTTINI und OLIVER KORN (Herausgeber): *Proceedings of the 8th International Conference on PErvasive Technologies Related to Assistive Environments, PETRA 2015, Island of Corfu, Greece, July 01 - 03, 2015*. ACM, 2015.

[GH15b] GRUNERT, HANNES und ANDREAS HEUER: *Generating Privacy Constraints for Assistive Environments*. Technischer Bericht, Fakultät für Informatik und Elektrotechnik der Universität Rostock, 2015.

[GH15c] GRUNERT, HANNES und ANDREAS HEUER: *Slicing in Assistenzsystemen - Wie trotz Anonymisierung von Daten wertvolle Analyseergebnisse gewonnen werden können*. In: *GvD*, Band 1366 der Reihe *CEUR Workshop Proceedings*, Seiten 24–29. CEUR-WS.org, 2015.

[GH16a] GRUNERT, HANNES und ANDREAS HEUER: *Datenschutz im PArADISE*. Datenbank-Spektrum, 16(2):107–117, 2016.

[GH16b] GRUNERT, HANNES und ANDREAS HEUER: *Privacy Protection through Query Rewriting in Smart Environments*. In: *EDBT*, Seiten 708–709. OpenProceedings.org, 2016.

[GH17] GRUNERT, HANNES und ANDREAS HEUER: *Rewriting Complex Queries from Cloud to Fog under Capability Constraints to Protect the Users' Privacy*. Open Journal of Internet Of Things (OJIOT), 3(1):31–45, 2017. Special Issue: Proceedings of the International Workshop on Very Large Internet of Things (VLIoT 2017) in conjunction with the VLDB 2017 Conference in Munich, Germany.

[GH18] GRUNERT, HANNES und ANDREAS HEUER: *Query Rewriting by Contract under Privacy Constraints*. Open Journal of Internet Of Things (OJIOT), 4(1):54–69, 2018. Special Issue: Proceedings of the International Workshop on Very Large Internet of Things (VLIoT 2018) in conjunction with the VLDB 2018 Conference in Rio de Janeiro, Brazil.

[GKB+16] GRUNERT, HANNES, MARTIN KASPARICK, BJÖRN BUTZIN, ANDREAS HEUER und DIRK TIMMERMANN: *From Cloud to Fog and Sunny Sensors*. In: *LWDA*, Band 1670 der Reihe *CEUR Workshop Proceedings*, Seiten 83–88. CEUR-WS.org, 2016.

[GKK04] GOMULKIEWICZ, MARCIN, MAREK KLONOWSKI und MIROSLAW KUTYLOWSKI: *Onions Based on Universal Re-encryption - Anonymous Communication Immune Against Repetitive Attack*. In: *WISA*, Band 3325 der Reihe *Lecture Notes in Computer Science*, Seiten 400–410. Springer, 2004.

[GKM+03] GUNOPULOS, DIMITRIOS, RONI KHARDON, HEIKKI MANNILA, SANJEEV SALUJA, HANNU TOIVONEN und RAM SEWAK SHARM: *Discovering All Most Specific Sentences*. ACM Transactions on Database Systems, 28(2):140–174, 2003.

[GLY99] GARCIA-MOLINA, HECTOR, WILBURT LABIO und RAMANA YERNENI: *Capability-Sensitive Query Processing on Internet Sources*. In: *ICDE*, Seiten 50–59. IEEE Computer Society, 1999.

[GM93] GRAEFE, GOETZ und WILLIAM J. MCKENNA: *The Volcano Optimizer Generator: Extensibility and Efficient Search*. In: *ICDE*, Seiten 209–218. IEEE Computer Society, 1993.

[GM08] GARCIA-MOLINA, HECTOR: *Database Systems: The Complete Book*. Pearson Education India, 2008.

[GMLY99] GARCIA-MOLINA, HECTOR, WILBURT LABIO und RAMANA YERNENI: *Capability-Sensitive Query Processing on Internet Sources*. In: *Proceedings of the 15th International Conference on Data Engineering*, Seiten 50–59. IEEE, 1999.

[Gol17] GOLTZ, JOHANNES: *De-Anonymisierungsverfahren: Kategorisierung und deren Anwendung für Datenbankanfragen*. Bachelorarbeit, Universität Rostock, 2017. `http://eprints.dbis.informatik.uni-rostock.de/802/`.

[Goo21] GOOGLE SCHOLAR: *Top publications: Database & Information Systems*. `https://scholar.google.com/citations?view_op=top_venues&hl=en&vq=eng_databasesinformationsystems`, 2021. Zuletzt aufgerufen am 06.10.2021.

[GOP14a] GOTTLOB, GEORG, GIORGIO ORSI und ANDREAS PIERIS: *Query Rewriting and Optimization for Ontological Databases*. CoRR, abs/1405.2848, 2014.

[GOP+14b] GRAY, SCOTT C, FATMA OZCAN, HEBERT PEREYRA, BERT VAN DER LINDEN und ADRIANA ZUBIRI: *SQL-on-Hadoop without compromise*. Whitepaper, IBM Software Group, 2014.

[Gra93] GRAEFE, GOETZ: *Query Evaluation Techniques for Large Databases*. ACM Computing Surveys, 25(2):73–170, 1993.

[GRT04] GRUMBACH, STÉPHANE, MAURIZIO RAFANELLI und LEONARDO TININI: *On the equivalence and rewriting of aggregate queries*. Acta Informatica, 40(8):529–584, 2004.

[Gru14] GRUNERT, HANNES: *Distributed Denial of Privacy*. In: *Informatik 2014, 44. Jahrestagung der Gesellschaft für Informatik e.V. (GI), Big Data - Komplexität meistern, 22.-26. September 2014, Stuttgart*, 2014.

[Gry99] GRYZ, JAREK: *Query Rewriting Using Views in the Presence of Functional and Inclusion Dependencies*. Information Systems, 24(7):597–612, 1999.

[GT14] GHIGLIERI, MARCO und ERIK TEWS: *A privacy protection system for HbbTV in smart TVs*. In: *11th IEEE Consumer Communications and Networking Conference (CCNC)*, Seiten 357–362. IEEE, 2014.

[GV16] GENG, QUAN und PRAMOD VISWANATH: *Optimal Noise Adding Mechanisms for Approximate Differential Privacy*. IEEE Transactions on Information Theory, 62(2):952–969, 2016.

[GW99] GIANNELLA, C. und C.M. WYSS: *Finding Minimal Keys in a Relation Instance*. Technischer Bericht, Pennsylvania State University, 1999.

[Hal01] HALEVY, ALON Y.: *Answering queries using views: A survey*. The VLDB Journal, 10(4):270–294, 2001.

[Hal02] HALPERIN, ERAN: *Improved Approximation Algorithms for the Vertex Cover Problem in Graphs and Hypergraphs*. SIAM Journal on Computing, 31(5):1608–1623, 2002.

[Han11] HANSEN, MARIT: *Top 10 Mistakes in System Design from a Privacy Perspective and Privacy Protection Goals*. In: *PrimeLife*, Band 375 der Reihe *IFIP Advances in Information and Communication Technology*, Seiten 14–31. Springer, 2011.

[HDF14] HAJIAN, SARA, JOSEP DOMINGO-FERRER und ORIOL FARRÀS: *Generalization-based privacy preservation and discrimination prevention in data publishing and mining*. Data Mining and Knowledge Discovery, 28(5-6):1158–1188, 2014.

[HDM+15] HAJIAN, SARA, JOSEP DOMINGO-FERRER, ANNA MONREALE, DINO PEDRESCHI und FOSCA GIANNOTTI: *Discrimination- and privacy-aware patterns*. Data Mining and Knowledge Discovery, 29(6):1733–1782, 2015.

[HFLP89] HAAS, LAURA M., JOHANN CHRISTOPH FREYTAG, GUY M. LOHMAN und HAMID PIRAHESH: *Extensible Query Processing in Starburst*. In: *SIGMOD Conference*, Seiten 377–388. ACM Press, 1989.

