Proceedings

21. Workshop Grundlagen von Datenbanken

02.-05. Juni 2009, Rostock-Warnemünde

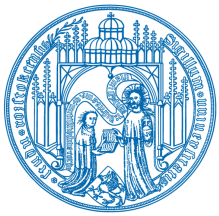
herausgegeben von

Matthias Virgin, André Peters und Dagmar Köhn

aus der Reihe
"Preprints aus dem Institut für Informatik der Universität Rostock"

Sponsoren

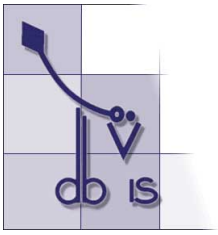
Gesellschaft für Informatik
www.gi-ev.de



Universität Rostock
www.uni-rostock.de

Graduiertenkolleg dIEM oSiRiS
www.diemosiris.de

Graduiertenkolleg MuSAMA
www.musama.de



Lehrstuhl für Datenbank- und
Informationssysteme
dbis.uni-rostock.de

IT Science Center Rügen
www.it-science-center.de

Gesellschaft für Computer- und
Kommunikationssysteme mbH
www.gecko.de

Fraunhofer Institut für Graphische
Datenverarbeitung
www.igd-r.fraunhofer.de

MLP
Finanzdienstleistungen AG
www.mlp-rostock.de

Ausrichter

Universität Rostock
Institut für Informatik
Lehrstuhl Datenbank- und Informationssysteme
D-18051 Rostock

Graduiertenkolleg dIEM oSiRiS
Universität Rostock
http://www.diemosiris.de

Graduiertenkolleg MuSAMA
Universität Rostock
http://www.musama.de

Preprints aus dem Institut für Informatik der Universität Rostock – Proceedings

Seitenzahl: 140
ISSN: 0944-6900
Heftnummer: CS-02-09

Herausgeber

Matthias Virgin
Lehrstuhl für Datenbank- und Informationssysteme
Institut für Informatik
Universität Rostock
E-Mail: mv@informatik.uni-rostock.de

André Peters
Lehrstuhl für Datenbank- und Informationssysteme
Institut für Informatik
Universität Rostock
E-Mail: ap@informatik.uni-rostock.de

Dagmar Köhn
Lehrstuhl für Datenbank- und Informationssysteme
Institut für Informatik
Universität Rostock
E-Mail: dagmar.koehn@uni-rostock.de

Druck

Fraunhofer Institut für Graphische Datenverarbeitung Rostock

Veröffentlicht bei

dblp.uni-trier.de

Rostock, Mai 2009

Inhaltsverzeichnis

1 Eingeladene Vorträge **1**

1 Gunter Saake (Otto-von-Guericke-University Magdeburg): *Tailor-made Data Management - A Challenge for Database Technology?* 1

2 Erhard Rahm (University of Leipzig): *Instance-based Matching of Large Ontologies* 3

3 Johann-Christoph Freytag (Humboldt-Universität zu Berlin): *Privacy Challenges for Database Systems* 5

4 Andreas Heuer (Universität Rostock): *Die unerträgliche Leichtigkeit des Seins — Where Simplicity beats Power* . 7

2 Einreichungen **9**

A Session 1: Performance 9

1 Marina Tropmann, Bernhard Thalheim und Ralf Korff (Christian-Albrechts-Universität zu Kiel, Vattenfall Europe Information Services GmbH): *Performance Forecasting* 9

2 Erik Neitzel und Michael Höding (FH Brandenburg): *Flash-Disks für das schnelle Commit in Oracle* 15

3 Andreas Lübcke (Otto-von-Guericke-University Magdeburg): *Self-tuning of Data Allocation and Storage Management: Advantages and Implications* 21

4 Frank Huber und Johann-Christoph Freytag (Humboldt-Universität zu Berlin): *Query Processing on Multi-Core Architectures* 27

B Session 2: Knowledge & Semantics 33

5 Katrin Zaiß (Heinrich-Heine-Universität Düsseldorf): *Entwicklung eines Evaluations-Frameworks für instanzbasierte Ontologie-Matching-Verfahren* 33

6 Nils Weber, Christoph Eigenstetter, Markus Berg und Antje Düsterhöft (Hochschule Wismar): *Ontologiespeicherung in Datenbanken im Kontext natürlichsprachlicher Dialogsysteme* 39

C Session 3: Data Integration 45

7 Ingolf Geist und Stephan Vornholt (Otto-von-Guericke-Universität Magdeburg): *Eine Link-Datenbank zur Integration von Virtual Engineering-Daten* 45

8 Michael Soffner, Norbert Siegmund, Mario Pukall und Veit Köppen (Otto-von-Guericke-Universität Magdeburg): *Towards Real-Time Data Integration and Analysis for Embedded Devices* 51

D Session 4: Information Retrieval 57

9 David Zellhöfer (Brandenburgische Techn. Universität Cottbus): *Ein Hybridansatz zur Interaktion mit Retrievalsystemen* 57

10 Sebastian Lehrack (Brandenburgische Techn. Universität Cottbus): *A quantum logic-based query processing approach for extending relational query languages* 63

E Session 5: Storage 69

11 Hagen Höpfner und Christian Bunse (International University in Germany): *Towards an Energy Aware DBMS-Energy Consumptions of Sorting and Join Algorithms* . . 69

12 Sebastian Leuot und Wolfgang Benn (Chemnitz University of Technology): *A Self-Adaptive Insert Strategy for Content-Based Multidimensional Database Storage* 75

F Session 6: XML 81

13 Daniel Schall und Henrik Loeser (Technische Universität Kaiserslautern, IBM Deutschland Research & Development): *Anforderungen an die Anwendungsprotokollierung mittels XML* 81

14 Francis Gropengießer (TU Ilmenau): *Kooperatives XML-Authoring: Integration von Workgroups in Workspace-Szenarien* 87

15 Michael Seiferle (Universität Konstanz): *XQuery & OLAP* 93

G Session 7: Data Management 99

16 Yvette Teiken (Carl von Ossietzky Universität Oldenburg): *Modellgetriebene semi-automatische Bereitstellung von Analytischen Informationssystemen* 99

17 Thomas Hasart, Mario Gleichmann, Ilvio Bruder, Peter Forbrig und Andreas Heuer (IT Science Center Rügen): *Anwendungsspezifische Adaption und Erweiterung von OLAP-Anwendungen auf Basis von MDX* 105

18 Jiwu Zhao (Heinrich-Heine-Universität Düsseldorf): *Automatische Parameterbestimmung durch Gravitation in Subspace Clustering* 111

19 Martin Düffer und Andre Peters (Universität Rostock): *Replikation im mobilen Pflegeassistenzsystem MARIKA* . 117

H Session 8: Data Quality 123

20 Fabian Panse (Universität Hamburg): *Datenunvollständigkeit aufgrund der mangelnden Modellierungsmächtigkeit aktuell dominierender Datenmodelle* 123

21 Ludmila Himmelspach (Heinrich-Heine-Universität Düsseldorf): *Vergleich von Strategien zum Clustern von Daten mit fehlenden Werten* 129

22 Heiko Müller (University of Edinburgh): *Archiving and Maintaining Curated Databases* 135

Tailor-made Data Management – A Challenge for Database Technology?

Gunter Saake
School of Computer Science
University of Magdeburg, Germany
e-mail: saake@ovgu.de

The need for tailor-made data management is apparent in database research. There is an impressive body of research on extensible or customizable DBMS like kernel systems or component DBMS. Nevertheless, all these approaches have drawbacks, e.g., regarding customizability or performance, and there is no general solution for efficient development of tailorable DBMS. Novel software engineering techniques (e.g, software product lines and *feature-oriented programming*), can help in developing customizable DBMS and reduce the complexity of building variable solutions as we could show for embedded systems [2]. Applying such an approach to other domains like stream processing, column stores, XML databases, etc. is promising to create DBMS that provide only needed functionality, can be better tuned and maintained, and provide reuse within a domain and even between different domains.

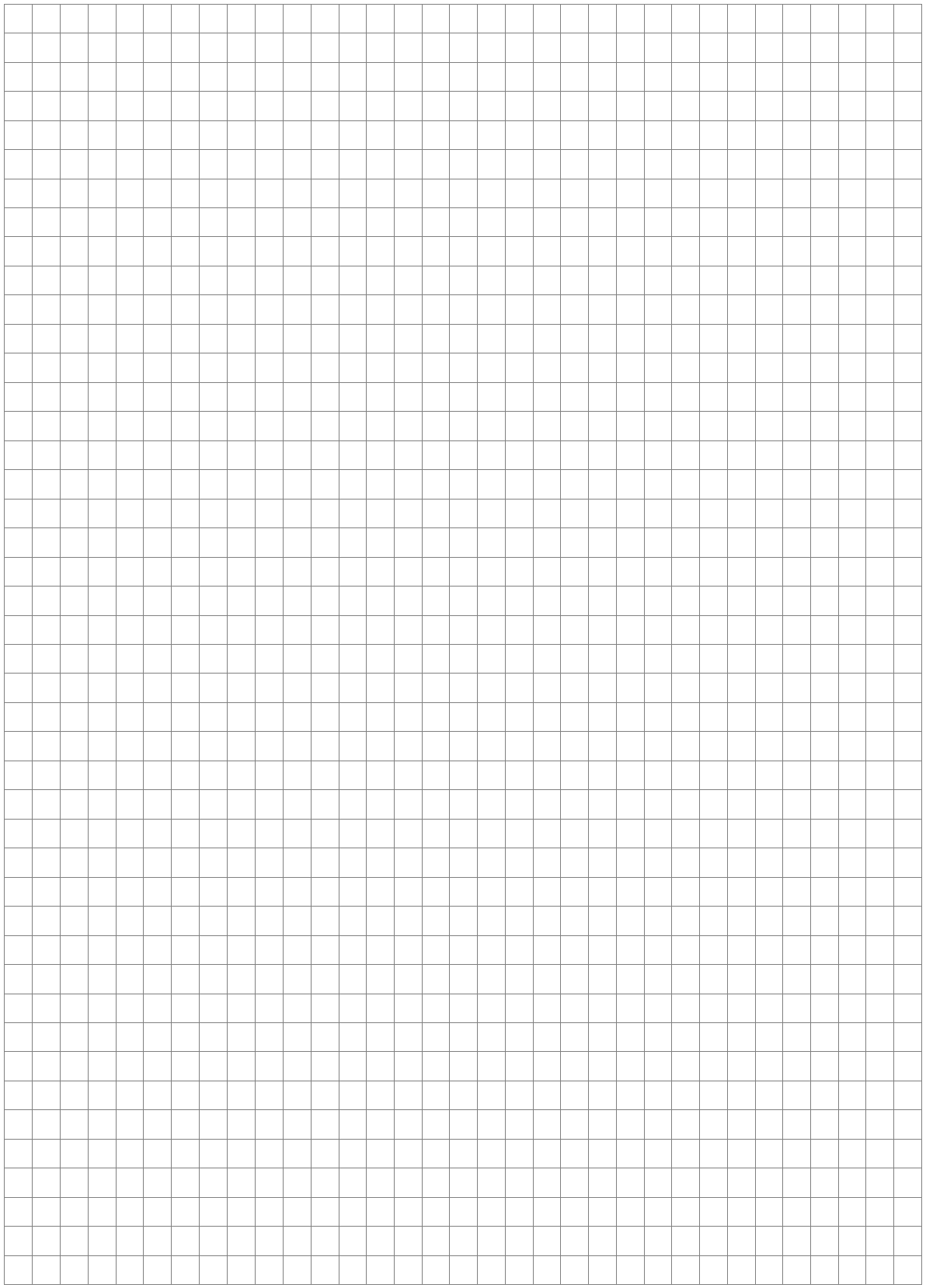
Customizability is also required on the level of the query language, i.e., SQL, which grows with every new standard. Independent of the actual task, which may be as simple as querying a single value, developers are confronted with the full arsenal of SQL. Furthermore, applications only use a small subset of the standard and require special extensions, e.g., for sensor networks or stream processing. This results in *SQL dialects* that are used in different domains and are decoupled from the SQL standard. We could show that SQL (i.e., the grammar and the parser) can be decomposed with a feature-oriented approach. This results in a family of SQL dialects from which tailor-made SQL parsers, e.g., for sensor networks, can be generated [1].

By combining tailor-made SQL dialects and customizable DBMS we think that one can build DBMS that can be fully tailored to an application domain or a special use case within a domain while providing high reuse. Creating such fully customizable systems is a highly challenging task. It involves customizability on all levels of a DBMS such as the query processor and optimizer, the transaction subsystem, or the storage system. Because of interdependencies between those subsystems an architecture has to be developed that is able to handle the resulting complexity. For example, changing the query language affects the whole DBMS including the query optimizer which is highly connected with all other parts of a DBMS. New extensibility mechanisms as provided by feature-oriented programming can help in building such variable systems. However, this is probably not sufficient and the community has to investigate in an architecture and new mechanisms for handling the variability that keep maintainability and performance of DBMS.

Literatur

[1] M. Rosenmüller, C. Kästner, N. Siegmund, S. Sunkle, S. Apel, T. Leich, and G. Saake. Sql à la carte – toward tailor-made data management. In *13. GI-Fachtagung Datenbanksysteme für Business, Technologie und Web (BTW)*, pages 117–136, March 2009.

[2] M. Rosenmüller, N. Siegmund, H. Schirmeier, J. Sincero, S. Apel, T. Leich, O. Spinczyk, and G. Saake. FAME-DBMS: Tailor-made Data Management Solutions for Embedded Systems. In *EDBT'08 Workshop on Software Engineering for Tailor-made Data Management (SETMDM)*, pages 1–6, March 2008.



Instance-based Matching of Large Ontologies

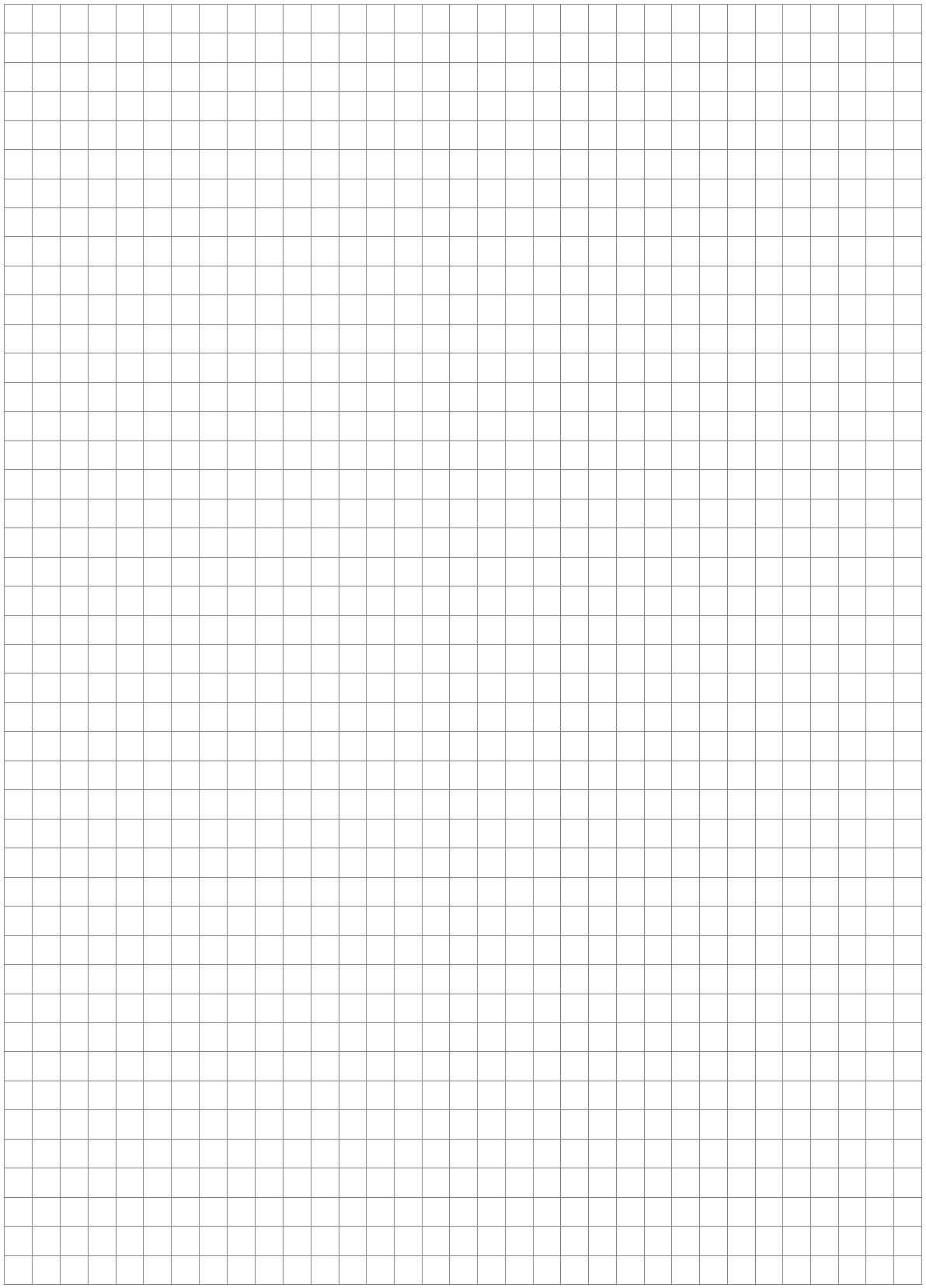
Erhard Rahm
University of Leipzig
http://dbs.uni-leipzig.de

Abstract

Ontologies are increasingly used to semantically categorize data and content. Hence there is an increasing need to match (align) and merge related ontologies, e. g., for data integration or enhanced data analysis. Semi-automatic ontology matching has been an active research area in the last decade and many tools have been developed. The benchmark results of the OAEI (Ontology Alignment Evaluation Initiative) illustrate the current state of the art and indicate significant improvements over the years. Still there are many open challenges in particular for matching large ontologies, such as product catalogs, web directories, or large biomedical ontologies. We illustrate some of the match problems and discuss possible match techniques. In particular, we present different instance-based match approaches and their combination with metadata-based matching. Furthermore, we analyze the influence of ontology evolution on ontology mappings, especially the results of instance-based matching.