[HG20] HEUER, ANDREAS und HANNES GRUNERT: *The Internet of Things as a Privacy-Aware Database Machine*. Open Journal of Internet Of Things (OJIOT), 6(1):53–65, 2020.

[HHK09] HEIN, ALBERT, ANDRÉ HOFFMEYER und THOMAS KIRSTE: *Utilizing an Accelerometric Bracelet for Ubiquitous Gesture-Based Interaction*. In: STEPHANIDIS, CONSTANTINE (Herausgeber): *Universal Access in Human-Computer Interaction. Intelligent and Ubiquitous Interaction Environments, 5th International Conference, UAHCI 2009, Held as Part of HCI International 2009, San Diego, CA, USA, July 19-24, 2009. Proceedings, Part II*, Band 5615 der Reihe *Lecture Notes in Computer Science*, Seiten 519–527. Springer, 2009.

[HK05] HEIDER, THOMAS und THOMAS KIRSTE: *Smart Environments and Self-Organizing Appliance Ensembles*. In: *Dagstuhl Seminar Proceedings*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2005.

[HK08] HEIN, ALBERT und THOMAS KIRSTE: *Towards Recognizing Abstract Activities: An Unsupervised Approach*. In: GOTTFRIED, BJÖRN und HAMID K. AGHAJAN (Herausgeber): *Proceedings of the 2nd Workshop on Behaviour Monitoring and Interpretation, BMI'08, Kaiserslautern, Germany, September 23, 2008*, Band 396 der Reihe *CEUR Workshop Proceedings*, Seiten 102–114. CEUR-WS.org, 2008.

[HK10] HEIN, ALBERT und THOMAS KIRSTE: *Unsupervised Detection of Motion Primitives in very High Dimensional Sensor Data*. In: *Proceedings of the 5th Workshop on Behaviour Monitoring and Interpretation, BMI*, Band 10, 2010.

[HKHT06] HEUER, ANDREAS, THOMAS KIRSTE, WOLFGANG HOFFMANN und DIRK TIMMERMANN: *Coast: Concept for Proactive Assistive Systems and Technologies*. Technischer Bericht, Fakultät für Informatik und Elektrotechnik der Universität Rostock, 2006.

[HL715] HL7 DEUTSCHLAND E. V.: *Arztbrief 2014/2015 auf Basis der HL7 Clinical Document Architecture Release 2 für das deutsche Gesundheitswesen*, 2015.

[HLM12] HARDT, MORITZ, KATRINA LIGETT und FRANK MCSHERRY: *A Simple and Practical Algorithm for Differentially Private Data Release*. In: *NIPS*, Seiten 2348–2356, 2012.

[HMD13] HE, XI, ASHWIN MACHANAVAJJHALA und BOLIN DING: *Blowfish Privacy: Tuning Privacy-Utility Trade-offs using Policies*. CoRR, abs/1312.3913, 2013.

[HMD14] HANEY, SAMUEL, ASHWIN MACHANAVAJJHALA und BOLIN DING: *Answering Query Workloads with Optimal Error under Blowfish Privacy*. CoRR, abs/1404.3722, 2014.

[Hoe12] HOEPMAN, JAAP-HENK: *Privacy Design Strategies*. CoRR, abs/1210.6621, 2012.

[HQA+13] HEISE, ARVID, JORGE-ARNULFO QUIANÉ-RUIZ, ZIAWASCH ABEDJAN, ANJA JENTZSCH und FELIX NAUMANN: *Scalable Discovery of Unique Column Combinations*. PVLDB, 7(4):301–312, 2013.

[HS91] HEUER, ANDREAS und MARC H. SCHOLL: *Principles of Object-Oriented Query Languages*. In: APPELRATH, HANS-JÜRGEN (Herausgeber): *Datenbanksysteme in Büro, Technik und Wissenschaft, GI-Fachtagung, Kaiserslautern, 6.-8. März 1991, Proceedings*, Band 270 der Reihe *Informatik-Fachberichte*, Seiten 178–197. Springer, 1991.

[HS10] HINTOGLU, AYÇA AZGIN und YÜCEL SAYGIN: *Suppressing microdata to prevent classification based inference*. The VLDB Journal, 19(3):385–410, 2010.

[HS17] HECKMANN, DIRK und ALEXANDER SCHMID: *Rechtliche Aspekte automatisierter Systeme - Rechtskonforme Gestaltung unserer Zukunft*. Informatik Spektrum, 40(5):430–439, 2017.

[HT15] HARIRI, BABAK BAGHERI und VAL TANNEN: *Decidability of Equivalence of Aggregate Count-Distinct Queries*. CoRR, abs/1509.00100, 2015.

[JB14] JELASITY, MÁRK und KENNETH P BIRMAN: *Distributional Differential Privacy for Large-Scale Smart Metering*. In: *Proceedings of the 2nd ACM Workshop on Information Hiding and Multimedia Security*, Seiten 141–146. ACM, 2014.

[JKRP18] JINDAL, ALEKH, KONSTANTINOS KARANASOS, SRIRAM RAO und HIREN PATEL: *Selecting Subexpressions to Materialize at Datacenter Scale*. PVLDB, 11(7):800–812, 2018.

[JLY+21] JI, TIANXI, PAN LI, EMRE YILMAZ, ERMAN AYDAY, YANFANG YE und JINYUAN SUN: *Differentially Private Binary- and Matrix-Valued Data Query: An XOR Mechanism*. Proc. VLDB Endow., 14(5):849–862, 2021.

[JMS14] JAFER, YASSER, STAN MATWIN und MARINA SOKOLOVA: *Task Oriented Privacy Preserving Data Publishing Using Feature Selection*. In: *Canadian Conference on AI*, Band 8436 der Reihe *Lecture Notes in Computer Science*, Seiten 143–154. Springer, 2014.

[Joh16] JOHN, BODO: *Vergleichende Analyse von Datenschutzalgorithmen und -konzepten*. Bachelorarbeit, Universität Rostock, 2016. http://eprints.dbis.informatik.uni-rostock.de/820/.

[Kas21] KASELER, MAX TILMAN: *Eignung von funktionalen Abhängigkeiten und Inklusionsabhängigkeiten zur Reformulierung von Anfragen unter Datenschutzaspekten*. Bachelorarbeit, Universität Rostock, 2021. http://eprints.dbis.informatik.uni-rostock.de/1039/.

[KB96] KOHAVI, RONNY und BARRY BECKER: *Adult Data Set*. http://archive.ics.uci.edu/ml/datasets/Adult, 1996. Zuletzt aufgerufen am 16.12.2019.

[KBB14] KESSLER, STEPHAN, ERIK BUCHMANN und KLEMENS BÖHM: *Deploying and evaluating pufferfish privacy for smart meter data*. Technischer Bericht, Karlsruhe Institute of Technology, 2014.

[KHL13] KOOPS, BERT-JAAP, JAAP-HENK HOEPMAN und RONALD LEENES: *Open-source intelligence and privacy by design*. Computer Law & Security Review, 29(6):676–688, 2013.

[KHP15] KOUFOGIANNIS, FRAGKISKOS, SHUO HAN und GEORGE J. PAPPAS: *Gradual Release of Sensitive Data under Differential Privacy*. CoRR, abs/1504.00429, 2015.

[KK13] KEPSKI, MICHAL und BOGDAN KWOLEK: *Unobtrusive Fall Detection at Home Using Kinect Sensor*. In: *CAIP (1)*, Band 8047 der Reihe *Lecture Notes in Computer Science*, Seiten 457–464. Springer, 2013.

[KL51] KULLBACK, SOLOMON und RICHARD A LEIBLER: *On Information and Sufficiency*. The Annals of Mathematical Statistics, Seiten 79–86, 1951.

[Kle98] KLETTKE, MEIKE: *Akquisition von Integritätsbedingungen in Datenbanken*, Band 51 der Reihe *DISDBIS*. Infix Verlag, St. Augustin, Germany, 1998.

[Kle20] KLEIN, ERIC: *Parallele Anonymisierung von großen Datenbeständen*. Bachelorarbeit, Universität Rostock, 2020. http://eprints.dbis.informatik.uni-rostock.de/1012/.

[Klu82] KLUG, ANTHONY C.: *Equivalence of Relational Algebra and Relational Calculus Query Languages Having Aggregate Functions*. Journal of the ACM, 29(3):699–717, 1982.

[Klu88] KLUG, ANTHONY C.: *On Conjunctive Queries Containing Inequalities*. Journal of the ACM, 35(1):146–160, 1988.

[Klu17] KLUTH, JOHANN: *Eignung von Query Containment-Techniken zur Reformulierung von Anfragen bei der datenschutzfreundlichen Gestaltung von Informationssystemen*. Bachelorarbeit, Universität Rostock, 2017. http://eprints.dbis.informatik.uni-rostock.de/882/.

[KM14] KIFER, DANIEL und ASHWIN MACHANAVAJJHALA: *Pufferfish: A Framework for Mathematical Privacy Definitions*. ACM Transactions on Database Systems, 39(1):3:1–3:36, 2014.

[KMRS14] KOUTRIS, PARASCHOS, TOVA MILO, SUDEEPA ROY und DAN SUCIU: *Answering Conjunctive Queries with Inequalities*. CoRR, abs/1412.3869, 2014.

[KN18] KRUSE, SEBASTIAN und FELIX NAUMANN: *Efficient Discovery of Approximate Dependencies*. PVLDB, 11(7):759–772, 2018.

[Kol13] KOLAITIS, PHOKION G.: *The Query Containment Problem: Set Semantics vs. Bag Semantics*. In: BRAVO, LORETO und MAURIZIO LENZERINI (Herausgeber): *Proceedings of the 7th Alberto Mendelzon International Workshop on Foundations of Data Management, Puebla/Cholula, Mexico, May 21-23, 2013*, Band 1087 der Reihe *CEUR Workshop Proceedings*. CEUR-WS.org, 2013.

[Kol15] KOLKOWSKA, ELLA: *Privacy Principles in Design of Smart Homes Systems in Elderly Care*. In: *HCI (22)*, Band 9190 der Reihe *Lecture Notes in Computer Science*, Seiten 526–537. Springer, 2015.

[Kos19] KOSCHMIDER, AGNES: *Interview mit Thorsten Strufe zu Herausforderungen von Datenschutz und Datensicherheit für das Internet der Dinge*. Informatik Spektrum, 42(5):324–326, Oct 2019.

[Kru18] KRUSE, SEBASTIAN: *Scalable Data Profiling – Distributed Discovery and Analysis of Structural Metadata*. Doktorarbeit, Universität Potsdam, 2018.

[KSH14] KULKARNI, SAURABH, SHAYAN SAHA und RYLER HOCKENBURY: *Preserving Privacy in Sensor-Fog Networks*. In: *ICITST*, Seiten 96–99. IEEE, 2014.

[KSK05] KUNTSCHKE, RICHARD, BERNHARD STEGMAIER und ALFONS KEMPER: *Data Stream Sharing*. Technischer Bericht, Technische Universität München, Institut für Informatik, 2005.

[KV98] KOLAITIS, PHOKION G. und MOSHE Y. VARDI: *Conjunctive-Query Containment and Constraint Satisfaction*. 17. Symposium on Principles of Database Systems, Seattle, Seiten 205–213, 1998.

[KYHK13] KRÜGER, FRANK, KRISTINA YORDANOVA, ALBERT HEIN und THOMAS KIRSTE: *Plan Synthesis for Probabilistic Activity Recognition*. In: *ICAART (2)*, Seiten 283–288. SciTePress, 2013.

[Lan01] LANGHEINRICH, MARC: *Privacy by Design - Principles of Privacy-Aware Ubiquitous Systems*. In: ABOWD, GREGORY D., BARRY BRUMITT und STEVEN A. SHAFER (Herausgeber): *Ubicomp 2001: Proceedings of the Third International Conference on Ubiquitous Computing, Atlanta, Georgia, USA, September 30 - October 2, 2001*, Band 2201 der Reihe *Lecture Notes in Computer Science*, Seiten 273–291. Springer, 2001.

[Lan02] LAND MECKLENBURG-VORPOMMERN: *Gesetz zum Schutz des Bürgers bei der Verarbeitung seiner Daten (Landesdatenschutzgesetz - DSG M-V)*, 2002.

[Lan17] LANGMACHER, CHRISTIAN: *Vertikale Verteilung von SQL-Anfragen auf DBMS-Servern abnehmender Leistungsfähigkeit*. Masterarbeit, Universität Rostock, 2017. `http://eprints.dbis.informatik.uni-rostock.de/884/`.

[LCC14] LI, SHI, KYUNG CHOI und KIJOON CHAE: *OCPM: Ortho code privacy mechanism in smart grid using ring communication architecture*. Ad Hoc Networks, 22:93–108, 2014.

[LDR06] LEFEVRE, KRISTEN, DAVID J. DEWITT und RAGHU RAMAKRISHNAN: *Mondrian Multidimensional K-Anonymity*. In: *ICDE*, Seiten 25–35. IEEE Computer Society, 2006.

[LGKM11] LIU, YABING, P. KRISHNA GUMMADI, BALACHANDER KRISHNAMURTHY und ALAN MISLOVE: *Analyzing Facebook Privacy Settings: User Expectations vs. Reality*. In: THIRAN, PATRICK und WALTER WILLINGER (Herausgeber): *Proceedings of the 11th ACM SIGCOMM Internet Measurement Conference, IMC '11, Berlin, Germany, November 2, 2011*, Seiten 61–70. ACM, 2011.

[LHF18] LINDEN, THOMAS, HAMZA HARKOUS und KASSEM FAWAZ: *The Privacy Policy Landscape After the GDPR*. CoRR, abs/1809.08396, 2018.

[LLBW11] LI, JIUYONG, JIXUE LIU, MUZAMMIL M. BAIG und RAYMOND CHI-WING WONG: *Information based data anonymization for classification utility*. Data & Knowledge Engineering, 70(12):1030–1045, 2011.

[LLV07] LI, NINGHUI, TIANCHENG LI und SURESH VENKATASUBRAMANIAN: *t-Closeness: Privacy Beyond k-Anonymity and l-Diversity*. In: CHIRKOVA, RADA, ASUMAN DOGAC, M. TAMER ÖZSU und TIMOS K. SELLIS (Herausgeber): *Proceedings of the 23rd International Conference on Data Engineering*, Seiten 106–115. IEEE Computer Society, 2007.

[LLZM12] LI, TIANCHENG, NINGHUI LI, JIAN ZHANG und IAN MOLLOY: *Slicing: A new approach for privacy preserving data publishing*. IEEE Transactions on Knowledge and Data Engineering, 24(3):561–574, 2012.

[LMSS95] LEVY, ALON Y., ALBERTO O. MENDELZON, YEHOSHUA SAGIV und DIVESH SRIVASTAVA: *Answering Queries Using Views*. In: *PODS*, Seiten 95–104. ACM Press, 1995.

[LN07] LESER, ULF und FELIX NAUMANN: *Informationsintegration - Architekturen und Methoden zur Integration verteilter und heterogener Datenquellen*. dpunkt.verlag, 2007.

[LRO96a] LEVY, ALON, ANAND RAJARAMAN und JOANN ORDILLE: *Querying heterogeneous information sources using source descriptions*. Technischer Bericht, Stanford InfoLab, 1996.

[LRO96b] LEVY, ALON Y., ANAND RAJARAMAN und JOANN J. ORDILLE: *Querying Heterogeneous Information Sources Using Source Descriptions*. In: VIJAYARAMAN, T. M., ALEJANDRO P. BUCHMANN, C. MOHAN und NANDLAL L. SARDA (Herausgeber): *VLDB'96, Proceedings of 22th International Conference on Very Large Data Bases, September 3-6, 1996, Mumbai (Bombay), India*, Seiten 251–262. Morgan Kaufmann, 1996.