References

1. Aumller, D., Do, H., Massmann, S., Rahm, E.: Schema and ontology matching with COMA++. Proc. ACM SIGMOD, 2005.
2. Hartung M, Kirsten T., Rahm E.: Analyzing the evolution of life science ontologies and Mappings. Proc. DILS 2008. LNCS 5109, 11-27. Springer, 2008
3. Kirsten T., Thor A., Rahm E.: Instance-based matching of large life science ontologies. Proc. DILS 2007. LNCS 4544, 172-187. Springer, 2007
4. Massmann, S.; Rahm, E.: Evaluating Instance-based matching of web directories. Proc. 11th Int. Workshop on the Web and Databases (WebDB), 2008
5. Rahm, E., Bernstein, P.: A survey of approaches to automatic schema matching. The VLDB Journal, 10(4): 334-350, 2001.
6. Thor A., Hartung M., Gro A., Kirsten T., Rahm E.: An evolution-based approach for assessing ontology mappings - A case study in the life sciences. Proc. 13th German Database Conf. (BTW), 277-286, 2009
7. Thor, A., Kirsten, T., Rahm, E.: Instance-based matching of hierarchical ontologies. Proc. 12th German Database Conf. (BTW), 2007.



Privacy Challenges for Database Systems

Prof. Johann-Christoph Freytag, Ph.D.
DBIS @ Humboldt-Universität zu Berlin
freytag@dbis.informatik.hu-berlin.de

Abstract

Over the last years the means to collect personal data implicitly or explicitly over the Web and by various kinds of sensors and to combine data for profiling individuals has dramatically increased. These improved abilities have a dramatic impact on the privacy of every individual. This talk focuses on presenting and discussing techniques, concepts, and algorithms how to prevent privacy breaches when personal data is either stored or individuals are at risk to reveal which data they access.

In the first part of the talk we introduce several examples of privacy breaches that help us to better understand what is at risk and to categorize the kind of privacy threads and privacy attacks. These examples will give us a better understanding of the different attacks on privacy as observed on the real world. They also show us that privacy is at risk in different fields such as in the field of communication and in the field of database systems.

The latter will be the focus for the rest of the presentation. To begin with we first discuss some general principles called the principles of Hippocratic database systems that should guide any privacy solution for database systems. These principles rest in part on various requirements coming from the privacy (non computer science) community and from the legal world. They will help us in better understand the different solutions and concepts to protect the privacy of individuals when storing personal data in a database system.

Based on the notion of access privacy we describe solutions to protect the users privacy when access a database such that (s)he does not reveal which data is accessed in the database. We present different solutions with different levels of privacy always assuming the same model of privacy leakage.

We then turn to the notion of data privacy. The challenge of data privacy is to prevent the linkage of stored data to individuals. For example, if we store patient and disease data, it should be impossible to find out which individual has which disease when the data is released. As a first step we introduce the notion of k-anonymity as a way to release data without revealing the identity of the individual they belong to. We show that the basic notion of k-anonymity does not suffice; it must be improved in various ways due to the different kind of privacy leakages (or possible attacks) that might occur. Throughout the talk we also discuss the ques-

tion of how to measure privacy and the leakage of privacy. Whenever appropriate solutions are known we include those in our talk.

During the last part of the talk we extend the notion of privacy to distributed systems, in our case to mobile cooperative distributed systems as found in Intelligent Transportation Systems (ITS), by presenting some initial results of the EU project PRECIOSA which we participate in.

Some of this research work was performed as part of the EU project PRECIOSA. I would like to thank Martin Kost and Lukas Dölle, Ph.D. students of the DBIS research group at Humboldt-Universitt zu Berlin who have been essential in putting together this presentation.

Die unerträgliche Leichtigkeit des Seins — Where Simplicity beats Power

Prof. Dr. Andreas Heuer
Institut für Informatik, Universität Rostock
heuer@informatik.uni-rostock.de

Das Sein.

Seit fast fünf Jahrzehnten werden in Dateiverwaltungssystemen oder Datenbanken große Mengen von Informationen verwaltet. Effiziente Zugriffsstrukturen waren seit jeher ein Thema, mit der Zeit kamen Datenbankentwurfsprinzipien, Transaktionskonzepte und Anfrageoptimierung (neben vielen anderen) als Schwerpunkte der Forschung und Entwicklung dazu.

Die Leichtigkeit des Seins.

Ein auf den ersten Blick simples Datenbankmodell, das relationale Datenbankmodell, setzte sich seit 1970 in der Forschung und seit 1980 auch im täglichen Einsatz durch. Datenbankentwurfsprinizipen konnten formalisiert werden, Transaktionskonzepte und Anfrageoptimierung für verschiedene Anforderungen weiterentwickelt werden. Das Relationenmodell wurde für ein Jahrzehnt zum Erfolgsmodell.

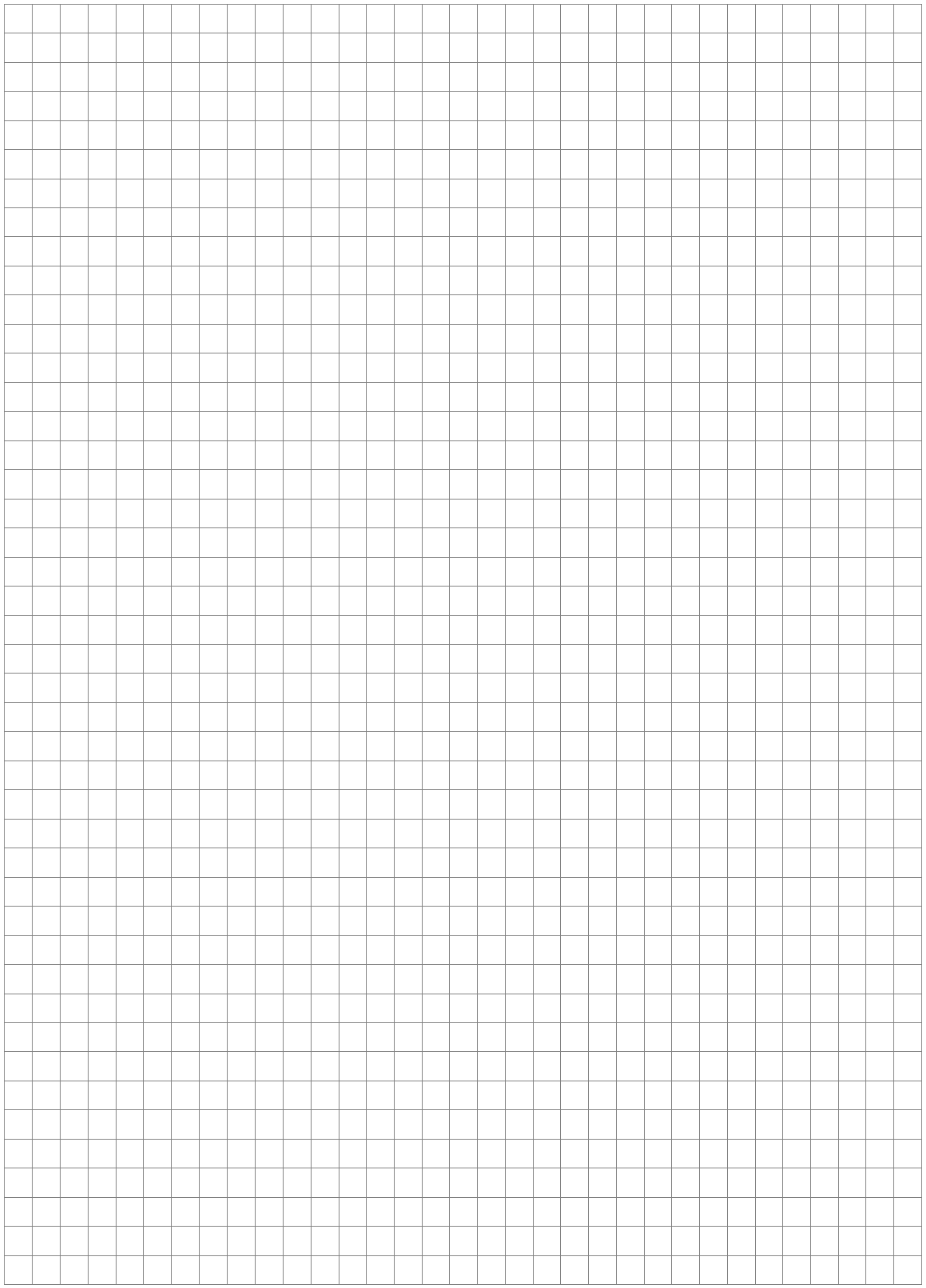
Die unerträgliche Leichtigkeit des Seins.

Für viele Anwendungen war das Relationenmodell allerdings zu simpel. Seit etwa 1990 setzten sich in der Forschung objektorientierte, später als Kompromiss objektrelationale Datenbanksysteme durch. Mit Verbreitung des Webs wurden Markup-Sprachen zur Beschreibung von Dokumenten beliebt, die in Form von XML dann auch der nächsten Generation von Datenbanksystemen ein Pseudo-Datenbankmodell lieferten. Mit Hochdruck arbeiteten die Forscher dann daran, effiziente Zugriffsstrukturen, Transaktionskonzepte und Anfrageoptimierungstechniken sowie Datenbankentwurfsprinzipen auf diese neuen Datenbankmodelle hin anzupassen – oder nach Möglichkeit unter Beibehaltung der alten Stärken noch zu verbessern.

Where Simplicity beats Power

Die Forschung ist in dieser Hinsicht jedoch häufig gescheitert. Mit den Anfrageoperatoren für die komplexeren Datenbankmodelle gingen entscheidende Optimierungseigenschaften verloren. Die Transaktionskonzepte wurden komplex und die Konsistenz des Datenbankzustands in hohem Maße gefährdet. Effiziente Zugriffstrukturen für die neuen komplexen Datentypen wurden kaum umgesetzt und oft der Umweg über alte relationale Strukturen bevorzugt.

Im Vortrag sollen die Gründe für diese Schwierigkeiten aufgezeigt werden. Am Beispiel von Datenbankentwurfsprinzipien, Zugriffsstrukturen, Anfrageoptimierung und Transaktionskonzepten soll aufgezeigt werden, wo die Einfachheit des relationalen Datenbankmodells jede Mächtigkeit der neueren Datenbankmodelle schlägt. Als Fazit kann gelten: *Für neue Anwendungen und neue Herausforderungen (etwa Datenbanklösungen für Assistenzsysteme) kann das relationale Datenbankmodell wieder die Grundlage sein. Ein Ausweichen auf modernere, aber auch komplexere Datenbankmodelle ist nicht immer die Ideallösung.*



Performance Forecasting

Marina Tropmann (mtr@is.informatik.uni-kiel.de)[1]
Bernhard Thalheim (thalheim@is.informatik.uni-kiel.de)[1]
Ralf Korff (ralf.korff@vattenfall.de)[2]

[1]Technologie der Informationssysteme, Institut für Informatik
Christian-Albrechts-Universität zu Kiel, Olshausenstr. 40, 24098 Kiel
[2]Vattenfall Europe Information Services GmbH
Kopenhagener Str. 83-89, 13158 Berlin

Zusammenfassung

Schon seit mehreren Jahrzehnten wird bei meisten Unternehmen Performance Tuning von Rechnersystemen betrieben. Dieses Thema wurde wie von der Seite der Unternehmen als auch von der Wissenschaft mittlerweile in zahlreichen unterschiedlichen Aspekten erforscht und ausgearbeitet. Es existieren inzwischen viele wissenschaftliche Artikel und Bücher zu dem Thema, wie z.B. von Dennis Shasha und Philippe Bonnet „Database Tuning: Principles, Experiments, and Troubleshooting Techniques“ [8] oder von Sitansu S. Mittra „Database Performance Tuning and Optimization“ [4], nur um einige zu nennen.
Das Thema Performance Forecasting hat dagegen erst in letzten Jahren an Bedeutung gewonnen, zusammen mit dem Rechnerkonsolidierungsansatz. Es ist in diesem Zusammenhang ungeheuer wichtig geworden, die Performanceprobleme nicht erst nach ihrem Auftreten zu beseitigen, sondern vorher zu „wissen“, welche Probleme, wann, wo, in welchem Ausmaß und mit welchen Folgen und Auswirkungen auftreten können, um die Ursachen noch vor einer Systemstörung zu eliminieren.
Die meisten existierenden Arbeiten in diesem Bereich (z.B. [7] und [1]) betrachten vor allem die physische Ebene und weniger die logische. Für die meisten Probleme liegen aber die Ursachen eher in der konzeptioneller bzw. logischer Ebene und eine Lösung auf dem physischen Niveau beseitigt oft das Problem nicht, sondern führt eher zu einem kurzfristigen Erfolg.
Unser Projekt „Intelligent Performance Analyser“ beschäftigt sich in erster Linie mit Performance Forecasting und proaktivem Tuning. Das Ziel des Projektes ist, eine möglichst vollständig automatische, lernfähige und benutzerfreundliche Anwendung zu entwickeln, die in einem breiten Spektrum von Unternehmen eingesetzt werden kann.
Wir beschäftigen uns mit der physischen, wie auch mit der konzeptionellen Ebene und treffen Forecasting - Aussagen in Bezug auf explizite Vorhersage - Modelle.

1 Einleitung

Schon immer wollten Menschen die Zukunft sehen und planen können. Während es ihnen im eigenen Lebensverlauf eher selten gelingt, gibt es inzwischen mehrere Bereiche, wie Klimaforschung, Geologie, Medizin, Wirtschaft und einige andere, die mittlerweile zuverlässige und detaillierte Prognosen liefern, die direkt oder indirekt alle Menschen betreffen. Zuverlässige und exakte Prognosen haben auch in der Informatik, speziell im Bereich der System - Performance vor allem in letzten Jahren an Bedeutung gewonnen.
Viele Unternehmen betreiben seit einiger Zeit starke Rechnerkonsolidierung. Durch die Konsolidierung der Rechner in einem großen Umfang wollen die Unternehmen einerseits einen umweltschonenden Betrieb ihrer Rechenzentren und andererseits beachtliche Kosteneinsparungen erreichen. So werden unterschiedlichste Anwendungen und Datenbanken öfters ohne eine Systematik zusammengelegt. Mit dem Konsolidierungsansatz haben die Unternehmen dann eine

unregelmäßig verteilte, zusammengelegte Landschaft, auf der im Fall einer Systemstörung die Ursachenfindung einen enormen Aufwand und hohe Kosten bedeutet. Mit der wachsenden Menge an Information, die verarbeitet werden muss, und mit der Erhöhung der Komplexität der Systeme und der Anforderungen ist dieser Prozess keinesfalls einfacher, sondern um vieles schwieriger geworden. Da meistens dazu noch Verpflichtungen gegenüber Dritten (z.B. Kunden, die Nutzer des Rechenzentrums sind) vorhanden sind, wird die Verfügbarkeit bzw. die Ausfallsicherheit der Systeme zum wichtigsten Punkt.
Deshalb ist es wichtig geworden, die Performanceprobleme nicht erst nach ihrem Auftreten zu lösen, sondern die Ursachen noch vorher zu erkennen und zu beseitigen. Dazu sind zuverlässige und möglichst exakte Performance - Prognosen notwendig.
In dieser Arbeit möchten wir Ideen und Methoden von Performance - Forecasting vorstellen, die wir in einem gemeinsamen Projekt mit einem großen Industrieunternehmen entwickeln und einsetzen.

2 Forschungsstand und themenbezogene Arbeiten

Die meisten Arbeiten im Bereich der System - Performance [8, 4, 2, 3], sei es Rechner - Performance (Hardware, Betriebssystem) oder Datenbank - Performance, beschäftigen sich hauptsächlich mit reaktivem Performance Tuning und bieten Lösungen zu Problemen erst nach ihrem Eintreten. Unser Ansatz ist, mögliche zukünftige Probleme noch vor ihrem Eintreten zu erkennen und mit proaktivem Tuning zu verhindern.
Die wenigen existierenden Arbeiten in Bereich Performance Forecasting sind entweder oberflächlich und betreffen nur vereinfachte Szenarios mit dem Schwerpunkt auf der physischen Ebene [7, 1] oder beschäftigen sich mit einzelnen Systemkomponenten oder einzelnen Prognose - Methoden [9, 5, 10]. Wir versuchen sowohl komponentenweise als auch systemübergreifend Prognosen zu erstellen, wobei wir uns nicht nur auf die physische Ebene konzentrieren, sondern viel mehr auf die logische bzw. konzeptionelle Ebene ausrichten.

3 Methoden und Konzepte

Das primäre Ziel bei Performance Forecasting ist, mögliche zukünftige Probleme rechtzeitig zu erkennen, Risikofaktoren zu identifizieren und die Ursachen mit proaktivem Tuning zu beseitigen. Um dies zu bewerkstelligen, ist es wichtig, das System permanent zu überwachen und ausreichend und gezielt Statistiken zu sammeln und auszuwerten.
Ein System kann vereinfacht als eine „black box“- Struktur mit Eingängen, Ausgängen und einer Menge von Einwirkfaktoren dargestellt werden. Wir verstehen unter dem Systembegriff einen Gesamtkomplex, zusammengesetzt aus einzelnen Bestandteilen, laufenden und geplanten Prozessen und Steuerungsfunktionen. Jedes System besteht aus miteinander zusammenhängenden und aufeinander wirkenden Komponenten, besitzt Schnittstellen zu seiner Umgebung und stellt in seiner Gesamtheit eine abgeschlossene Funktionseinheit dar. Da die Komponenten eines Systems somit eine komplexe Struktur bilden, in die umfangreiche Beziehungen zwischen den Elementen und demzufolge viele Abhängigkeiten integriert sind, beein-

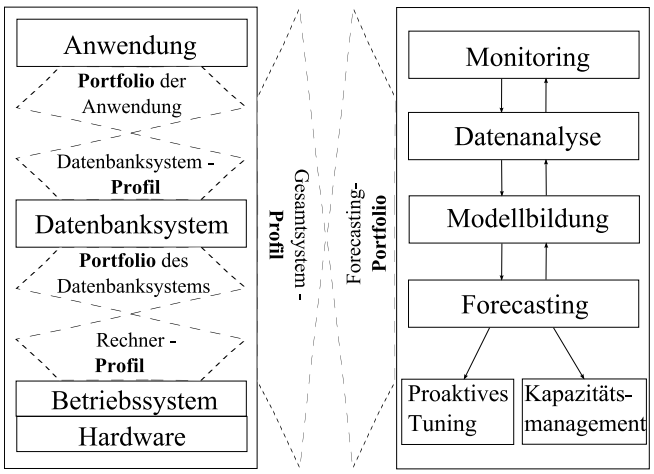


Abbildung 1: Systemanalyse - Modell

flussen die Veränderungen des Verhaltens von einigen Komponenten und Prozessen die anderen. Somit, um eine Systematik und Konsistenz in die Systembeschreibung einzubringen und die Wechselwirkungen zwischen den Komponenten beschreiben zu können, unterteilen wir das Gesamtsystem in einzelne (vorhandene) Subsysteme, wie sie in der Abbildung 1 (links) dargestellt sind. Auf dieser Abbildung sind vier Subsysteme zu sehen: Hardware, Betriebssystem, Datenbanksystem und Anwendung. Die Wechselwirkungen und Beeinflussungen der Subsysteme aufeinander werden hier durch das *Profil - Portfolio - Konzept* ausgedrückt.
Profil bedeutet in diesem Fall die Leistungsfähigkeit oder das Potential des Systems bzw. Subsystems und umfasst Austattung, Merkmale, Performance - Charakteristiken etc. *Portfolio* beschreibt dagegen die Belastung, die durch das entsprechende Subsystem auf das darunterliegende erzeugt wird, und umfasst Aufgaben, Rechte, Beschränkungen etc.
Somit besitzt jedes Subsystem ein entsprechendes *Profil* und *Portfolio*. Wir haben die beiden Komponenten Hardware und Betriebssystem zu einem Subsystem zusammengefasst, da es bei unserem Ansatz nur in seltensten Fällen sinnvoll ist, sie zu trennen. Denn, wenn wir von großen Systemen sprechen, für die dieser Ansatz in erster Linie entwickelt wurde, liegt meistens die Situation vor, dass die Hardware mit einem auf sie abgestimmten Betriebssystem ausgeliefert wird, so dass ein Betriebssystem-Wechsel von Anfang an nicht vorgesehen ist.
Für alle Subsysteme ist die Abstimmung von Profil und Portfolio essenziell. Sobald das Profil von einem Subsystem und das Portfolio von einem anderen erstellt wurden, kann überprüft werden, ob die vorliegende Austattung und die vorgenommenen Konfigurationen für die zu leistende Arbeit und Belastung optimal bzw. akzeptabel sind. Anhand der Profil - Portfolio - Untersuchung können ebenfalls Analysen zur Identifizierung von Risikofaktoren in der zukünftigen Entwicklung des Systems durchgeführt und Forecasting - Aussagen gemacht werden. Außerdem bietet dieses Konzept die Möglichkeit, für Kapazitätsmanagement eine nahezu optimale Nutzung von Hardware - Ressourcen zu berechnen und hilft bei der Rechnerkonsolidierung die richtige Entscheidung zu treffen.
Das *Profil - Portfolio - Konzept* beschreibt auch den Zusammenhang zwischen den beiden Teilen (links und rechts) in der Abbildung 1. Auf der einen Seite befindet sich das Profil für das Gesamtsystem, auf der anderen Seite das Forecasting - Portfolio mit den Aufgaben, die bezüglich des Systems verrichtet werden. Im rechten Teil der Abbildung 1 sind die einzelnen Aufgaben zur Systemanalyse aufgelistet. Der gesamte Vorgang wird in folgende Schritte unterteilt:

- **Monitoring:** Das System - Monitoring dient einer zentralen Überwachung der System - Performance. Durch das Monitoring kann der Ursprung der auftretenden Probleme schnell und relativ einfach identifiziert werden. Eine Voraussetzung für zuverlässiges Monitoring ist Sammlung und Aufbereitung von Statistiken für das gesamte System und für einzelne Subsysteme. Es ist wichtig, alle Systemkomponenten zu berücksichtigen, da die Ressourcen der Komponenten durch Abhängigkeiten und Wechselwirkungen eng miteinander verbunden sind. Die Statistikdaten werden zuerst als Rohdaten in ihrem ursprünglichen Format gesammelt, ohne Unterteilung in mehr oder weniger wichtige. Als Nächstes ist es notwendig, die Qualität der Daten zu kontrollieren, um Ausreißer, fehlerhafte Daten oder Duplikate zu markieren bzw. zu entfernen. Erst danach werden die Statistikdaten aufbereitet. Sie werden in ein einheitliches Format gebracht und zentral in einem Repository für weitere Verarbeitung zur Verfügung gestellt.

- **Datenanalyse:** Als Nächstes wird der charakteristische Workload des Systems bzw. der Subsysteme bestimmt. Die aufbereiteten Daten werden analysiert und bezüglich bestimmter Anforderungen überprüft, um alle Veränderungen im System und eventuelle Verletzungen vorgegebener Bedingungen zu erfassen. Eine weitere Aufgabe der Datenanalyse ist die zielgerichtete Daten - Vorbereitung für unterschiedliche Methoden der Modellbildung und Prognostizierung.

- **Modellbildung:** Die Forecasting - Modelle dienen im Allgemeinen einer Veranschauli-

chung bzw. Vereinfachung der Realität, um eine bessere Handhabung und Analyse eines vorliegenden komplexen Systems zu ermöglichen. Da jede Form der Modellierung ihre Vorteile und Nachteile besitzt, ist es nicht ausreichend, nur eine Modellierungsmethode zu betrachten. Um ein möglichst vollständiges Modell für ein vorliegendes System zu erhalten, müssen die einzelnen Strukturen und die Funktionalität des Systems beschrieben werden. Die Strukturmodellierung (oder statische Modellierung) umfasst die Basisvariablen der Prozesse und Komponenten in bestimmten Zuständen, wobei die zeitlichen Veränderungen der Parameter nicht berücksichtigt werden. Die Modellierung der Funktionalität (oder dynamische Modellierung) betrifft dagegen die Zustandsübergänge, um die dynamischen Charakteristiken und zeitliche Veränderungen der Variablen zu erfassen. Die beschriebenen Charakteristiken können theoretisch, experimentell oder mittels einer Kombination untersucht werden.
Bei diesem Ansatz betrachten wir zwei Arten der Modellierung: Benchmark-Modelle aus dem Gebiet der *physischen Modellierung* und einige ausgewählte Methoden der *mathematischen Modellierung.*

  - *Physische Modellierung.* Im Allgemeinen werden bei der physischen Modellierung Eigenschaften eines Systems aus den Experimenten ermittelt. Benchmarks kann man beschreiben als Systematisierung von Vorgehensweisen und Lösungen für ausgewählte Problembereiche mit Hilfe von Simulationen und Messungen auf einem Testsystem, das dem Produktionssystem möglichst ähnlich ist. Dabei ist es keinesfalls notwendig, dass die Systeme identisch sind. Es ist ausreichend, Simulationen auf einem kleineren Testsystem durchzuführen und Ergebnisse auf das große Produktionssystem zu skalieren. Wobei die physischen Modelle viel aufwendiger im Vergleich zu mathematischen Modellen sind, sind sie genauer und zuverlässiger für Prognosen, da sie die Komplexität der realen Systeme am besten widerspiegeln.
  - *Mathematische Modellierung.* Ein mathematisches Modell ist eine vereinfachte strukturierte Beschreibung eines realen Systems. Es ist wichtig, für die vorliegende Problematik ein geeignetes Modell zu bestimmen, das die Eigenschaften des Systems möglichst vollständig und korrekt widerspiegelt. Für mathematische Modellierung sind am meisten die Methoden der Regression, Queuing-Theorie, Neuronale Netze und verallgemeinerte lineare Modelle geeignet.

Eine wichtige Voraussetzung für die Modellierung ist die Beständigkeit des Systems. Falls das vorliegende System verändert wird (durch ein Update, eine neue Softwareversion oder eine weitere Hardwarekomponente), kann nicht mehr die Korrektheit der aufgestellten Modelle garantiert werden.

- **Forecasting:** Alle Prognosen sind immer mit bestimmten Fragestellungen verbunden. Im Bereich der System - Performance sind die häufigsten Fragestellungen die folgenden:

  - Wie sehen die Performance - Charakteristiken für das System oder ausgewählte System - Komponenten in der Zukunft aus?
  - Wie verändern sich einzelne Charakteristiken bei einer Vergrößerung/Verkleinerung von einem (oder auch mehreren) Parameter?

  In Abhängigkeit von der Fragestellung wird ein geeignetes Modell ausgewählt und anhand des Modells werden Aussagen getroffen und Empfehlungen ausgesprochen.
  Bezüglich des Kapazitätsmanagements müssen die betrachteten Systeme noch auf *Verträglichkeit* untersucht werden. Der Verträglichkeitstest besteht aus zwei ineinander übergreifenden Teilen: physische Verträglichkeit und logische Verträglichkeit.

  - Die *physische Verträglichkeit* bedeutet die Kompatibilität der Systemleistungen und umfasst z.B. Hardware- und Netzwerkressourcen sowie die Datenbankeinstellungen.
  - Die *logische Verträglichkeit* bedeutet die Kompatibilität der Spezifikationen und umfasst z.B die Lastprofile der Anwendungen sowie die Datenbankspezifikationen.

  Neben der Verträglichkeit der Systeme spielt die *Belastbarkeit* für die dynamische Zuordnung der Systemressourcen eine große Rolle. Ein wichtiger Parameter dabei ist die *maximale Belastbarkeit*. Ähnlich zu Verträglichkeit lässt sich die Belastbarkeit in physische und logische aufteilen. Während die physische Belastbarkeit gemessen werden kann, ist die logische Belastbarkeit abhängig von der Anwendung und bestimmten Spezifikationen.

- **Nutzung:** nachdem die Prognosen erstellt und mögliche Risikofaktoren identifiziert wurden, können anhand von Aussagen und Empfehlungen mit proaktivem Tuning die Ursachen möglicher Probleme eliminiert werden. Die Methoden dabei sind die gleichen, wie bei dem reaktiven Tuning. Abhängig von den vorliegenden Systemen existieren in entsprechenden Bereichen mittlerweile viele Arbeiten zu diesem Thema.

4 Ausblick

In dieser Arbeit wurden Methoden und Konzepte für Performance Forecasting vorgestellt. Während mittlerweile einzelne Schritte teilweise automatisiert wurden, fehlt noch die vollständige Automatisierung für den Gesamtvorgang. Nicht zu unterschätzen ist auch die Tatsache, dass beim Forecasting immer ein Kompromiss geschlossen werden muss zwischen der Genauigkeit der Prognosen und der zusätzlichen Belastung des Systems durch die Sammlung von Statistiken. Die Systeme sind oft so ausgelastet, dass jede zusätzliche Belastung für sie kritisch sein kann. Außerdem können dadurch auch die Statistiken selbst verfälscht werden. Ein wichtiger Punkt dabei ist eine gezielte (problemorientierte) Sammlung von Statistiken. Dadurch können Ergebnisse schneller berechnet und Ressourcen gespart werden.

Literatur

[1] Lutz Fröhlich. *Oracle 11g Performance Forecast.* Hanser, 2008.

[2] Raj Jain. *The Art of Computer Systems Performance Analysis.* John Wiley & Sons, 1991.

[3] Horst Langendörfer. *Leistungsanalyse von Rechensystemen.* Carl Hanser Verlag, 1992.

[4] Sitansu S. Mittra. *Database Performance Tuning and Optimization.* Spriger, 2003.

[5] Edwin Pednault. Transform regression and the kolmogorov superposition theorem. *Proceedings of the Sixth SIAM International Conference on Data Mining*, pages 35–46, 2006.