[LRU99] LEVY, ALON Y., ANAND RAJARAMAN und JEFFREY D. ULLMAN: *Answering Queries Using Limited External Query Processors*. Journal of Computer and System Sciences, 58(1):69–82, 1999.

[Lub00] LUBINSKI, ASTRID: *Replizieren und Reduzieren von Daten für ressourcenbegrenzte Umgebungen*. In: *Grundlagen von Datenbanken*, Seiten 66–70, 2000.

[MA14] MAX-PLANCK-INSTITUT FÜR SOFTWARESYSTEME und AIRCLOAK: *Aircloak Analytics: Anonymized User Data without Data Loss*. White Paper, Aircloak, 2014.

[Mar06] MARKEL, MIKE: *Safe harbor and privacy protection: A looming issue for IT professionals*. IEEE Transactions on Professional Communication, 49(1):1–11, 2006.

[MBK14] MOOS, DANIEL, SEBASTIAN BADER und THOMAS KIRSTE: *From Annotated Objects to Distributed Planning in Heterogeneous and Dynamic Environments*. In: STREITZ, NORBERT und PANOS MARKOPOULOS (Herausgeber): *Distributed, Ambient, and Pervasive Interactions - Second International Conference, DAPI 2014, Held as Part of HCI Interational 2014, Heraklion, Crete, Greece, June 22-27, 2014. Proceedings*, Band 8530 der Reihe *Lecture Notes in Computer Science*, Seiten 462–473. Springer, 2014.

[MC04] MILNE, GEORGE R und MARY J CULNAN: *Strategies for reducing online privacy risks: Why consumers read (or don't read) online privacy notices*. Journal of Interactive Marketing, 18(3):15–29, 2004.

[MC15] MOTTI, VIVIAN GENARO und KELLY CAINE: *Users' Privacy Concerns About Wearables - Impact of Form Factor, Sensors and Type of Data Collected*. In: *Financial Cryptography Workshops*, Band 8976 der Reihe *Lecture Notes in Computer Science*, Seiten 231–244. Springer, 2015.

[Mei03] MEINDL, J.D.: *Beyond Moore's Law: the interconnect era*. Computing in Science Engineering, 5(1):20–24, 2003.

[Mei14] MEIER, CHRISTIAN J.: *Der Daten-Schutzmantel*. MaxPlanckForschung, 14(1):18–23, 2014.

[Mei15] MEIER, MARC-ERIC: *Implementierung eines Data-Mining-Verfahrens für adaptive Datenquellen*. Bachelorarbeit, Universität Rostock, 2015. http://eprints.dbis.informatik.uni-rostock.de/128/.

[Mel92] MELTON, JIM: *(Second Informal Review Draft) ISO/IEC 9075:1992, Database Languages – SQL*. Technischer Bericht, ISO/IEC, 1992.

[Mel03] MELTON, JIM: *ISO/IEC DIS 9075:2003 Information Technology - Database Languages – SQL*. Technischer Bericht, ISO/IEC, 2003.

[Mel16] MELTON, JIM: *ISO/IEC DIS 9075:2016 Information Technology - Database Languages – SQL*. Technischer Bericht, ISO/IEC, 2016.

[Mey92] MEYER, BERTRAND: *Applying Design by Contract*. IEEE Computer, 25(10):40–51, 1992.

[MFB02] MÜHL, GERO, LUDGER FIEGE und ALEJANDRO P. BUCHMANN: *Filter Similarities in Content-Based Publish/Subscribe Systems*. In: *ARCS*, Band 2299 der Reihe *Lecture Notes in Computer Science*, Seiten 224–240. Springer, 2002.

[MFHH02] MADDEN, SAMUEL, MICHAEL J. FRANKLIN, JOSEPH M. HELLERSTEIN und WEI HONG: *TAG: A Tiny AGgregation Service for Ad-Hoc Sensor Networks*. In: *5th Annual Symposium on Operating Systems Design and Implementation (OSDI)*. USENIX Association, 2002.

[MFHH05] MADDEN, SAMUEL, MICHAEL J. FRANKLIN, JOSEPH M. HELLERSTEIN und WEI HONG: *TinyDB: An Acquisitional Query Processing System for Sensor Networks*. ACM Transactions on Database Systems, 30(1):122–173, 2005.

[MGMR02] MELNIK, SERGEY, HECTOR GARCIA-MOLINA und ERHARD RAHM: *Similarity Flooding: A Versatile Graph Matching Algorithm and its Application to Schema Matching*. In: *Proceedings of the 18th International Conference on Data Engineering*, Seiten 117–128. IEEE, 2002.

[MH15a] MARTEN, DENNIS und ANDREAS HEUER: *A framework for self-managing database support and parallel computing for assistive systems*. In: MAKEDON, FILLIA (Herausgeber): *Proceedings of the 8th ACM International Conference on PErvasive Technologies Related to Assistive Environments, PETRA 2015, Corfu, Greece, July 1-3, 2015*, Seiten 25:1–25:4. ACM, 2015.

[MH15b] MARTEN, DENNIS und ANDREAS HEUER: *Transparente Datenbankunterstützung für Analysen auf Big Data*. In: SAAKE, GUNTER, DAVID BRONESKE, SEBASTIAN DOROK und ANDREAS MEISTER (Herausgeber): *Proceedings of the 27th GI-Workshop Grundlagen von Datenbanken, Gommern, Germany, May 26-29, 2015.*, Band 1366 der Reihe *CEUR Workshop Proceedings*, Seiten 36–41. CEUR-WS.org, 2015.

[MH17] MARTEN, DENNIS und ANDREAS HEUER: *Machine Learning on Large Databases: Transforming Hidden Markov Models to SQL Statements*. OJDB, 4(1):22–42, 2017.

[MHF03] MILLSTEIN, TODD D., ALON Y. HALEVY und MARC FRIEDMAN: *Query containment for data integration systems*. Journal of Computer and System Sciences, 66(1):20–39, 2003.

[MKGV07] MACHANAVAJJHALA, ASHWIN, DANIEL KIFER, JOHANNES GEHRKE und MUTHURAMAKRISHNAN VENKITASUBRAMANIAM: *l-Diversity: Privacy Beyond k-Anonymity*. ACM Transactions on Knowledge Discovery from Data (TKDD), 1(1):3, 2007.

[MKSS95] MICHAELIS, J, A KRTSCHIL, I SCHMIDTMANN und J SCHÜZ: *Bundeskrebsregistergesetz und Stand der Pilotstudie Krebsregister Rheinland-Pfalz*. Ärzteblatt Rheinland-Pfalz, 48:71–4, 1995.

[Mül16] MÜLLER, MARTIN: *Generalisierung von Attributen in Relationalen Datenbanken*. Technischer Bericht, Fakultät für Informatik und Elektrotechnik der Universität Rostock, 2016.

[Mün16] MÜNCH, ISABEL (Herausgeber): *IT-Grundschutz-Kataloge, 15. Ergänzungslieferung*. Bundesamt für Sicherheit in der Informationstechnik, 2016.

[Moo65] MOORE, GORDON E.: *Cramming more components onto integrated circuits*. Electronics, 38(8), 1965.

[MSD06] MURALIDHAR, KRISHNAMURTY, RATHINDRA SARATHY und RAMESH A. DANDEKAR: *Why Swap When You Can Shuffle? A Comparison of the Proximity Swap and Data Shuffle for Numeric Data*. In: *Privacy in Statistical Databases*, Band 4302 der Reihe *Lecture Notes in Computer Science*, Seiten 164–176. Springer, 2006.

[MSS14] MOHAMMED, REDWAN ABDO A., LARS SCHWABE und OLIVER G. STAADT: *Gaze Location Prediction with Depth Features as Auxiliary Information*. In: KUROSU, MASAAKI (Herausgeber): *Human-Computer Interaction. Advanced Interaction Modalities and Techniques - 16th International Conference, HCI International 2014, Heraklion, Crete, Greece, June 22-27, 2014, Proceedings, Part II*, Band 8511 der Reihe *Lecture Notes in Computer Science*, Seiten 281–292. Springer, 2014.