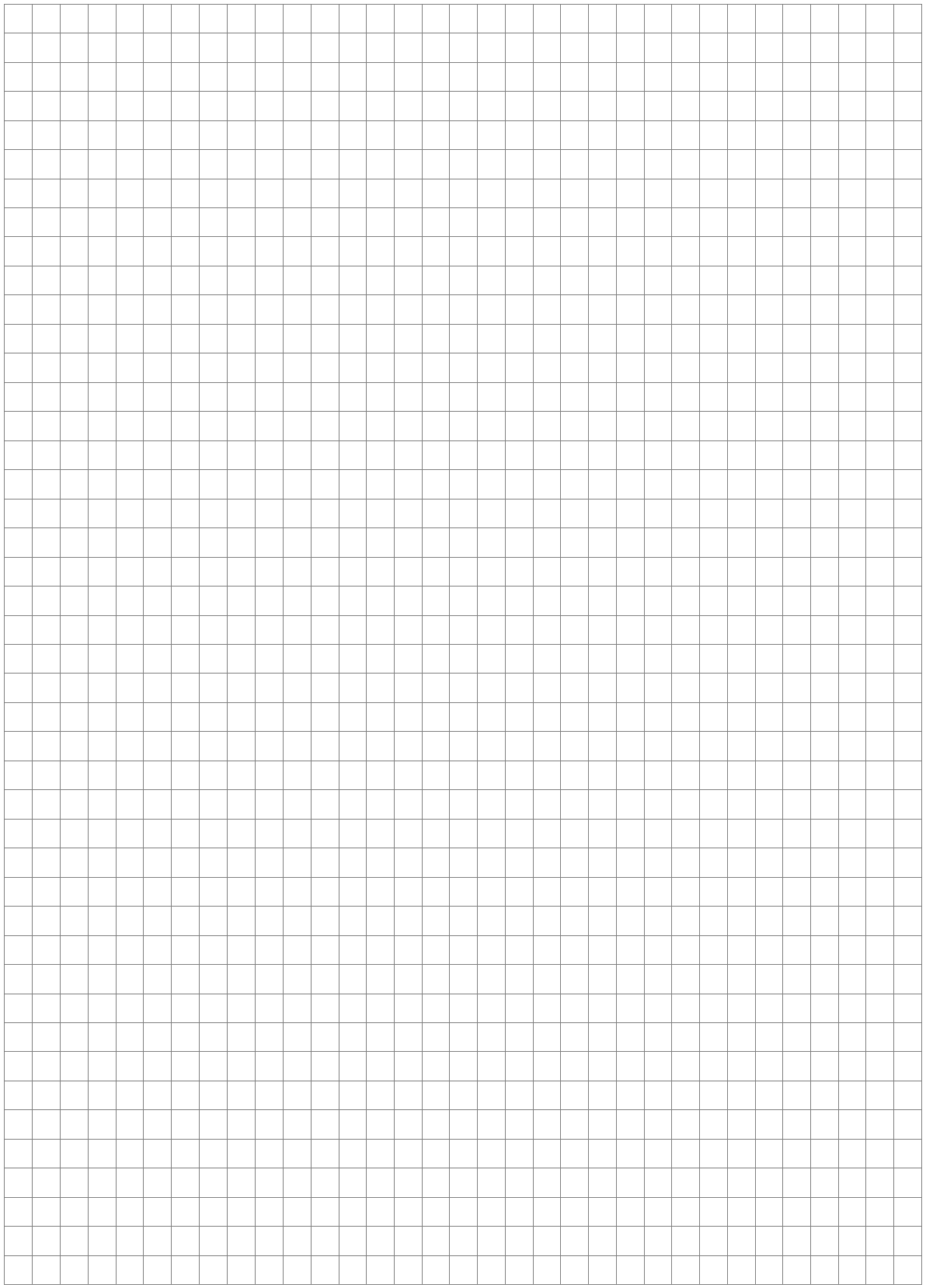
[6] Erhard Rahm and Theo Härder. *Datenbanksysteme. Konzepte und Techniken der Implementierung.* Springer, 2001.

[7] Craig Shallahamer. *Forecasting Oracle Performance.* Apress, 2007.

[8] Dennis Shasha and Philippe Bonnet. *Database Tuning: Principles, Experiments, and Troubleshooting Techniques.* Morgan Kaufmann, 2002.

[9] Christopher Stewart, Terence Kelly, and Alex Zhang. Exploiting nonstationarity for performance prediction. *SIGOPS Oper. Syst. Rev.*, 41(3):31–44, 2007.

[10] Rich Wolski. Dynamically forecasting network performance using the network weather service. *UCSD Technical Report TR-CS96-494*, 1998.



Flash-Disks für das schnelle Commit in Oracle

Erik Neitzel, Michael Höding
FH Brandenburg, Magdeburger Straße 50, 14770 Brandenburg/Havel
neitzel@fh-brandenburg.de, hoeding@fh-brandenburg.de

Zusammenfassung

In diesem Beitrag werden Einsatzmöglichkeiten von Flash-Disks (auch Solid State Disks bzw. SSDs genannt) in DBMS-basierten Systemen betrachtet. Anhand des DBMS Oracle diskutieren wir eine Idee durch Flash-Disk-Einsatz das Commit in DBMS zu beschleunigen. Für detaillierte Messungen wird derzeit ein Testframe entwickelt. Erste Versuche und Messungen demonstrieren das Verhalten von SSDs zur Speicherung von Transaktionslogs.

1 Motivation

Die Verbesserung der Performance ist ein Hauptziel der Forschung im Bereich Datenbanken. Motivation ist letztendlich ein für jeden Benutzer akzeptables Antwortzeitverhalten im Produktivbetrieb. Dabei werden Ergebnisse aus nahezu allen Bereichen der Informatik genutzt oder Forschung in diesen Bereichen motiviert. Die technische Informatik liefert schnelle und optimierte Hardware, die theoretische Informatik Algorithmen. Sogar der KI-Bereich bietet interessante Prognoseverfahren zur Optimierung.

Das zentrale Problem ist dabei die Speicherlücke zwischen Hauptspeicherzugriff und Sekundarspeicherzugriff [6]. Physischer Zugriff soll einerseits so weit wie möglich vermieden werden. Andererseits muss der physische Zugriff beschleunigt werden.

Eine aktuelle technische Entwicklung sind Solid State Disks oder Flask-Disks, die als nichtmechanischer Speicher der traditionellen Festplatte Konkurrenz machen. Neben Aspekten wie mechanischer Robustheit und Energieeffizienz versprechen SSDs technologiebedingt einen schnelleren wahlfreien Lesezugriff. Ihr unmittelbarer Einsatz für Datenbanksysteme bringt allerdings nicht in jedem Fall eine Performancesteigerung wie in [4] und [5] gezeigt wurde. Ziel ist es nun, die Einsatzmöglichkeiten von SSDs in Datenbanksystemen näher zu untersuchen, um so Möglichkeiten und Grenzen zu erkennen.

2 Technische Aspekte, Betrachtungsebenen

SSDs bestehen aus Flash- und Controllerchips, angeordnet auf einer Leiterplatte. Die Anzahl der Chips hat hierbei analog zur RAID-Technologie großen Einfluss auf die Geschwindigkeit. Bei reinen Lesevorgängen und Multitasking-Operationen (gleichzeitigem Lesen und Schreiben) sind Flash-Disks den herkömmlichen Festplatten überlegen [3]. Beim alleinigen Schreiben waren bzw. sind SSD hingegen technologiebedingt etwas langsamer. Eine SSD ist in Blöcke unterteilt, wobei unter Umständen der gesamte Block neu geschrieben muss. Wenn ein Bit seinen Wert von 1 auf 0 ändert, müssen alle Bits des Blocks durch eine negative Spannung zurückgesetzt werden. Ein weiteres Problem ist die begrenzte Zahl der Schreibzyklen für Flash-Speicherzellen.

Wir möchten in diesem Beitrag Einsatzmöglichkeiten von SSDs in Datenbanksystemen betrachten. Wichtig sind dabei die folgenden Ebenen:

- **Anwendung**
  In [5] wurde der DBT2-Benchmark genutzt. Als typischer Anwendungsbenchmark emuliert DBT2 eine OLTP-Last und misst die Anzahl der durchgeführten Transaktionen in einem

Zeitraum. Nachteilig waren die problematische Portierbarkeit für verschiedene DBMS und die fehlenden Eingriffsmöglichkeiten für detaillierte Untersuchungen. Aus diesen Gründen wird derzeit ein Testframe entwickelt, der diese Probleme zu überwinden hilft.

- **Datenbanksystem**
  Auf der Ebene des DBMS kommen Transaktionslogs zum Einsatz, die durch schnelles sequenzielles Schreiben ein schnelles Commit ermöglichen. Eine Optimierungsmöglichkeit mit SSDs wird im nächsten Abschnitt diskutiert. Hinsichtlich des wahlfreien Schreibens von Datenbankblöcken müssen die genutzten Verfahren für den SSD-Einsatz hinsichtlich Fragmentierung und Clusterung untersucht werden.

- **Betriebssystem, Treiber, Hardware**
  Die Probleme der begrenzten Schreibzyklenzahl und der geringen Schreibgeschwindigkeit durch Flashen sind gegenwärtig die Entwicklungsschwerpunkte bei den Herstellern. So werden Schreibzugriffe auf Controller- oder Treiberebene mit dem Ziel verteilt, möglichst alle Speicherzellen der SSD gleich häufig zu beschreiben. Eine physische Fragmentierung sollte dabei kein Problem sein, da es keine Wartezeiten zur Positionierung von Lese-Schreibköpfen gibt. Allerdings ist dieser Aspekt beim schreibenden DBMS-Zugriff zu beachten. Diese Entwicklungen müssen berücksichtigt werden, da sie mit Optimierungsansätzen im DBMS wechselwirken können.

Die Entwicklung von SSDs befindet sich derzeit in einer sehr aktiven Phase. Inzwischen sind SSDs zu traditionellen Magnetplatten konkurrenzfähig, wie erste Experimente zeigen.

3 Nutzung in Oracle

Am Beispiel von Oracle möchten wir nun die gezielte Nutzung von SSDs für das Transaktionslog diskutieren. Abbildung 1 abstrahiert den schreibenden Zugriff bei Oracle. Beim Commit werden im Schritt 5b die Einträge des Redolog-Puffers in die Logdateien geschrieben. Ist diese Schreiboperation beendet, so bekommt der Anwender den Erfolg gemeldet. Die Geschwindigkeit der Schreiboperation bestimmt also entscheidend das Antwortzeitverhalten.

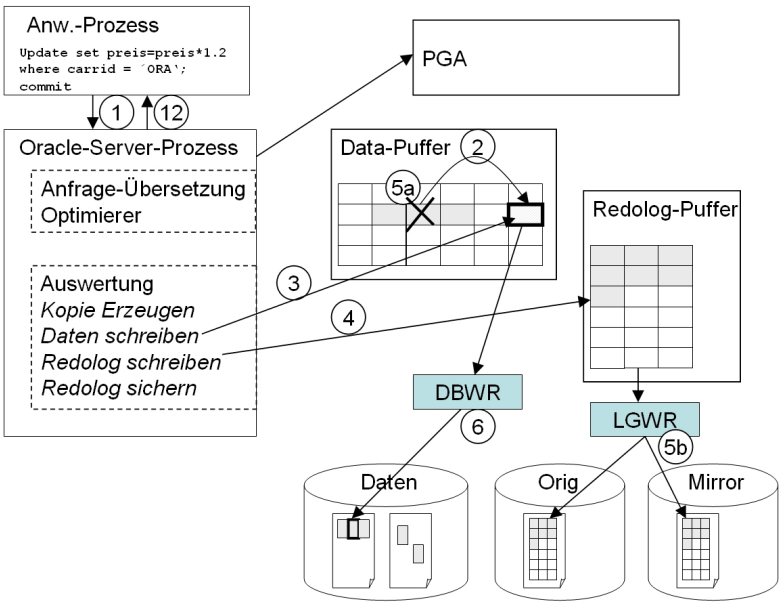


Abbildung 1: Transaktionlog am Beispiel von Oracle [1]

Dieser Mechanismus, der in ähnlicher Form in allen DBMS implementiert ist, nutzt die Tatsache, dass das sequentielle Schreiben in eine Datei (hier die Log-Datei) einen großen Geschwindigkeitsvorteil gegenüber dem verteilten Schreiben, z.B. in die Daten-Dateien bietet. In Oracle sind die Log-Dateien als Ring-Puffer organisiert. Ist eine Log-Datei voll oder wird ein expliziter Log-Switch ausgelöst, so kommt die nächste Log-Datei zum Einsatz (vgl. Abb. 2, Schritt 1). Inzwischen kann die geschlossene Log-Datei archiviert werden (Schritt 2). Ist die Log-Datei nun bei der nächsten Nutzung noch mit alten (ungültigen) Inhalten beschrieben, so stellt dies für den SSD-Einsatz ein Problem dar. Auf Hardware- oder Treiberebenen wird nicht erkannt, dass die Altdaten keine Bedeutung haben. Entsprechend ist mit jedem Schreiben eines Logeintrags ein teures Flashen und Zurückschreiben des Speicherblocks verbunden.

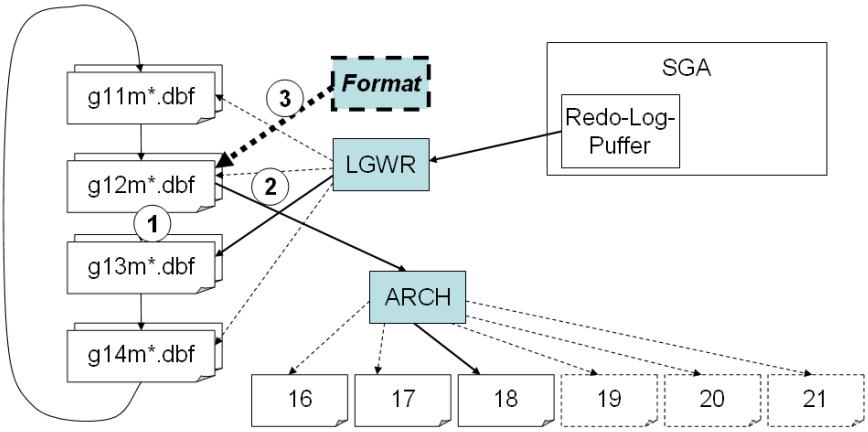


Abbildung 2: Log-Switch, Archivierung und Formatierung

Zur Lösung dieses Problems schlagen wir vor, die Log-Dateien nach der Archivierung zu formatieren, so dass ein sequenzielles Schreiben ohne Flashen möglich wird (vgl. Abb. 2, Schritt 3). Dies kann durch einfache Betriebssystemprozesse erfolgen. Problematisch bei der Untersuchung ist, dass SSD-Treibersoftware herkömmliche Dateisysteme emuliert und dadurch das Monitoring schwierig ist. So ist die Lokation von Speicherblöcken nicht zuverlässig nachvollziehbar. Über durchgeführte Experimente und Ergebnisse werden wir in Zukunft berichten.

4 Testframe

Zur Durchführung konkreter Messungen dient ein eigens in Java implementierter Benchmark-Testframe, welcher sich durch verschiedene Angaben parameterisieren lässt. Darunter fallen die Angabe konkurrierender Nutzer (die parallel DML-Befehle an die Datenbank senden), deren Startzeitpunkt (die Akteure starten um eine Zeiteinheit versetzt), die Anzahl der Tabellen und die Anzahl von `selects`, `inserts`, `updates` und `deletes` die ein Nutzer auf diesen Tabellen ausführt. Zusätzlich lässt sich über einen Parameter `repeat` festlegen, wie oft ein kompletter Durchlauf wiederholt werden soll.

Nachfolgend ein Ausschnitt des Parameter-Files:

```
#Benchmark Parameter File
#Fri Mar 27 01:29:57 CET 2009
users=10
sleep=100
tablenumber=5
tableinsertnumber=10
tableselectnumber=10
```

```
tableupdatenumber=10
tabledeletenumber=10
repeat=30
```

Es findet der Oracle Thin JDBC Treiber Verwendung [2]. In der Theorie bietet der Oracle OCI-Treiber zwar durch seine nativere Natur einen Performance-Vorsprung, jedoch ist man beim Benchmarking nur angewiesen auf den Vergleich von identischen Ausstattungen. Da nicht zu erwarten ist, dass sich der Datenbank-Treiber bei Zugriff auf eine SSD maßgeblich anders verhält als bei einer herkömmlichen HDD, steht hierbei die leichtere Entwicklung des Java-Testframes im Vordergrund.

Der Testframe generiert die angegebene Anzahl von Tabellen mit einem zentralen User, wobei drei Indexe auf drei Attributen (Datentypen `number`, `varchar`, `varchar2`) erzeugt werden. Danach wird die hinterlegte Anzahl von Test-Usern erzeugt. Diese arbeiten dann parallel auf den zentralen Tabellen statische SQL-Operationen ab: zuerst `inserts`, dann `selects`, gefolgt von `updates` und `deletes`.

Gemessen wird die Zeit für jeden SQL-Akteur, beginnend ab der ersten `insert`- und der letzten `delete`-Operation auf einer Tabelle, sowie die Zeit für jede der genannten SQL-Operationen selbst. Letztlich interessant ist die Gesamtlaufzeit eines Testlaufes samt all seiner User, wobei für ein verwertbares Ergebnis mehrere Testläufe vollzogen und Mittelwerte gebildet werden.

5 Testlauf-Beispiel

Es folgen konkret durchgeführte Testläufe, mit Logdateien auf einer HDD und einer SSD. Hier wurden Messdaten mit 30 Durchläufen erfasst. Um freier von dritten Einflüssen wie der Arbeit von Garbage-Kollektoren zu sein, wurden die restlichen Werte nicht zu hoch gewählt. Konkret wurde mit folgender Parameter-Konfiguration gearbeitet:

```
-------------------------------------------------------------------------
|  DB-Type          : oracle
|  AutoCommit       : true
|  Sleep Intervall  : 100
|  Test Users       : 10
|  Quota Amount     : 50
|  Table Number     : 5
|  Table Inserts    : 10
|  Table Selects    : 10
|  Table Updates    : 10
|  Table Deletes    : 10
|  Repeat           : 30
-------------------------------------------------------------------------
```

Hier das Ergebnis der HDD-Durchläufe:

```
################ Benchmark Overall Analysis ###############
 Results for all actors and all repeat cycles!
 Overall time (all cycles)         :  324.604 ms
 Average overall time (all cycles) :   10.820 ms
 Average actor time                :    1.081 ms
 Average actor insert time         :       49 ms
 Average actor select time         :       18 ms
 Average actor update time         :      425 ms
 Average actor delete time         :       18 ms
###########################################################
```

Hier das Ergebnis der SSD-Durchläufe:

```
1   ############### Benchmark Overall Analysis ###############
     Results for all actors and all repeat cycles!
3    Overall time (all cycles)         :  201.598 ms
     Average overall time (all cycles) :    6.719 ms
     Average actor time                :      671 ms
6    Average actor insert time         :       46 ms
     Average actor select time         :       16 ms
     Average actor update time         :       75 ms
9    Average actor delete time         :       16 ms
    ###########################################################
```

6 Fazit

Die ersten Messungen zeigen bereits einen spürbaren Performance-Vorteil von SSD. Es kann festgestellt werden, dass lesende und weitreichend schreibende Zugriffe schneller möglich sind als bei herkömmlichen HDDs. Die Messergebnisse zeigen auf, dass die mittleren Gesamtlaufzeiten bei der Nutzung von SSDs stets um einige hundert Millisekunden kürzer sind als bei der Verwendung von HDDs als Logdatei-Speicher.

Diese Ergebnisse basieren zu einem Großteil auf den physikalischen Vorzügen von Solid State Disks. Es sind jedoch auch weitere Optimierungen denkbar, wie beispielsweise die Variation der Logdatei-Größe von standardmäßig 51200 KB auf einen Wert, der einer Blockeinheit auf der SSD entspricht. Weiterhin könnte man durch leeres Beschreiben inaktiver Logdateien den Vorteil des sequenziellen Schreibens verstärkt zum Vorschein kommen lassen. Auch Optimierungen außerhalb der Logdateien sind denkbar - dies soll in weiteren Untersuchungen betrachtet werden.

Literatur

[1] Andre Faustmann, Michael Höding, Gunnar Klein und Ronny Zimmermann: *Oracle-Datenbankadministration für SAP*. Galileo-Verlag, 2007.

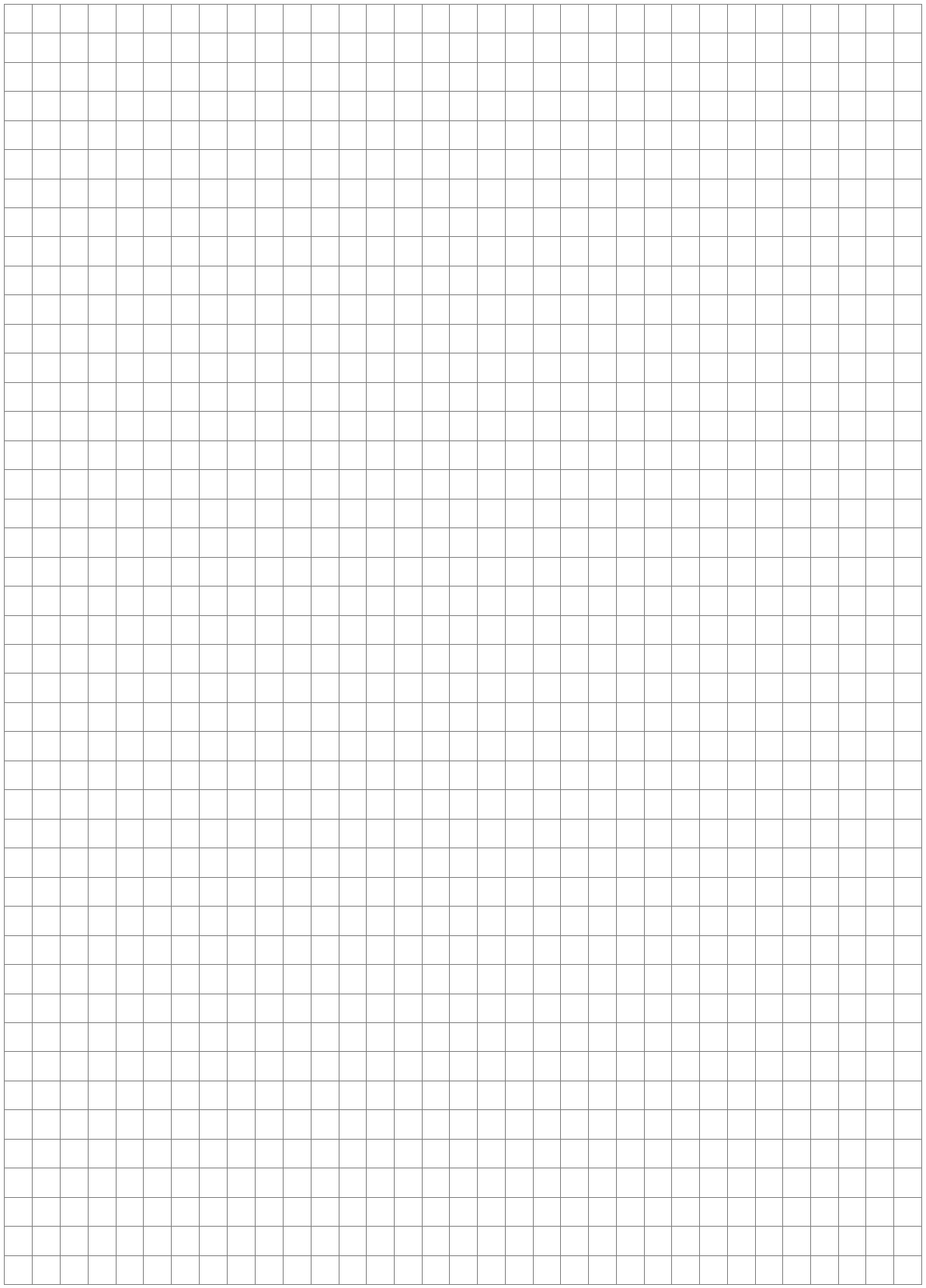
[2] Bales, Donald: *Java programming with Oracle JDBC*. O'Reilly & Associates, Inc., Sebastopol, CA, USA, 2002.

[3] Benz, Benjamin und Boi Feddern: *Festplatten ade.* c't Magazin für Computer und Technik, 25(21):100–105, 2007.

[4] Härder, Theo, Karsten Schmidt, Yi Ou und Sebastian Bächle: *Towards Flash Disk Use in Databases - Keeping Performance While Saving Energy?* In *BTW*, Seiten 167–186, 2009.

[5] Höding, Michael: *Flash-Disks - Neue Speicherhierarchieen für Datenbankmanagementsysteme?* In Höpfner, Hagen und Friederike Klan (Herausgeber): *Grundlagen von Datenbanken*, Jahrgang 01/2008 Reihe *Technical Report*, Seiten 126–130. School of Information Technology, International University in Germany, 2008.

[6] Saake, Gunter und Andreas Heuer: *Datenbanken: Implementierungstechniken*. MITP-Verlag, 1999.



Self-tuning of Data Allocation and Storage Management: Advantages and Implications

Andreas Lübcke
Otto-von-Guericke-University Magdeburg
Department of Computer Science
Institute for Technical and Business Information Systems
D-39016 Magdeburg, Germany
P.O. Box 4120
andreas.luebcke@ovgu.de

1 Introduction

Many important business applications use complex database management systems (DBMS). These DBMS have to be administrated and optimized for an optimal performance, especially in time-critical applications. Administration and optimization are very complex and costly tasks. Therefore, researchers and DBMS vendors focus on development of self-tuning techniques for a continuous adaption, e.g., the COMFORT automatic tuning project [6] or the MAPE approach by IBM [2]. However, the optimization and usage of self-tuning techniques for allocation and storage management of data are less investigated. DBMS vendors inform their customers about advantages and disadvantages of optimizing data storage in their user manuals and tuning guidelines. They recommend usage of functionality to optimize data storage with respect to the higher administration costs. But, DBMS vendors do not publish guidelines for this. Optimization of data storage is a complex (high administrative needs) task with many of options and parameters. For instance, the number of parameters for table space creation is huge, e.g., `page size` or database `partition group`. These two parameters affect essentially the performance of data allocation and storage management. In the scope of data allocation and storage management, many parameters dependencies and implications are unobserved. Our approach will observe the affect of the parameters. This paper presents first steps for better solutions and estimations for data allocation and storage management based on the parameter `page size` and its configuration.

2 Approaches for Automatic data allocation and storage management

In this section, we present current approaches from two DBMS vendors for automatic data allocation and to point out their limits. The first approach is from IBM used in DB2 to allocate data dynamically with `system managed spaces` (SMS) and `database managed spaces` (DMS). Oracle implemented another approach which provides a vertically integrated file system and volume manager.

2.1 IBM's Automatic Storage Approach

IBM's automatic storage approach uses `system managed spaces` and `database managed spaces` [1], which are elementary types of table spaces in DB2. In general, SMS table spaces are better suited for databases with many small tables, because they require less maintenance effort.

Data, indexes and large objects are stored within the same table space. DMS table spaces are recommended for large databases with large tables. A DMS table space consists of one or more container in form of a file or raw device. Thus, data, indexes and large objects can be separated from each other.

The hybrid approach of DB2 prevents administration overhead of DMS table spaces for many small tables, and take advantage of data allocation with DMS table spaces for large tables [7]. The huge benefit of this approach is the lowered administrative costs, because DB2 automatically increases the size of a database [3]. Hence, we do not have to think about creating table spaces (for database size), adding containers or observing table space usage. This option can only be set while creating a database, and cannot be changed afterwards.

While creating a new table space, DB2 performs the decision between SMS and DMS by itself. Regular and large table spaces will be created as DMS table space, because they contain mainly large tables and a huge amount of data. In contrast, user temporary table spaces and system temporary table spaces are created as SMS table space. These two table space types contain mostly small tables. A database administrator (DBA) can not influence this decision.

2.2 Automatic Storage Management by Oracle

Oracle uses another strategy to enhance the performance of data storage called: Automatic Storage Management (ASM). Oracle implements a volume manager for single-instance databases and a file system for application cluster configurations [4]. ASM uses disk groups to store data files. A disk group is a collection of disks which is managed as a unit by ASM. Inside a disk group, ASM provides a file system interface for Oracle database files. The data stored in a disk group is stripped or distributed.

In contrast to IBM, ASM is based on a new type of table space, and independent of `dictionary managed table spaces` (DMT) and `local managed table spaces` (LMT). LMT and DMT use different management of allocating `extents` (group of contiguous free blocks) to a `segment`. DMTs manage the `extents` by using the data dictionary. And each LMT manages its own free and used space by using a bitmap structure. Now, ASM does the `extent` management automatically. Regarding this fact, the administration gets less complex, because some parameters do not have to be considered anymore, e.g., `pctused` or `freelists`.

2.3 Limitations of Existing Approaches

The following considerations will summarize the state of the art for of automatic storage management. We point out the latest improvements, but we will also reveal limitations of existing approaches. An overview of following considerations can be found in Table 1.

The latest approaches of DBMS vendors principally improve the allocation of new space to existing data containers. DBAs do not need to monitor each data container, because DBMS automatically increases the size of data containers. These approaches reduce administration costs, but they also imply new considerations by DBAs. New parameters have been introduced in DB2 and Oracle like `initial size`, which sets the starting size of a table space in DB2. The `initial size` can be set by the database manager (perhaps a standard value), but no algorithms are given to reconstruct these estimations. So, we have to trust the database manager. Furthermore, partition groups

Functionality	Oracle	IBM
Auto resizing	yes	yes
Distribution scheme	manually	manually
Distributed allocation	automatically	automatically
Granularity	coarse	finer
Auto create	no	no
Setting parameters	manually	manually
Complexity	probably equal	increased
Migration	possible	no

Table 1: Summary of key features

have to be created and evaluated to distribute data in DB2. Similarly, a DBA has to take into account disk groups in Oracle. Once these distribution schemes have been created, the DBMS automatically allocates data. The granularity of distribution varies from DBMS to DBMS. In Oracle, the distribution only concerns with whole disks. Partition groups in DB2 follow the same strategy, but data belonging to one table space can be distributed by hand. Therefore, data containers of one table space can be distributed over several disks. In contrast to Oracles approach, several table spaces can be distributed over the same disks.

Nevertheless, DBAs have still to create each data container by hand for a new database. This implies a number of parameters have still to be set manually. In Oracle, some parameters disappeared, and some new parameters are introduced. The complexity may remain equally. In DB2, the complexity even has been increased by introducing new parameters. So, the simplified data allocation involves a more costly administration.

The migration of existing databases is another aspect, we have to consider. Oracle supports a migration to the ASM approach. A coexistence of different allocation approaches is supported, too. In contrast, DB2 does not support any migration. IBM's approach can only be used for the design of new databases. Either the option is set while creating a database or the database cannot use automatic data allocation.

3 Steps to Overcome Complexity of Optimizing Data Storage

This section discusses the need of complexity reduction to implement new algorithms for data allocation and storage management. We will explain our decision for the parameter `page size`. Finally, we introduce our approach using heuristics.

3.1 Research Strategy

Our previous considerations show the high complexity of optimizing data allocation and storage management. The complexity of this process has to be reduced, because we cannot develop an approach for all aspects at once. Due to the complexity of this process, we only consider one aspect for now.

Hence, we have to find a starting point to improve data allocation and storage management. An appropriate starting point should influence performance of data allocation directly. Afterwards, we can examine the dependencies to other aspects. This inductive strategy gives us the possibility to overcome the problem of high complexity. The results will be integrated into our approach. Afterwards, we will repeat this strategy for the different aspects of data allocation and storage management. In this way, we will gain a more general solution.

We decide to optimize the `page size` [5] in the first step. This decision is based on a simple but important reason. The data allocation using pages is one of the fundamental concepts of current relational DBMS. Hence, the `page size` is an important parameter regarding the performance of DBMS at all. Our approach will start to design an adviser for `page size` using heuristics.

3.2 Optimizing Data Storage Using Heuristics

In this part, we will describe the first steps for our approach. We use our knowledge base in database administration to derive heuristics for data allocation and storage management.

There are two problems regarding `page size`. First, the `page size` can be too small. Amplified, a too small `page size` implies probably that only one tuple fits into the page. But even worse, a tuple can be bigger than one page. The administration overhead and resource consumption will be excessive. Second, the `page size` has been chosen too large. Hence,

DBMS have to search in large pages (perhaps for small tuples), and performance (data access) will decrease.

At this point, we have again to reason the granularity. Oracle defines the `page size` for the whole database. We need to find a `page size` which fits all kind of data. In other words, the average cost of data access has to be minimized. In contrast, DB2 provides the opportunity to set the `page size` for each table space. The complexity of decision is increased by this choice, but on demand, the `page size` can be optimized for each single table in our database. As a result, we obtain a more powerful tool to improve data allocation and data access.