[MTKW18] MAIER, DAVID, K. TUNCAY TEKLE, MICHAEL KIFER und DAVID S. WARREN: *Datalog: Concepts, History, and Outlook*. In: KIFER, MICHAEL und YANHONG ANNIE LIU (Herausgeber): *Declarative Logic Programming: Theory, Systems, and Applications*, Seiten 3–100. Association for Computing Machinery and Morgan & Claypool, New York, NY, USA, 2018.

[MTV97] MANNILA, HEIKKI, HANNU TOIVONEN und A INKERI VERKAMO: *Discovery of Frequent Episodes in Event Sequences*. Data Mining and Knowledge Discovery, 1(3):259–289, 1997.

[Müh02] MÜHL, GERO: *Large-Scale Content-Based Publish/Subscribe Systems*. Doktorarbeit, Darmstadt University of Technology, Germany, 2002.

[MW04] MEYERSON, ADAM und RYAN WILLIAMS: *On the Complexity of Optimal K-Anonymity*. In: *PODS*, Seiten 223–228. ACM, 2004.

[MWRF13] MANZESCHKE, ARNE, KARSTEN WEBER, ELISABETH ROTHER und HEINER FANGERAU: *Ethische Fragen im Bereich Altersgerechter Assistenzsysteme*. Bundesministerium für Bildung und Forschung, 2013.

[MX07] MOTWANI, RAJEEV und YING XU: *Efficient Algorithms for Masking and Finding Quasi-Identifiers*. Technischer Bericht, Stanford University, Department of Computer Science, 2007.

[NBL+10] NI, QUN, ELISA BERTINO, JORGE LOBO, CAROLYN BRODIE, CLARE-MARIE KARAT, JOHN KARAT und ALBERTO TROMBETTA: *Privacy-Aware Role-Based Access Control*. ACM Transactions on Information and System Security, 13(3):24:1–24:31, 2010.

[Nia78] NIAMIR, BAHRAM: *Attribute Partitioning in a Self-Adaptive Relational Database System.* Technischer Bericht, MIT Cambridge Lab for Computer Science, 1978.

[Nie13] NIELSEN, JAN HENDRIK: *Verteilte Anonymisierung von vertikal partitionierten Daten*. Diplomarbeit, Humboldt-Universität zu Berlin, 2013.

[NK15] NYOLT, MARTIN und THOMAS KIRSTE: *On Resampling for Bayesian Filters in Discrete State Spaces*. In: *27th IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2015, Vietri sul Mare, Italy, November 9-11, 2015*, Seiten 526–533. IEEE Computer Society, 2015.

[NKY+15] NYOLT, MARTIN, FRANK KRÜGER, KRISTINA YORDANOVA, ALBERT HEIN und THOMAS KIRSTE: *Marginal filtering in large state spaces*. International Journal of Approximate Reasoning, 61:16–32, 2015.

[NYK15] NYOLT, MARTIN, KRISTINA YORDANOVA und THOMAS KIRSTE: *Checking Models for Activity Recognition*. In: *ICAART (2)*, Seiten 497–502. SciTePress, 2015.

[Nyo19] NYOLT, MARTIN: *Efficient human situation recognition using Sequential Monte Carlo in discrete state spaces*. Doktorarbeit, Universität Rostock, 2019.

[OAF+19] OWAIDA, MUHSEN, GUSTAVO ALONSO, LAURA FOGLIARINI, ANTHONY HOCK-KOON und PIERRE-ETIENNE MELET: *Lowering the Latency of Data Processing Pipelines Through FPGA based Hardware Acceleration*. Proceedings of the VLDB Endowment, 13(1):71–85, 2019.

[OAS13] OASIS: *eXtensible Access Control Markup Language*. https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml, 2013. Zuletzt aufgerufen am 04.02.2014.

[Ope17a] OPENFOG CONSORTIUM: *The 8 Pillars of the OpenFog Reference Architecture*. https://www.openfogconsortium.org/wp-content/uploads/OpenFog-Pillars-10-page-summary.pdf, 2017. Zuletzt aufgerufen am 01.08.2018.

[Ope17b] OPENFOG CONSORTIUM: *The OpenFog Consortium Reference Architecture: Executive Summary*. https://www.openfogconsortium.org/wp-content/uploads/OpenFog-Reference-Architecture-Executive-Summary.pdf, 2017. Zuletzt aufgerufen am 01.08.2018.

[Ora17] ORACLE: *Oracle Berkeley Database Products*. https://www.oracle.com/technetwork/database/database-technologies/berkeleydb/learnmore/berkeley-db-family-datasheet-132751.pdf, 2017. Zuletzt aufgerufen am 14.01.2019.

[Ora18] ORACLE: *Oracle Berkeley Database 18.1*. https://www.oracle.com/technetwork/database/database-technologies/berkeleydb/berkeley-db-datasheet-132390.pdf, 2018. Zuletzt aufgerufen am 14.01.2019.

[Org13] ORGANISATION FOR ECONOMIC CO-OPERATION AND DEVELOPMENT: *Guidelines on the Protection of Privacy and Transborder Flows of Personal Data*. OECD, 2013.

[Orw49] ORWELL, GEORGE: *1984*. Ullstein Buchverlage GmbH, Berlin, 35. Auflage, 1949.

[OW08] OLLIS, JEFFREY D und MICHAEL R WIMBERLY: *System and Method For Controlling Devices in a Home-Automation Network*, Oktober 30 2008. US Patent App. 11/742,237.

[OYM+13] OGE, YASIN, MASATO YOSHIMI, TAKEFUMI MIYOSHI, HIDEYUKI KAWASHIMA, HIDETSUGU IRIE und TSUTOMU YOSHINAGA: *An Efficient and Scalable Implementation of Sliding-Window Aggregate Operator on FPGA*. In: *CANDAR*, Seiten 112–121. IEEE Computer Society, 2013.

[PGH98] PAPAKONSTANTINOU, YANNIS, ASHISH GUPTA und LAURA HAAS: *Capabilities-Based Query Rewriting in Mediator Systems*. Distributed and Parallel Databases, 6(1):73–110, 1998.

[PH01] POTTINGER, RACHEL und ALON HALEVY: *MiniCon: A scalable algorithm for answering queries using views*. The VLDB Journal - The International Journal on Very Large Data Bases, 10(2-3):182–198, 2001.

[PKLK14] PRASSER, FABIAN, FLORIAN KOHLMAYER, RONALD LAUTENSCHLAEGER und KLAUS A KUHN: *ARX – A Comprehensive Tool for Anonymizing Biomedical Data*. In: *AMIA Annual Symposium Proceedings*, Band 2014, Seite 984. American Medical Informatics Association, 2014.

[PL00] POTTINGER, RACHEL und ALON Y. LEVY: *A Scalable Algorithm for Answering Queries Using Views*. In: *VLDB*, Seiten 484–495. Morgan Kaufmann, 2000.

[PN17] PAPENBROCK, THORSTEN und FELIX NAUMANN: *A Hybrid Approach for Efficient Unique Column Combination Discovery*. In: *BTW*, Band P-265 der Reihe *LNI*, Seiten 195–204. GI, 2017.

[Poh18] POHLE, JÖRG: *Datenschutz und Technikgestaltung – Geschichte und Theorie des Datenschutzes aus informatischer Sicht und Folgerungen für die Technikgestaltung*. Doktorarbeit, Humboldt-Universität zu Berlin, 2018.

[Pop01] POPA, LUCIAN: *Object/relational query optimization with chase and backchase*. Doktorarbeit, University of Pennsylvania, 2001.