We set the `page size` to discrete values, but pages contain tuples which have not discrete values according to their `tuple size`. So, we need to know the `tuple size`, too. We can obtain this information from DBMS. If not, we compute the `tuple size` for an existing database using statistic functions. Given the number of rows and the size of a table, the `tuple size` is computed as:

$$tuple\ size = \frac{size\ of\ table}{number\ of\ rows}.$$

Whereby, the size of a table is estimated by DBMS. We need another way to compute the `tuple size`, if the DBMS does not provide the size of table calculation. We estimate the `tuple size` using `page size`, number of pages allocated for this table, and number of rows:

$$tuple\ size = \frac{page\ size * number\ of\ pages}{number\ of\ rows}$$

This estimation assumes that the pages are filled completely. This imprecise estimation does not work for database design, because we will not have data in a new database. We recommend a function based on number of attributes and their data types. Therefore, we need the length of each attribute ($length\,(A)$) of table (R). Our first proposal is:

$$tuple\ size = \sum_{A \in R} (length\,(A)) + overhead.$$

We can improve this estimation using another statistic. If supported, the average length of an attribute ($avg_length(A)$) is obtained from database statistics. We assume a more exact estimation, because we compute it with the real consumed space of an attribute. Therefore, we have only to replace $length\,(A)$ by $avg_length(A)$ in our algorithm. Again, the adjusted algorithm is only adaptable for existing databases.

A function that maps the `tuple size` to the `page size` is needed, i.e., number of tuples stored in one page. Therefore, the optimal value of tuples per page has to be figured out. Extensive case studies are necessary to evaluate the optimal value. Perhaps, we have to evaluate the optimal value for each DBMS, because internal algorithms and specific implementations will influence the result. In addition, we have to point out thresholds for the `tuple size` regarding the `page size`; because we need to map continuous values (`tuple size`) onto discrete values (`page size`). We assume closed intervals for `tuple sizes` and assign them to each possible discrete value of the `page size`. In this way, we can assign each table to the corresponding table space (optimal `page size`). Therefrom, we can develop a solution for a fine granularity regarding the `page size`.

In case of a coarse granularity regarding the `page size`, e.g., used by Oracle, we need to extend our algorithm. We assume one `page size` for an amount of data or even a database. To compute an optimal value for the `page size`, we have to introduce a weight function. There are two possible approaches. First, we can compute the average `tuple size` for the whole database. The result seems to be a crude estimation. A second approach could compute `tuple sizes` for each table (t). We assume the number of rows of each table as weight to their `tuple size`. The sum of weighted `tuple sizes` is divided by the total number of rows of a database (D). The

estimated `tuple size` is defined as:

$$estimated\ tuple\ size = \sum_{t \in D} \frac{number\ of\ rows(t) * tuple\ size(t)}{total\ number\ of\ rows(D)}$$

This algorithm only works for existing databases. Furthermore, the algorithm estimates the access rate of data is evenly distributed.

Both algorithms for `page size` estimation mostly based on the same functions. So, we can develop one framework with two different aspects regarding the granularity of `page size`.

4 Conclusion

This paper points out the state of the art of automatic data allocation and storage management in current DBMS. We discussed the limitations of current approaches and considered the need of complexity reduction. We argued that the `page size` is an appropriate starting point for the future steps. Finally, we used our knowledge base to discover heuristics for `page size` optimization. Our first step was the development of algorithms to estimate the optimal `page size` for a table or a whole database.

We will perform extensive case studies to derive a function which computes the optimal number of tuples per page. To develop a useful adviser for `page size` configuration, we have to extend our approach. First, we will improve our estimations of `tuple size` (including the *estimated tuple size*) for new databases. Second, we integrate the workload into the `page size` estimation for a whole database.

References

[1] W.-J. Chen, A. Fisher, A. Lalla, A. D. McLauchlan, and D. Agnew. *Database Partitioning, Table Partitioning, and MDC for DB2 9*, volume First Edition. IBM Redbooks, 2007. Draft Document June 18,2007.

[2] IBM. An architectural blueprint for autonomic computing. White Paper, June 2006. Fourth Edition, IBM Corporation.

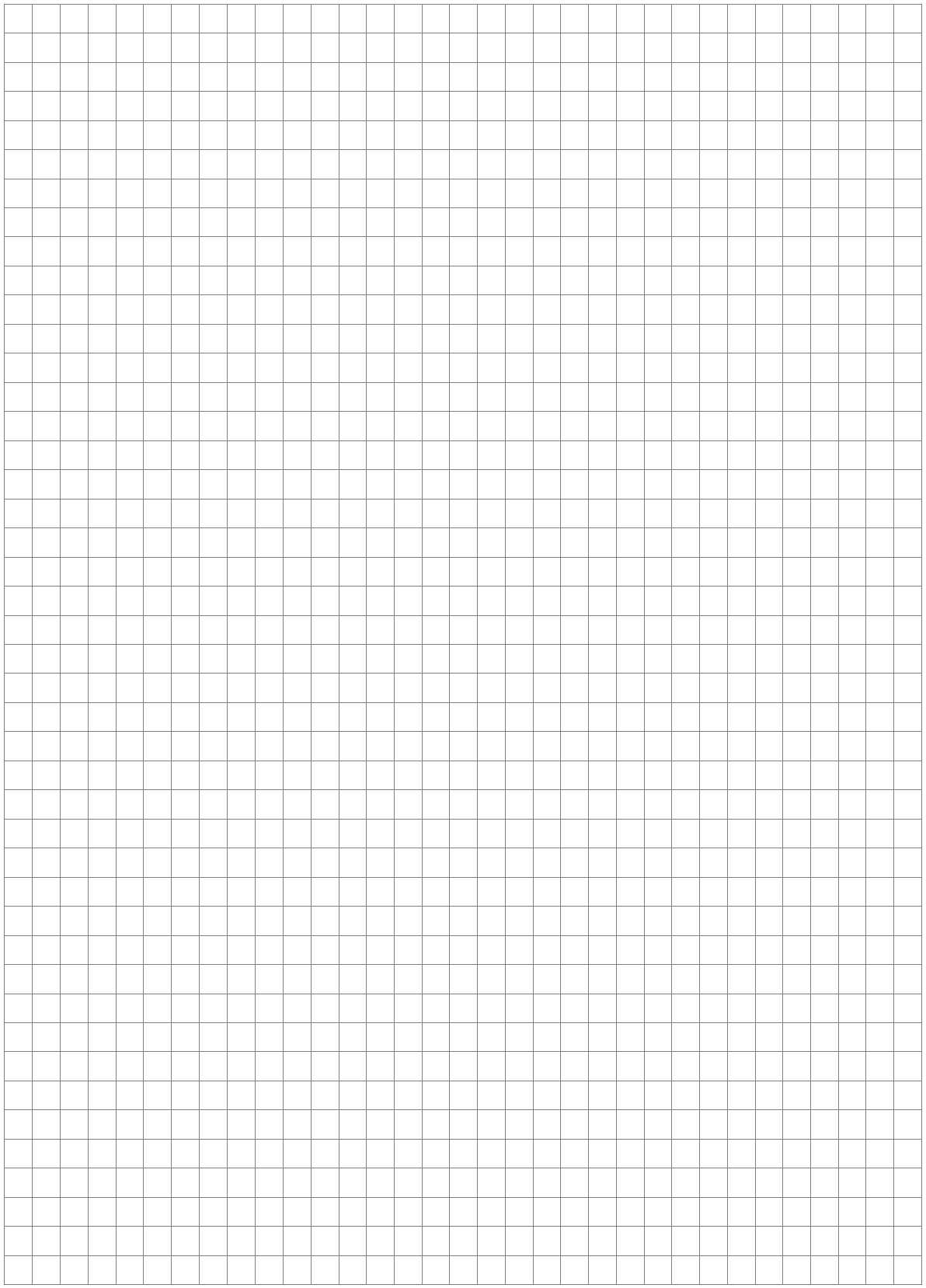
[3] IBM. Introducing db2 9, part 4: Automatic and other enhancements in db2 9. Technical Article, June 2006. IBM Corporation, http://www.ibm.com/developerworks/data/library/techarticle/dm-0606ahuja2/index.html.

[4] Oracle. Automatic storage management: Technical overview. Technical White Paper, November 2003. Oracle Corporation.

[5] P. Rob and C. Coronel. *Database Systems: Design, Implementation, and Management*, volume Seventh Edition. Course Technology.

[6] G. Weikum, C. Hasse, A. Moenkeberg, and P. Zabback. The COMFORT Automatic Tuning Project, Invited Project Review. *Information Systems*, 19(5):381–432, 1994.

[7] P. Zikopolous, G. Baklarz, and L. Katsnelson. *IBM DB2 Version 9 New Features*. McGraw-Hill Osborne, 2007. Page 75 et seq.



Query Processing on Multi-Core Architectures

Frank Huber and Johann-Christoph Freytag
Department for Computer Science, Humboldt-Universität zu Berlin
Rudower Chaussee 25, 12489 Berlin, Germany
{huber,freytag}@dbis.informatik.hu-berlin.de

Abstract

The upcoming generation of computer hardware poses several new challenges for database developers and engineers. Software in general and database management systems (DBMSs) in particular will no longer benefit from performance gains of future hardware due to increase clock speed, as it was the case for the last 35 years; instead, the number of cores per CPU will increase steadily. Today's approach is to run each query on a single core or only a few different cores using parallel query execution. This approach suffers from several problems (e.g. contention problem) and therefore leads to poor speed up and scale up behavior. These observations open several important research questions on how to use the new multi-core CPU architecture for improving the overall performance of DBMSs.

This paper outlines our approach for query processing on multi-core CPU architectures. We present an abstract architecture view for multi-core CPUs, meta operators to control and to interact with the hardware, and a new query operator model that makes use of the meta operators to control the parallel execution of a query over different cores. We illustrate how each of these parts fits in our framework for query processing on multi-core architectures.

1 Introduction

The next generation of computer hardware poses several new challenges for software developers and engineers. Due to physical limitation, next generation CPUs will no longer scale in clock rate. Instead, a scale in number of cores within a single CPU is most likely. The extended address scheme of 64bits allows us to use large main memory installations even for low budget systems. With almost infinite memory, we can assume that a majority of databases fit entirely into main memory. Hence, disk - main memory latency and bandwidth will become less important factors. Instead, main memory - CPU latency mismatch and limited bandwidth form the "memory wall". Hence, parallelization and cache efficiency are major objectives.

Parallelization of query execution can be viewed in three dimensions [1]. The first dimension, *interquery parallelism*, determines the number of queries that can be executed concurrently. The second dimension, *interoperator parallelism*, defines the number of operators in a single query executed in parallel. The last dimension, *intraoperator parallelism*, indicates to what extend a single operator is parallelized.

Most of today's DBMSs make use of the first dimension and a specialized version of the second - *horizontal interoperator parallelism*[1]. Furthermore, queries are currently optimized to minimize disk main memory transfers. These limitations and the increasing number of available cores on a single chip lead to serious problems. E.g. running hundreds of queries in parallel will result in different contention problems. The first problem relates *data contention*, i.e. different queries require access to the same object at the same time. Even without data contention, stagnancy in CPU clock rates will result in poor speed up performance. *Resource contention* is the second problem: different queries utilize the same hardware, especially CPU - main memory

[1] Exchange Operator[2]

bandwidth and CPU caches. If, for example, two queries are executed on two cores that share the same physical cache, both cores will try to prefetch necessary data thereby causing thrashing of the cache most likely. Such behavior results in additional data loads (i.e. CPU stalls, waste of bandwidth). In summary, today's approach may results in faster query execution on single core CPUs, but not on multi-core CPUs.

2 A Framework for Query Processing on Multi-Cores

The new generation of hardware technology raises two main research questions. First, how do we handle the memory wall? Second, how do we use the new opportunities that come with multi-core architectures?

Concepts like vertical fragmentation [3, 4] seem to be a good choice when it comes to attack the memory wall. In our opinion the open challenge is, how to relate these new concepts to query processing highly parallelized for CMPs[2]. For this reason, we have developed a framework that simplifies the development and the maintenance of DBMSs on multi-core architectures. Among others, our work covers the following issues:

- Integration of multi-core in the whole process of query processing;
- Finding parallelisms in query optimization;
- The influence of multi-core on cost estimation and plan selection;
- The choice of execution model that best fits query execution on CMPs;
- Ease of development and maintenance of DBMSs on different multi-core architectures;
- The design and implementation of a global optimizer for such systems;
- Is there an optimal CMP architecture with respect to DBMSs? Which functionality should be provided by such a chip?

Regarding the complete flow of query processing, we focus on the two phases (i) query optimization and (ii) query execution. Both phases are reflected in our architecture framework that has the following layout: The bottom layer consists of an abstract view of the hardware architecture and a physical data model. On top, the next level comprises of meta operators to control the hardware by using the abstract view. A model of query execution and query optimization form the upper level of our framework. Query optimization employs the abstract view to create optimized QEPs from a global point of view as well as from a local point of view. Query Execution utilizes the meta operators to describe and to implement its operators over the physical data model. In the following we briefly outline the physical data model and detail the abstract architecture view, meta operators, and query execution.

Physical Data Model

On multi-core architectures, we must consider the physical (memory) layout of the data very carefully for efficient query execution. In order to provide high inter- and intraoperator parallelism the physical data model provides data structures that are cache efficient and optimal for partitioning. Since vertical fragmentation of the data has been shown to be a good choice for cache efficiency, we will use this representation in the following. Like C-Store [4], we use bitmap indices to represent sets of ids.

[2] chip multiprocessor

Abstract Multi-Core Architecture View

Highly parallelized query execution depends very much on the actual hardware architecture (i.e. what kinds of functions are supported? How are cores interconnected? etc.). To be independent from specific architectures and for simplicity, we use an abstract multi-core architecture view. This view focuses on the common and important aspects of the different architectures, thus hiding some of the complexities of modern CPU architectures. Our foci are caches and cores. Therefore, our view contains these two hardware components. In addition, we add a logical (non-existent) component to group different cores and caches. We call this logical component a **workgroup**. During query execution an operator might be distributed over one or more workgroups. Different setups are conceivable. First, each core in a workgroup executes the same operation on different data. Second, a complex operation is decomposed into N smaller and simpler, operational fragments. Each of these N fragments is executed by one core of a workgroup of size N, thus forming a worker queue that feeds each other. Lastly, a combination of the previous two alternatives is possible, thus forming a hybrid combination. Furthermore, components - in particular cores - might be shared among different workgroups. In particular on SMT[3] systems, cores might be logically shared among different workgroups. We emphasize that workgroups do not have to be equally equipped, e.g. it is possible to create workgroups with a single core and without any assignment of cache.

Meta Operators

We define a number of meta operators for the following reasons. First, most modern programming languages do not provide any support for expressing cache or multi-core related issues. For example, in *C* or *C++* developers have to use intrinsic or inline assembler statements for this reason. But none of those statements is hardware independent, thereby leading to a high degree of CPU vendor / CPU architecture dependency and huge efforts in software development. Second, our goal is to separate the actual hardware implementation and optimization of operators from their development. Third, with our meta operators, we would like to give developers the ability to express all their knowledge (e.g. data access pattern) in a simple and comprehensive manner.

For example, consider a developer of a DBMS writing a new join operator that might be executed in parallel on two cores and needs some data exchange during its processing. The developer might use the meta operator *TransferData* to initiate a data transfer from one core to another one. The actual implementation of this operator is very system specific: A hardware expert will implement and optimize those meta operators for each individual hardware architecture.

Our abstract (hardware) architecture consists of meta operators controlling the behavior of cores, caches, and workgroups. We designed operators for resource management like allocation and deallocation of workgroups, for communication and data transfers between workgroups and cores, as well as for work scheduling. Our meta operators and abstract view consider cores and caches as regular system resources that might be allocated and deallocated. Allocated resources may not be shared by different system processes. This restriction assures that it is always known in which state each resource is at any give execution point.

To provide the reader with a flavor of the semantics and possible implementation of meta operators, we depict three of them in the following:

- $AllocWorkGroup(C, S)$ allocates a workgroup of C cores and a cache size of S and returns some kind of handle. This handle can be used to manage and to interact with the workgroup.
- $AssignTaskToWorkGroup(WG, T)$ assigns a task T to the workgroup with handle WG. Tasks are code pieces specific for the workgroup setup.

[3] simultaneous multithreading

• $TransferData(WG_1, WG_2)$ transfers data from workgroup WG_1 to workgroup WG_2. One implementation could be to use an interconnection bus if available. Alternatively, one could write the data back into main memory and the load the data into a different cache, or just do not perform any operation if WG_1 and WG_2 sharing the same physical cache.

Query Execution and Operator Model

With the abstract view and our meta operators in hand, we are able to describe the behavior of different multi-core CPU architectures in a simple manner. One of our questions was to identify an execution model that fits best for query execution on CMPs. It is our belief the iterator model as the operator model for query execution is inappropriate for query execution on multi-core architectures. The main reason for this opinion is that by its nature the iterator model is highly synchronized i.e. only one operator is executed at any time[4]. This behavior results in no vertical parallelism. In particular, vertical parallelism is very promising for CMP systems as it provides an opportunity to create query operators that run in parallel and that feed each other with data (asynchronously). This kind of parallelism might greatly utilize the sharing of caches (i.e. the output of one operator is the input another one) thus reducing CPU main memory bandwidth utilization. This reduction in bandwidth might be used to increase the degree of horizontal parallelism, as the degree of horizontal parallelism is bounded by the available bandwidth.

In the following, we outline some ideas for designing a new **asynchronous** operator model that consists of two kinds of operators. The first kind is a set of already known operators like join, selection, aggregation, projection, and grouping. Each of these operators is implemented according to our physical data model and uses different meta operators. The operator model uses pipes and buffers for data exchange i.e. the producing operator is responsible for transferring data to all its consumers. The second set currently consists of one special operator, named **watchdog**. This operator is responsible for managing the execution of a query by processing the QEP bottom-up. It is the only instance that knows about the current execution state. The watchdog uses the abstract architecture view to distribute the QEP and its operators among the different existing workgroups. The workgroups are managed and controlled by the meta operators. A workgroup regularly sends signals to the watchdog to give information about its current status. To make the query execution and the interaction of operators more concrete we give the following example.

Example

The example is the simplified Query 1 of the TPC-H benchmark [5]. Figure 1 depicts the query

```
SELECT l_returnflag,
       l_linestatus,
       SUM(l_quantity)
FROM lineitem
GROUP BY
       l_returnflag,
       l_linestatus
```

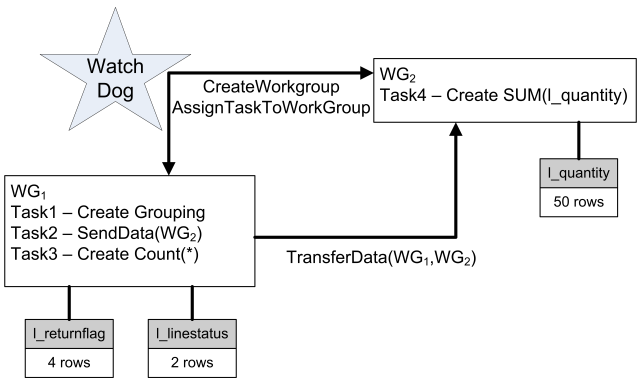


Figure 1: Query Execution of the example

[4]beside horizontal parallelism

execution of the sample query. In our case the watchdog creates Workgroup WG_1 and assigns Task 1 $CreateGrouping$. The $CreateGrouping$ task creates all possible groupings using the columns $l_returnflag$ and $l_linestatus$. A possible implementation could be to create the workgroup WG_1 with 4 cores. Then, each core loads one attribute value[5] from attribute $l_returnflag$ and intersects it with each attribute value of $l_linestatus$. This execution ensures that all bitmap indices of $l_returnflag$ and $l_linestatus$ must only be loaded once. The result of this task are 4 different non empty groupings on $l_returnflag$ and $l_linestatus$. Meanwhile the watchdog allocates Workgroup WG_2 with an appropriate number of cores and an appropriate cache of a predetermined size. Furthermore, the watchdog assigns the second ($SendData(WG_2)$) and third Task ($CreateCount(*)$) to WG_1 and the fourth Task $CreateSum(l_quantity)$ to WG_2. The task $SendData$ uses the $TransferData$ meta operator to send all previously calculated results to WG_2. Finally, each core of WG_1 counts the bitmap index of one group to calculate the aggregation $Count(*)$ as requested by Task 3. After WG_1 has send its data to WG_2 and notified WG_2 that now all data is available, WG_2 starts executing Task 4, by scanning through column $l_quantity$ and calculating the intersection according to each group. At the end, the results of both workgroups must be joined to form the final result. This latter part is not included in Figure 1.

3 Conclusion and Outlook

In this paper, we presented our ideas for a new query optimization and query execution model that takes advantage of multi-core CPU architectures. We outlined our underlying physical data model and introduced an abstract view of multi-core architecture together with meta operators. Both, the abstract view and meta operators will hide some heterogeneity and complexity of different hardware architectures. Furthermore, we presented a new operator model for query execution. We believe that our model supports intraoperator parallelism as well as interoperator parallelism (both horizontal and vertical). Lastly, we illustrated our framework on an examples, to provide a basic understanding of, how queries are executed using our framework and our query execution model.

The next steps of our work are a more complete design and implementation of our concepts. The implementation should provide useful feedback about our operator model. Although our current work focuses on query execution new approaches for query optimization in a multi-core execution environment are equally important; they will be part of our future work.

References

[1] G. Graefe. Query evaluation techniques for large databases. *ACM Comput. Surv.*, 25(2):73–170, 1993.

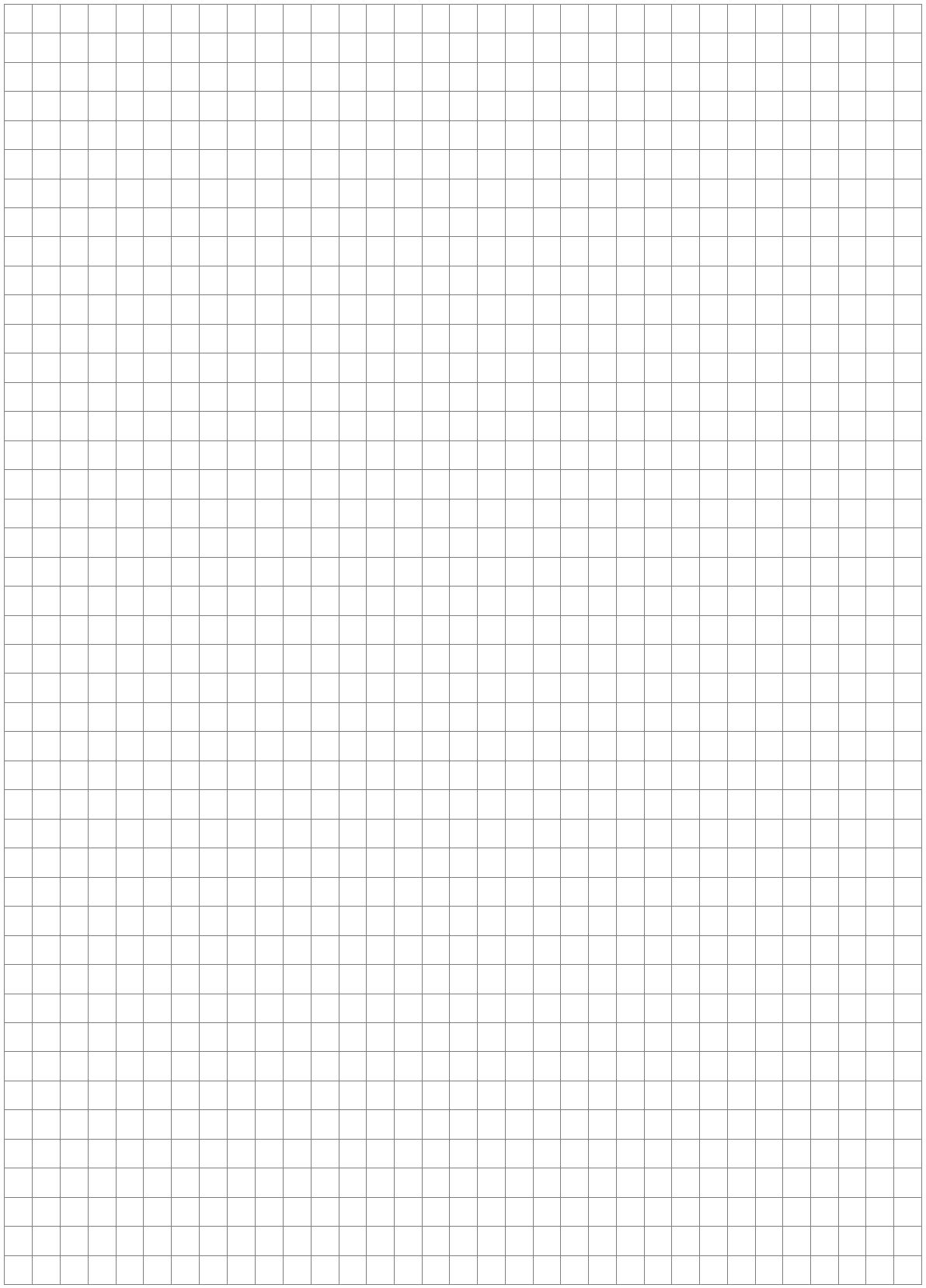
[2] G. Graefe and W. J. McKenna. The volcano optimizer generator: Extensibility and efficient search. In *ICDE*, pages 209–218, 1993.

[3] M. L. Kersten, S. Manegold, P. A. Boncz, and N. Nes. Macro- and micro-parallelism in a dbms. In *Euro-Par*, pages 6–15, 2001.

[4] M. Stonebraker, D. J. Abadi, A. Batkin, X. Chen, M. Cherniack, M. Ferreira, E. Lau, A. Lin, S. Madden, E. J. O'Neil, P. E. O'Neil, A. Rasin, N. Tran, and S. B. Zdonik. C-store: A column-oriented dbms. In *VLDB*, pages 553–564, 2005.

[5] Transaction Processing Performance Council. TPC Benchmark^{TM} H (decission support), 1999.

[5] value and bitmap index



Entwicklung eines Evaluations-Frameworks für instanzbasierte Ontologie-Matching-Verfahren

Katrin Zaiß
Institut für Informatik
Heinrich-Heine-Universität Düsseldorf
D-40225 Düsseldorf, Deutschland
zaiss@cs.uni-duesseldorf.de

Zusammenfassung

Ontologien sind ein weit verbreitetes Modell zur Repräsentation von Wissen, und werden u.a. im Semantic Web eingesetzt. In vielen verschiedenen Anwendungsgebieten wie z.B. der Informationsintegration, ist ein Matching der Ontologien notwendig. Es existieren schon einige Matchingsysteme, deren Qualität durch die Einführung von und die Teilnahme an Evaluations-Initiativen wie der OAEI, gut miteinander vergleichen werden kann. Für instanzbasierte Matcher bzw. für Systeme, die hauptsächlich Instanzinformationen zum matchen nutzen, gibt es aber kaum geeignete Test-Ontologien oder -Szenarien. Dieser Beitrag soll die Anforderungen an ein solches Evaluations-Framework spezifizieren und erste Ideen zur Umsetzung eines solchen aufzeigen.

1 Einleitung

Ontologien sind ein weit verbreitetes Modell zur Repräsentation von Wissen, und werden u.a. im Semantic Web eingesetzt. Ziel ist es vor allem das Wissen maschinenlesebar zu machen. So können z.B. Zusammenhänge zwischen Personen und personenbezogenen Daten wie Email o.ä. durch die Verwendung einer Ontologie eindeutig definiert werden. Das Matching von verschiedenen Ontologien (oder auch Schemata) kann in vielen verschiedenen Anwendungen notwendig oder sinnvoll sein und ist daher ein recht gut erforschtes Gebiet. Es existieren einige Matching-Systeme, wie z.B. Coma [DR02], Automatch [BM02], QOM [ES04] oder FCA-Merge [SM01], die verschiedene Matcher verwenden um ein möglichst gutes Mapping zu finden. Die verschiedenen Methoden können grob unterteilt werden in konzept- und instanzbasierte Matcher (siehe [RB01]). Konzeptbasierte Matching-Methoden nutzen die Metainformationen der Konzepte, wie z.B. Label, Kommentare, Datentypen o.ä. , welche mit geeigneter Distanz- oder Ähnlichkeitsfunktionen, wie der Edit-Distanz z-b. , verglichen werden. Dies ist ein sehr sinnvoller Ansatz, falls die Informationen recht ähnlich sind und die Struktur der Ontologien nicht zu unterschiedlich ist. Durch die Verwendung externer Quellen wie Thesauri, z.B. WordNet, können sogar Synonyme oder Homonyme entdeckt werden. Dennoch gibt es einige Konflikte, die nicht immer mit konzeptbasierten Methoden gelöst werden können. Ontologien werden normalerweise manuell erstellt, so dass die Metainformationen immer das subjektive Verständnis des Entwicklers widerspiegeln. Möglicherweise ergeben Metainformationen auch nur in einem sehr speziellen Kontext Sinn oder Konzepte werden unterschiedlich detailliert/strukturiert dargestellt. In diesen Fällen können instanzbasierte Matching-Methoden helfen semantisch äquivalente Klassen trotzdem zu finden. Instanzen liefern in ihrer Gesamtheit sogar mehr Informationen über ein Konzept als die Metainformationen. Die Schwierigkeit besteht darin diese Instanzinformationen in geeigneter Form aufzubereiten, so dass sie vergleichbar sind. Es wurden einige instanzbasierte Matcher entwickelt wie z.B. Glue [DMDH04], Coma++ [EM07], oder [ZSC08]. Das Mapping

wird entweder bestimmt mit Hilfe von Instanz-Duplikaten oder durch die Berechnung von Eigenschaften wie Durchschnittswerte, Wortverteilungen o.ä.; andere dagegen klassifizieren Instanzen mit Hilfe von Bayes-Klassifikatoren.

Um instanzbasierte mit konzeptbasierten Methoden oder auch komplette Systeme untereinander vergleichen oder testen/verbessern zu können, müssen entsprechende Evaluations-Frameworks definiert werden. Innerhalb des Frameworks sollten verschiedene Ontologien und Testszenarien enthalten sein, die einige Anforderungen erfüllen müssen, z.B. verschiedenartig strukturierte Ontologien. Es existieren schon derartige Frameworks, wie z.B. das von der Ontology Alignment Evaluation Initiative entwickelte, die aber vor allem für instanzbasierte Matching-Algorithmen nicht gut geeignet sind.

Der Rest des Papers ist wie folgt organisiert: In Kapitel 2 werden verwandte Arbeiten beschrieben und deren Defizite in Bezug auf instanzbasierte Matching-Verfahren erläutert. Kapitel 3 definiert die Anforderungen an ein Evaluations-Framework, deren Umsetzung in Kapitel 4 beschrieben wird. Abschließend gibt es eine Zusammenfassung und einen Ausblick in Kapitel 5.

2 Verwandte Arbeiten

Es gibt schon einige Frameworks zur Evaluation von Ontologie-Matching-Verfahren. Als wichtigstes ist die Ontology Alignment Evaluation Initiative (kurz: OAEI, [Oae08]) zu nennen. Die OAEI veröffentlicht jährlich verschiedene Testszenarien, die jedes Matching-System durchführen kann. Die Ergebnisse werden bei einem Workshop im Rahmen der International Semantic Web Conference veröffentlicht und diskutiert. Für unsere Zwecke besonders interessant sind die Benchmark-Tests, weil dort viele verschiedene Ontologien vorhanden und sehr viele verschiedene Tests durchgeführt werden. Zusätzlich sind die korrekten Alignments angegeben, was die Evaluation, d.h. vor allem die Bestimmung von Precision und Recall, vereinfacht. Die Referenz-Ontologie besteht aus 33 Konzepten, 64 Attributen (40 Objekt-Eigenschaften und 24 Datentyp-Eigenschaften), 56 Instanzen und 20 anonymen Instanzen. Insgesamt gibt es zusätzlich zur Referenz-Ontologie noch 50 weitere Ontologien, bei denen es sich immer um eine modifizierte Referenz-Ontologie handelt. Die Modifikationen umfassen die folgenden:

- Einbauen von Rechtschreibfehlern,
- Verändern der Schreibweise (Groß-/Kleinschreibung etc.),
- Ersetzen von Konzeptbezeichnungen durch ihre Synonyme oder zufällige Strings,
- Entfernen oder Übersetzen von Kommentaren,
- Übersetzen in eine andere Sprache (z.B. Französisch),
- Generalisierung der Sprache, Erweiterung von Klassen,
- Abflachung, Erweiterung oder Unterdrückung der Hierarchie.