[PRS+17] PRABHAVALKAR, ROHIT, KANISHKA RAO, TARA N. SAINATH, BO LI, LEIF JOHNSON und NAVDEEP JAITLY: *A Comparison of Sequence-to-Sequence Models for Speech Recognition*. In: LACERDA, FRANCISCO (Herausgeber): *Interspeech 2017, 18th Annual Conference of the International Speech Communication Association, Stockholm, Sweden, August 20-24, 2017*, Seiten 939–943. ISCA, 2017.

[PRZB11] POPA, RALUCA A., CATHERINE M. S. REDFIELD, NICKOLAI ZELDOVICH und HARI BALAKRISHNAN: *CryptDB: Protecting Confidentiality with Encrypted Query Processing*. In: *SOSP*, Seiten 85–100. ACM, 2011.

[PS17] PETRLIC, RONALD und CHRISTOPH SORGE: *Datenschutz: Einführung in technischen Datenschutz, Datenschutzrecht und angewandte Kryptographie*. Springer-Verlag, 2017.

[PSHSG21] PERRY, JULIA, SILKE SCHICKTANZ, BENJAMIN HERTEN und SCOTT STOCK GISSENDANNER: *Ethische und soziale Aspekte der Demenzforschung und -versorgung*. Universitätsverlag Göttingen, Göttingen, 2021.

[Qia96] QIAN, XIAOLEI: *Query Folding*. In: *ICDE*, Seiten 48–55. IEEE Computer Society, 1996.

[Ran13] RANDELSHOFER, WERNER: *Visualization of large tree structures*. `http://www.randelshofer.ch/treeviz/`, 2013. Zuletzt aufgerufen am 17.03.2021.

[REC15] REINHARDT, ANDREAS, FRANK ENGLERT und DELPHINE CHRISTIN: *Averting the privacy Risks of Smart Metering by Local Data Preprocessing*. Pervasive and Mobile Computing, 16:171–183, 2015.

[RG98] RICHTERS, MARK und MARTIN GOGOLLA: *On Formalizing the UML Object Constraint Language OCL*. In: LING, TOK WANG, SUDHA RAM und MONG-LI LEE (Herausgeber): *Conceptual Modeling - ER '98, 17th International Conference on Conceptual Modeling, Singapore, November 16-19, 1998, Proceedings*, Band 1507 der Reihe *Lecture Notes in Computer Science*, Seiten 449–464. Springer, 1998.

[RG17] RUIZ, ANTONIO RAMÓN JIMÉNEZ und FERNANDO SECO GRANJA: *Comparing Ubisense, BeSpoon, and DecaWave UWB Location Systems: Indoor Performance Analysis*. IEEE Transactions on Instrumentation and Measurement, 66(8):2106–2117, 2017.

[RGM+12] RAU, HENRIETTE, JACOB GRIEGER, CHRISTIAN MARZAHL, PETER PENNDORF und MARTIN STAEMMLER: *Unobtrusive Fall Detection and Prevention: Extending From a Prototype Test to a Pilot Trial*. In: *GI-Jahrestagung*, Band 208 der Reihe *LNI*, Seiten 1456–1473. GI, 2012.

[RLAS07] ROSENMÜLLER, MARKO, THOMAS LEICH, SVEN APEL und GUNTER SAAKE: *Von Mini- über Micro- bis zu Nano-DBMS: Datenhaltung in eingebetteten Systemen*. Datenbank-Spektrum, 7(20):33–47, 2007.

[RLRG16] RASSMANN, GUNNAR, CHRISTIAN LANGMACHER, IRENE MARTIN RODRIGUEZ und JOHANNES GOLTZ: *Projektarbeit: Privatheit durch Anonymisierung von Anfragen an Datenbanken*, 2016. `http://eprints.dbis.informatik.uni-rostock.de/1038/`.

[RMPADF13] REBOLLO-MONEDERO, DAVID, JAVIER PARRA-ARNAU, CLAUDIA DIAZ und JORDI FORNÉ: *On the measurement of privacy as an attacker's estimation error*. International Journal of Information Security, 12(2):129–149, 2013.

[Ros19] ROSE, FLORIAN: *Berechnung von Quasi-Identifikatoren nach Verbundoperationen*. Bachelorarbeit, Universität Rostock, 2019. http://eprints.dbis.informatik.uni-rostock.de/988/.

[Sal16] SALEH, ASEM: *Daten- und Dimensionsreduktionstechniken für Big Data Analytics*. Masterarbeit, Universität Rostock, 2016. http://eprints.dbis.informatik.uni-rostock.de/808/.

[Sam01] SAMARATI, PIERANGELA: *Protecting Respondent's Identities in Microdata Release*. IEEE Transactions on Knowledge and Data Engineering, 13(6):1010–1027, 2001.

[SBCD09] SATYANARAYANAN, MAHADEV, PARAMVIR BAHL, RAMÓN CÁCERES und NIGEL DAVIES: *The Case for VM-Based Cloudlets in Mobile Computing*. IEEE Pervasive Computing, 8(4):14–23, 2009.

[SBF+16] SCHMIDT, WERNER, STEPHAN BORGERT, ALBERT FLEISCHMANN, LUTZ HEUSER, CHRISTIAN MÜLLER und MAX MÜHLHÄUSER: *Digitale Mehrwertdienste in Smart Cities am Beispiel Verkehr*, Seiten 255–274. Springer Fachmedien Wiesbaden, Wiesbaden, 2016.

[SBHR06] SISMANIS, YANNIS, PAUL BROWN, PETER J. HAAS und BERTHOLD REINWALD: *GORDIAN: Efficient and Scalable Discovery of Composite Keys*. In: *VLDB*, Seiten 691–702. ACM, 2006.

[SBKN16] STEUER, SIMON, ABOUBAKR BENABBAS, NASR KASRIN und DANIELA NICKLAS: *Challenges and Design Goals for an Architecture of a Privacy-preserving Smart City Lab*. Datenbank-Spektrum, 16(2):147–156, 2016.

[SBMH09] SCHULZ, ARNE, HANNAH BAUMGARTNER, FRERK MÜLLER und ANDREAS HEIN: *Eine Multimediazentrale als Hörunterstützung im häuslichen Umfeld*. In: FISCHER, STEFAN, ERIK MAEHLE und RÜDIGER REISCHUK (Herausgeber): *39. Jahrestagung der Gesellschaft für Informatik, Im Focus das Leben, INFORMATIK 2009, Lübeck, Germany, September 28 - October 2, 2009, Proceedings*, Band P-154 der Reihe *LNI*, Seiten 910–924. GI, 2009.

[SCD14] SEPEHRI, MARYAM, STELVIO CIMATO und ERNESTO DAMIANI: *Privacy-Preserving Query Processing by Multi-Party Computation*. The Computer Journal, 58(10):2195–2212, 2014.

[SCDFSM14] SORIA-COMAS, JORDI, JOSEP DOMINGO-FERRER, DAVID SÁNCHEZ und SERGIO MARTÍNEZ: *Enhancing data utility in differential privacy via microaggregation-based k-anonymity*. VLDB Journal, 2014. DOI: 10.1007/s00778-014-0351-4.

[Sch10] SCHAAR, PETER: *Privacy by Design*. Identity in the Information Society, 3(2):267–274, 2010.

[Sch11] SCHÜLER, PETER: *Aus für Elena*. c't, 29(17):8, 2011.

[Sch13] SCHEFFLER, THOMAS: *Privacy Enforcement with Data Owner-defined Policies*. Doktorarbeit, Universität Potsdam, 2013.

[Sch16] SCHMUDE, MAIK: *Systematische Untersuchung der Anfragekapazität verschiedener DBMS*. Bachelorarbeit, Universität Rostock, 2016. http://eprints.dbis.informatik.uni-rostock.de/821/.

[Sch18] SCHMUDE, MAIK: *Erkennung von Ausreißern durch Row Pattern Matching*. Masterarbeit, Universität Rostock, 2018. http://eprints.dbis.informatik.uni-rostock.de/958/.

[SD16] SHI, WEISONG und SCHAHRAM DUSTDAR: *The Promise of Edge Computing*. IEEE Computer, 49(5):78–81, 2016.

[SDFB09] SCHERING, ALF-CHRISTIAN, MARTIN DUEFFER, ANDREAS FINGER und ILVIO BRUDER: *A Mobile Tourist Assistance and Recommendation System Based on Complex Networks*. In: WANG, JUN, SHI ZHOU und DELL ZHANG (Herausgeber): *Proceeding of the ACM First International Workshop on Complex Networks Meet Information & Knowledge Management, CIKM-CNIKM 2009, Hong Kong, China, November 6, 2009*, Seiten 81–84. ACM, 2009.

[Sei17] SEIB, ENRICO: *Automatisches Deployment und adaptive Platzierung ubiquitärer Anwendungen*. Doktorarbeit, Universität Rostock, 2017.

[Sep13] SEPEHRI, MARYAM: *Privacy-Preserving Query Processing by Multi-Party Computation and Encrypted Data Outsourcing*. Doktorarbeit, Università Degli Studi di Milano, 2013.

[Sha48] SHANNON, CLAUDE ELWOOD: *A Mathematical Theory of Communication*. Bell System Technical Journal, 27(3):379–423, 1948.

[Sil21] SILVA, JOÃO ANDRÉ ALMEIDA: *Data Storage and Dissemination in Pervasive Edge Computing Environments*. Doktorarbeit, Universidade Nova de Lisboa, 2021.

[Sim13] SIMON, H: *Why Machines Learn?* Machine Learning: An Artificial Intelligence Approach, 1, 2013.

[SJMK21] SHAOWANG, TED, NILESH JAIN, DENNIS MATTHEWS und SANJAY KRISHNAN: *Declarative Data Serving: The Future of Machine Learning Inference on the Edge*. Proc. VLDB Endow., 14(11):2555–2562, 2021.

[SL08] STEINHAGE, AXEL und CHRISTL LAUTERBACH: *Sensfloor (r): Ein AAL Sensorsystem für Sicherheit, Homecare und Komfort*. Ambient Assisted Living-AAL, 2008.

[SMS21] SILVA, PAULO, EDMUNDO MONTEIRO und PAULO SIMÕES: *Privacy in the Cloud: A Survey of Existing Solutions and Research Challenges*. IEEE Access, 9:10473–10497, 2021.

[SS98] SAMARATI, PIERANGELA und LATANYA SWEENEY: *Protecting Privacy when Disclosing Information: k-Anonymity and Its Enforcement through Generalization and Suppression*. Technischer Bericht, SRI International, 1998.

[SSH11] SAAKE, GUNTER, KAI-UWE SATTLER und ANDREAS HEUER: *Datenbanken – Implementierungstechniken, 3. Auflage*. MITP, 2011.

[SSH18] SAAKE, GUNTER, KAI-UWE SATTLER und ANDREAS HEUER: *Datenbanken – Konzepte und Sprachen, 6. Auflage*. MITP, 2018.

[Sta18] STATISTISCHES BUNDESAMT: *Verteilung der Fehlverhalten von Fahrzeugführern bei Unfällen im Straßenverkehr mit Personenschaden in den Jahren 2016 und 2017 nach Ursache*. Statista - Das Statistik-Portal, `https://de.statista.com/statistik/daten/studie/2110/umfrage/fehlverhalten-von-fahrzeugfuehrern-mit-unfallfolge/`, 2018. Zuletzt aufgerufen am 28.09.2018.

[Ste16] STEINKE, GUIDO: *Die Pflegedokumentation - (k)ein Geheimpapier? Wer darf Einblick nehmen?* Bundesinteressenvertretung für alte und pflegebetroffene Menschen (BIVA) e.V., 2016.

[STE21] SVERKO, MLADEN, NIKOLA TANKOVIC und DARKO ETINGER: *Dew Computing in Industrial Automation: Applying Machine Learning for Process Control*. In: *2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC)*, Seiten 1789–1794. IEEE, 2021.

[SW74] STONEBRAKER, MICHAEL und EUGENE WONG: *Access control in a relational data base management system by query modification*. In: *ACM Annual Conference (1)*, Seiten 180–186. ACM, 1974.

[Swe02] SWEENEY, LATANYA: *k-anonymity: A model for protecting privacy*. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 10(05):557–570, 2002.

[Tan09] TANENBAUM, ANDREW S.: *Moderne Betriebssysteme, 3. Auflage*. Pearson Studium, 2009.

[TDK+17] TALEB, TARIK, SUNNY DUTTA, ADLEN KSENTINI, MUDDESAR IQBAL und HANNU FLINCK: *Mobile Edge Computing Potential in Making Cities Smarter*. IEEE Communications Magazine, 55(3):38–43, 2017.

[TH20] TANG, LI und HAIBO HU: *OHEA: Secure Data Aggregation in Wireless Sensor Networks against Untrusted Sensors*. In: *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, Seiten 1425–1434, 2020.

[TL17] TITOK, ROMAN und ALEX LYMAR: *Differential Privacy – Abschlussbericht*. Studentisches Projekt, Universität Rostock, 2017.

[TNP14] TO, QUOC-CUONG, BENJAMIN NGUYEN und PHILIPPE PUCHERAL: *Privacy-Preserving Query Execution using a Decentralized Architecture and Tamper Resistant Hardware*. In: *EDBT*, Seiten 487–498. OpenProceedings.org, 2014.

[TNS13] TNS INFRATEST: *Zukunftspfade Digitales Deutschland 2020*. Bundesministerium des Innern, 2013.

[TT14] TURAN, UGUR und ISMAIL HAKKI TOROSLU: *Privacy Preserving Secure Decomposition Algorithm for Attribute Based Access Control Mechanism*. CoRR, abs/1402.5742, 2014.

[TTG14] THEOHARIDOU, MARIANTHI, NIKOS TSALIS und DIMITRIS GRITZALIS: *Smart Home Solutions for Healthcare: Privacy in Ubiquitous Computing Infrastructures*. Handbook of Smart Homes, Health Care and Well-Being, 2014.

[Tür99] TÜRKER, CAN: *Semantic Integrity Constraints in Federated Database Schemata*, Band 63 der Reihe *DISDBIS*. Infix Verlag, St. Augustin, Germany, 1999.

[Ull82] ULLMAN, JEFFREY D.: *Principles of Database Systems, 2nd Edition*. Computer Science Press, 1982.

[ULS08] ULLMAN, JEFFREY D, MONICA S LAM und RAVI SETHI: *Compiler: Prinzipien, Techniken und Werkzeuge*. Pearson Deutschland GmbH, 2008.

[Vas09] VASSALOS, VASILIS: *Answering Queries Using Views*. In: *Encyclopedia of Database Systems*. Springer US, 2009.

[VP00] VASSALOS, VASILIS und YANNIS PAPAKONSTANTINOU: *Expressive Capabilities Description Languages and Query Rewriting Algorithms*. Journal of Logic Programming, 43(1):75–122, 2000.

[VvdB18] VOIGT, PAUL und AXEL VON DEM BUSSCHE: *EU-Datenschutz-Grundverordnung (DSGVO): Praktikerhandbuch*. Springer-Verlag, 2018.

[W3C03] W3C: *Enterprise Privacy Authorization Language*. `http://www.w3.org/Submission/2003/SUBM-EPAL-20031110/`, 2003. Zuletzt aufgerufen am 04.02.2014.

[W3C07] W3C: *Platform for Privacy Preferences (P3P) Project*. `http://www.w3.org/P3P/`, 2007. Zuletzt aufgerufen am 04.02.2014.

[WA16] WEISS, MARTIN A und KRISTIN ARCHICK: *US-EU Data Privacy: From Safe Harbor to Privacy Shield*, 2016.

[Wan16] WANG, YINGWEI: *Definition and Categorization of Dew Computing*. Open Journal of Cloud Computing (OJCC), 3(1):1–7, 2016.

[Wei91] WEISER, MARK: *The Computer for the 21st Century*. Scientific American, 265(3):94–104, 1991.

[WH95] WEIK, THOMAS und ANDREAS HEUER: *An Algorithm for the Analysis of Termination of Large Trigger Sets in an OODBMS*. In: *ARTDB*, Workshops in Computing, Seiten 170–189. Springer, 1995.

[WH16] WÄCHTER, FELIX und MARTIN HAUFSCHILD: *Projektarbeit: PArADISE Sensor- & Appliance-Level*, 2016. `http://eprints.dbis.informatik.uni-rostock.de/823/`.

[WL03] WEI, FANG und GEORG LAUSEN: *Containment of Conjunctive Queries with Safe Negation*. In: CALVANESE, DIEGO, MAURIZIO LENZERINI und RAJEEV MOTWANI (Herausgeber): *Database Theory - ICDT 2003, 9th International Conference, Siena, Italy, January 8-10, 2003, Proceedings*, Band 2572 der Reihe *Lecture Notes in Computer Science*, Seiten 343–357. Springer, 2003.

[WLFW06] WONG, RAYMOND CHI-WING, JIUYONG LI, ADA WAI-CHEE FU und KE WANG: *(α, K)-anonymity: An Enhanced K-anonymity Model for Privacy Preserving Data Publishing*. In: *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, Seiten 754–759, New York, NY, USA, 2006. ACM.

[WLFW09] WONG, RAYMOND, JIUYONG LI, ADA FU und KE WANG: *(α, k)-anonymous data publishing*. Journal of Intelligent Information Systems, 33(2):209–234, 10 2009.

[WLLP01] WARNEKE, BRETT, MATT LAST, BRIAN LIEBOWITZ und KRISTOFER S. J. PISTER: *Smart Dust: Communicating with a Cubic-Millimeter Computer*. IEEE Computer, 34(1):44–51, 2001.

[WMYR14] WON, JONGHO, CHRIS Y. T. MA, DAVID K. Y. YAU und NAGESWARA S. V. RAO: *Proactive Fault-Tolerant Aggregation Protocol for Privacy-Assured Smart Metering*. In: *INFOCOM*, Seiten 2804–2812. IEEE, 2014.

[WMYR16] WON, JONGHO, CHRIS Y. T. MA, DAVID K. Y. YAU und NAGESWARA S. V. RAO: *Privacy-Assured Aggregation Protocol for Smart Metering: A Proactive Fault-Tolerant Approach*. IEEE/ACM Transactions on Networking, 24(3):1661–1674, 2016.

[WR67] WESTIN, ALAN F und OSCAR M RUEBHAUSEN: *Privacy and Freedom*, Band 1. Atheneum New York, 1967.

[WR17] WIBBE, CHRISTIAN und DIRK ROHDE: *Industrie 4.0 im automobilen Umfeld*. In: *Automobillogistik*, Seiten 37–52. Springer, 2017.

[WST13] WAGNER, BENJAMIN, BJORN STRIEBING und DIRK TIMMERMANN: *A System for Live Localization In Smart Environments*. In: *ICNSC*, Seiten 684–689. IEEE, 2013.

[WTRK12] WAGNER, BENJAMIN, DIRK TIMMERMANN, GERNOT RUSCHER und THOMAS KIRSTE: *Device-Free User Localization Utilizing Artificial Neural Networks and Passive RFID*. In: *UPINLBS*, Seiten 1–7. IEEE, 2012.

[Yam17] YAMAUCHI, KAZUNORI (Herausgeber): *Gran Turismo APEX*. Gran Turismo Magazine. Sony Interactive Entertainment Inc., 2017.

[Yao86] YAO, ANDREW CHI-CHIH: *How to Generate and Exchange Secrets (Extended Abstract)*. In: *FOCS*, Seiten 162–167. IEEE Computer Society, 1986.

[YLX14] YAO, YONGLEI, JINGFA LIU und NEAL N. XIONG: *Privacy-Preserving Data Aggregation in Two-Tiered Wireless Sensor Networks with Mobile Nodes*. Sensors, 14(11):21174–21194, 2014.

[Yor14] YORDANOVA, KRISTINA: *Methods for Engineering Symbolic Human Behaviour Models for Activity Recognition*. Doktorarbeit, Universität Rostock, 2014.

[YP12] YAKUT, IBRAHIM und HUSEYIN POLAT: *Privacy-preserving hybrid collaborative filtering on cross distributed data*. Knowledge and Information Systems, 30(2):405–433, 2012.

[Yu21] YU, YUAN-CHIH: *A Dew Computing Architecture for Smart Parking System with Cloud Image Recognition Service*. In: *2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC)*, Seiten 1805–1809. IEEE, 2021.

[YWL+20] YANG, JIANYU, TIANHAO WANG, NINGHUI LI, XIANG CHENG und SEN SU: *Answering Multi-Dimensional Range Queries under Local Differential Privacy*. Proc. VLDB Endow., 14(3):378–390, 2020.

[ZAB14] ZAKERZADEH, HESSAM, CHARU C. AGGARWAL und KEN BARKER: *Towards Breaking the Curse of Dimensionality for High-Dimensional Privacy: An Extended Version*. CoRR, abs/1401.1174, 2014.

[ZL11] ZU, CHEN-XI und HONG LI: *Thermodynamic analysis on energy densities of batteries*. Energy & Environmental Science, 4(8):2614–2624, 2011.

[ZSA09] ZHOU, YONGLUAN, ALI SALEHI und KARL ABERER: *Scalable Delivery of Stream Query Results*. PVLDB, 2(1):49–60, 2009.

[ZYI+17] ZANIOLO, CARLO, MOHAN YANG, MATTEO INTERLANDI, ARIYAM DAS, ALEXANDER SHKAPSKY und TYSON CONDIE: *Fixpoint Semantics and Optimization of Recursive Datalog Programs with Aggregates*. CoRR, abs/1707.05681, 2017.

[ZYWZ14] ZHANG, HAO, NENGHAI YU, YONGGANG WEN und WEIMING ZHANG: *Towards Optimal Noise Distribution for Privacy Preserving in Data Aggregation*. Computers & Security, 45:210–230, 2014.

Selbstständigkeitserklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe; die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht. Die Arbeit wurde weder im Inland noch im Ausland in gleicher oder ähnlicher Form einer Prüfungsbehörde zur Erlangung eines akademischen Grades vorgelegt.

Rostock, 02.03.2022

Lebenslauf

Persönliche Daten

Name	Hannes Grunert
Geburtsdatum und -ort	30. 08. 1987 in Ribnitz-Damgarten
Familienstand	ledig
Nationalität	deutsch

Berufliche Laufbahn

seit 09/2020	**Wissenschaftlicher Mitarbeiter**, Universität Rostock, Bereich Informatik, Zentrum für Künstliche Intelligenz in MV *Machbarkeitsstudien zur technischen Umsetzbarkeit von KI-Lösungen; Beraten von Unternehmen zu Anwendungsmöglichkeiten und Grenzen der KI*
12/2015 – 08/2020	**Wissenschaftlicher Mitarbeiter**, Universität Rostock, Institut für Informatik, Lehrstuhl für Datenbank- und Informationssysteme *Lehre im Bachelor- und Masterstudium sowie im Nebenfach (Datenbanken / Data Science); Forschung im Bereich Query Containment und Datenschutz*
06/2013 – 11/2015	**DFG-Promotionsstipendium**, Universität Rostock, Institut für Informatik, Graduiertenkolleg MuSAMA

Ausbildung

2013 – 2022	**Promotion** (Informatik), Universität Rostock
2011 – 2013	**Master of Science** (Informatik), Universität Rostock Titel der Abschlussarbeit: *Integration von Integritätsbedingungen bei der XML-Schemaevolution*
2007 – 2011	**Bachelor of Science** (Informatik), Universität Rostock Titel der Abschlussarbeit: *XML-Schema Evolution: Kategorisierung und Bewertung*
07. 07. 2007	**Abitur**, Richard-Wossidlo-Gymnasium, Ribnitz-Damgarten