Diese modifizierten Ontologien werden gegen die Referenz-Ontologie gematcht, und Precision und Recall werden mit Hilfe der angegeben korrekten Alignments bestimmt. Es ist wichtig anzumerken, dass hier nur 1:1 Korrespondenzen zwischen Konzepten und Attributen gefunden werden können bzw. sollen, und auch nur diese in den vorgegebenen Alignments zu finden sind. Die Struktur des Benchmarks ist sehr gut geeignet um konzeptbasierte Matcher bzw. Systeme, die vorwiegend solche enthalten, zu testen. Die sehr geringe Anzahl an Instanzen benachteiligt aber eindeutig (vorwiegend) instanzbasierte Matching-Systeme.

Ein weiteres Framework, welches aber eher zum Matchen von Instanzen geeignet ist, ist das ISLAB Instance Matching Benchmark. Die Referenz-Ontologie enthält 5 Klassen, 17 Attribute (4 Objekt-Eigenschaften, 13 Datentyp-Eigenschaften) und 302 Instanzen. Auch hier handelt es

sich bei den weiteren Ontologien um Modifikationen der Referenz-Ontologie, allerdings beziehen sich die Modifikationen ausschließlich auf die Instanzebene. Die Instanzwerte werden verändert, in dem Rechtschreibfehler eingefügt, Werte gelöscht/permutiert oder gleiche Instanzen in unterschiedliche Klassen eingeordnet werden. Die Ontologien dieses Benchmarks enthalten nur wenige Klassen, so dass die entsprechenden Tests für Matchingssysteme, die nicht ausschließlich auf Basis von Instanz-Matching arbeiten, nicht sehr geeignet sind.

Ein kürzlich neu entwickeltes Benchmark ist das STBenchmark [ATV08]. An einer vorgegeben Eingabe-Ontologie (z.B. aus DBLP oder BioWarehouse) werden verschiedene Transformationen durchgeführt, so dass jeweils ein Mapping-Szenario entsteht. Die erforderlichen Instanzen werden mit Hilfe des Datengenerators ToXGene künstlich erzeugt, was ein Nachteil dieses Systems ist. Zusätzlich ist die Anwendung des Systems nicht leicht verständlich (was den Anforderungen an ein Evaluations-Framework widerspricht, siehe nächstes Kapitel), und die korrekten Alignments sind nicht angegeben, was eine Evaluation erschwert.

3 Anforderungen

Wenn man ein Evaluations Framework entwickeln will, dann gibt es einige Prinzipien, die man beachten sollte. Im Folgenden sollen diese, wie in [ES07] beschrieben, kurz erläutert werden.

- Systematik: Die Tests müssen eindeutig und nachvollziehbar sein, und ihre Durchführung, auch zu verschiedenen Zeitpunkten, muss vergleichbar sein.
- Kontinuität: Eine kontinuierliche Wiederholung/Wiederholbarkeit der Tests soll gegeben sein, damit eine Entwicklung/Verbesserung festgestellt werden kann.
- Qualität und Quantität: Die Definition der Bewertungsregeln muss exakt und unmissverständlich formuliert sein. Außerdem sollte die Qualität der Test-Ontologien so gut wie möglich sein und keine der Test-Sets darf eine bestimmte Klasse von Matching-Systemen bevorzugen.
- Verbreitung: Das Benchmark und die Evaluationsresultate sollten frei zugänglich sein.
- Verständlichkeit: Die Resultate sollten analysiert werden können und für alle verständlich sein. Daher sollten nicht nur die allgemeinen Resultate, sondern auch die von den Systemen berechneten Alignments zur Verfügung gestellt werden.

Diese Prinzipien gelten allgemein für die Erstellung von Evaluations-Frameworks. Für unser Framework, welches auch bzw. insbesondere instanzbasierten Matching-Methoden eine Evaluation erlauben soll, definieren wir zusätzlich folgende Anforderungen:

- große Anzahl von Instanzen: Um die Skalierung und auch die Qualität von Matching-Algorithmen in möglichst realitätsnahen Szenarien testen zu können, muss die Anzahl der Instanzen ausreichend sein. Zudem sollten die vorhandenen Werte hinreichend unterschiedlich sein und nicht nur aus einigen wenigen aber oft wiederholten Instanzen bestehen (es sei denn, dies ist eine Eigenschaft des dazugehörigen Attributs).
- An- und Abwesenheit von Duplikaten: Wie oben beschrieben nutzen einige Systeme die Anwesenheit von Duplikaten um ein Mapping zu bestimmen. Da diese Systeme weder benach- noch bevorteiligt werden soll, sollte es Szenarien mit und ohne Instanz-Duplikate geben.
- unterschiedliche Strukturen: Ontologien können unterschiedlich strukturiert sein, d.h. sie sind z.B. unterschiedlich detailliert, obwohl sie semantisch ähnlich sind, oder enthaltenen andere/zusätzliche Relationen oder Attribute.

- unterschiedliche Formatierung: Instanzen können bei gleicher Semantik unterschiedlich formatiert sein, ein gutes Beispiel dafür ist das Datum. Um zu testen, in wie weit Matcher semantisch ähnliche aber unterschiedlich formatierte Instanzen verwenden können, sollen unterschiedliche Variationen in verschiedenen Ontologien vorhanden sein.

- Einbeziehung von Rechtschreibfehlern: (Reale) Ontologien werden größtenteils von Menschen erstellt, so dass die Instanzen natürlicherweise auch Rechtschreibfehler wie eingefügte oder ausgelassene Zeichen oder nicht korrekte Anwendung von Groß- und Kleinschreibung enthalten können.

- 1:n Mappings: Innerhalb des Frameworks soll es auch Ontologiepaare geben, in denen man 1:n Korrespondenzen finden kann, was eine logische Folge der Forderung von verschiedenen strukturierten Ontologien ist.

4 Ideen zur Umsetzung

Im vorherigen Kapitel wurden die allgemeinen und die speziellen Anforderungen an ein Framework zur Evaluation von (instanzbasierten) Matching-Sytemen oder -Algorithmen definiert. In diesem Kapitel sollen einige Umsetzungsmöglichkeiten skizziert und deren Vor-und Nachteile diskutiert werden.

Allgemeines Ziel ist die Erstellung eines Benchmarks, dass eine große Menge an Ontologien enthält. Ähnlich zu dem Benchmark der OAEI soll es eine Referenz-Ontologie geben, die sich auf eine bestimmte Domäne beschränkt. Die Wahl der Domäne hängt von der technischen Umsetzung ab, auf die später näher eingegangen wird. Desweiteren sollen die im vorherigen Kapitel und die von der OAEI beschriebenen Modifikationen der Ontologien (unterschiedliche Struktur/Formatierung etc.) umgesetzt und auch in verschiedenen Kombinationen zusammengesetzt werden. Zusätzlich ist es sinnvoll, die Referenzontologie in einigen Testszenarien auch um einige verwandte Konzepte/Themenbereiche zu erweitern (was auch eher der realen Welt entspricht), so dass das Verhalten der Systeme in diesen Fällen auch bewertet werden kann. Im Allgemeinen ist die Organisation der Testszenarien und die Erstellung unterschiedlicher Ontologien nicht sehr schwierig, wenn man erst einmal eine geeignete Referenzontologie hat.

Die Erstellung einer Referenzontologie ist differenzierter zu betrachten. Wir möchten möglichst realistische Ontologien für unsere Framework verwenden, so dass wir keine künstlichen Datengeneratoren und ausgedachte Ontologien verwenden, sondern auf Webinhalte zurückgreifen wollen. Grundlage dazu bilden ein Webcrawler und ein Parser, die geeignete Webseiten untersuchen und die passenden Inhalte extrahieren. Eine Möglichkeit wäre z.B. die Extraktion der Daten von der DBLP-Website. Die dort angebotenen Informationen sind in einer festen Struktur präsentiert, so dass man relativ leicht eine Ontologie extrahieren kann (manuell oder ggf. automatisch). Die Extraktion der Daten kann in jedem Fall automatisch durchgeführt werden. Ein Vorteil einer so erzeugten Ontologie ist, dass die Daten real und durch die Struktur leicht zu extrahieren sind. Allerdings wird die Bibliographie-Domäne schon von der OAEI genutzt, so dass ein anderes Gebiet für die Evaluation vielleicht etwas aufschlussreicher wäre. Eine andere Möglichkeit wäre die Nutzung der Wikipedia-Seiten (siehe auch [WWA+]). Insbesondere auf den englischen Seiten steht bei zahlreiche Themen eine Infobox zur Verfügung, die eine klare Struktur hat (zu finden auf http://de.wikipedia.org/wiki/Kategorie:Vorlage:Infobox). Diese Struktur könnte manuell oder automatisch in eine Ontologie transformiert werden, die Instanzen könnten automatisch von der entsprechenden Seiten extrahiert und den Konzepten zugeordnet werden. Interessant wäre es auch Ontologien zu einem Thema von verschiedensprachigen Seiten zu extrahieren und zu matchen. Passende Links auf den Seiten kann man auch als Relationen zu anderen Konzepten auffassen. Generell kommen verschiedenartige Websites in Frage, deren Eignung durch genauere Untersuchung und verschiedene Tests festgestellt werden muss.

5 Zusammenfassung und Ausblick

Das Matching von Ontologien ist ein weit verbreitetes Problem für das schon einige Lösungen existieren. Um die verschiedenen Ansätze sinnvoll vergleichen oder auch verbessern zu können ist es wichtig, dass geeignete Frameworks zur Evaluation entwickelt werden. Einige solcher Evaluationsinitiativen existieren schon, aber diese enthalten nur sehr wenige Instanzen und sind deswegen für instanzbasierte Matcher nicht sehr geeignet. Es wurden einige Anforderungen an das zu entwickelte Evaluations-Framework definiert. Die Umsetzung soll hauptsächlich mit Hilfe eines Webcrawlers und eines Parsers erfolgen, die automatisch Ontologien erzeugen bzw. Instanzen für einen vorgegebene Ontologie extrahieren. In naher Zukunft sollen die Erzeugung der Ontologien anhand verschiedener Webseiten getestet werden. Diese Referenzontologien müssen modifiziert werden, so dass ein Benchmark von verschiedenen Ontologien entsteht. Nachdem die erforderlichen Alignments definiert worden sind, sollen die Tests mit frei verfügbaren Systeme und unseren selbst entwickelten Matchern durchgeführt werden. Sobald die Entwicklung des Frameworks abgeschlossen ist, soll es der Allgemeinheit zur Verfügung gestellt werden.

Literatur

[ATV08] Bogdan Alexe, Wang-Chiew Tan, and Yannis Velegrakis. STBenchmark: Towards a benchmark for mapping systems. *Proc. VLDB Endow.*, 1(1):230–244, 2008.

[BM02] Jacob Berlin and Amihai Motro. Database Schema Matching Using Machine Learning with Feature Selection. In *Advanced Information Systems Engineering, 14th International Conference, CAiSE 2002, Toronto, Canada, May 27-31, 2002, Proceedings*, pages 452–466, 2002.

[DMDH04] AnHai Doan, Jayant Madhavan, Pedro Domingos, and Alon Y. Halevy. Ontology Matching: A Machine Learning Approach. In *Handbook on Ontologies*, pages 385–404. Springer, 2004.

[DR02] Hong Hai Do and Erhard Rahm. Coma - a system for flexible combination of schema matching approaches. In *VLDB 2002, Proceedings of 28th International Conference on Very Large Data Bases, August 20-23, 2002, Hong Kong, China*, pages 610–621, 2002.

[EM07] Daniel Engmann and Sabine Maßmann. Instance Matching with COMA++. In *Datenbanksysteme in Business, Technologie und Web (BTW 2007), Workshop Proceedings, 5.-6. März 2007, Aachen, Germany*, 2007.

[ES04] Marc Ehrig and Steffen Staab. QOM - Quick Ontology Mapping. In *INFORMATIK 2004 - Informatik verbindet, Band 1, Beiträge der 34. Jahrestagung der Gesellschaft für Informatik e.V. (GI), Ulm, 20.-24. September 2004*, pages 356–361. GI, 2004.

[ES07] Jérôme Euzenat and Pavel Shvaiko. *Ontology matching*. Springer-Verlag, Heidelberg (DE), 2007.

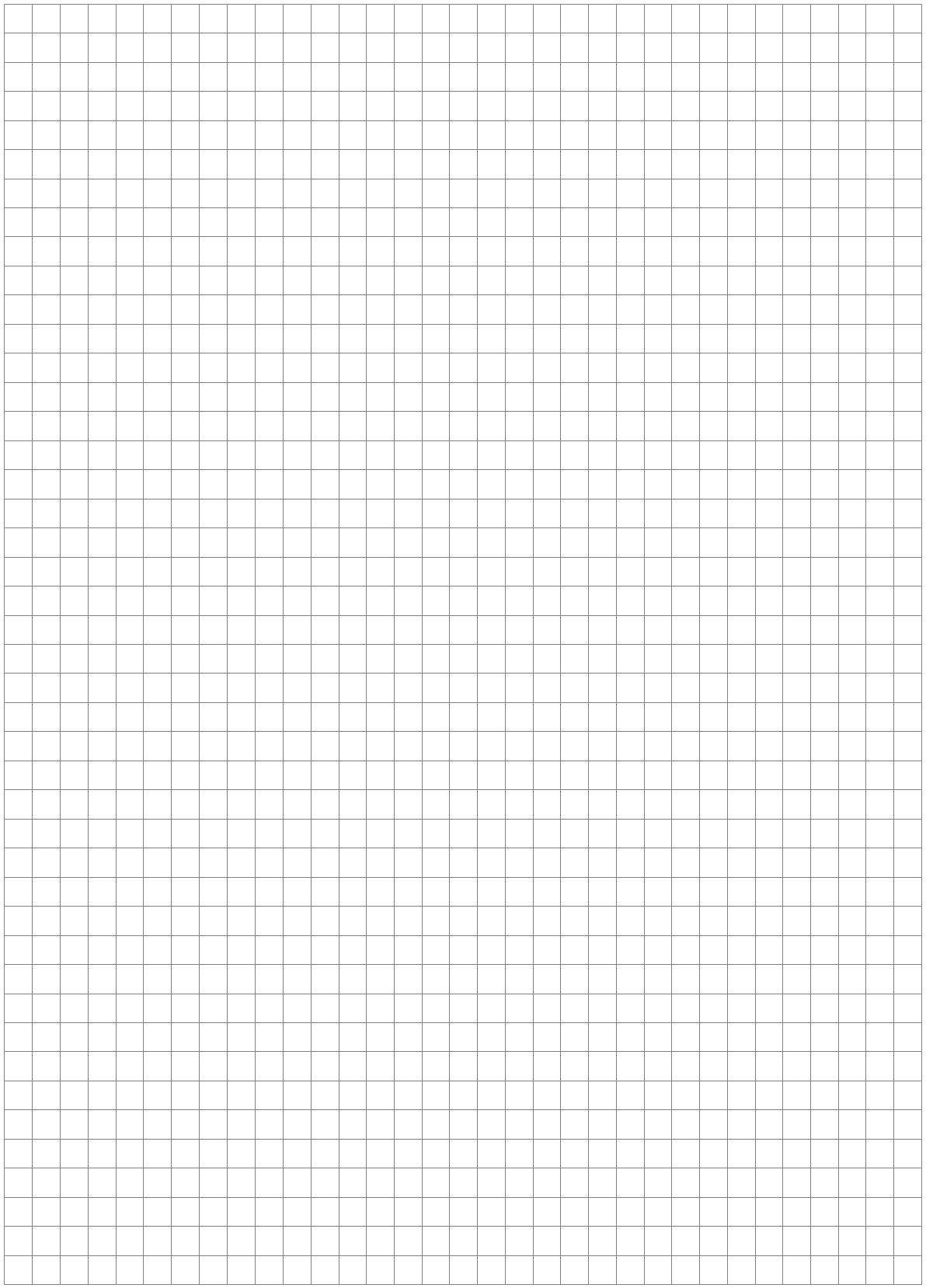
[Oae08] http://oaei.ontologymatching.org/2008/, 2008.

[RB01] Erhard Rahm and Philip A. Bernstein. A survey of approaches to automatic schema matching. *VLDB J.*, 10(4):334–350, 2001.

[SM01] Gerd Stumme and Alexander Maedche. FCA-MERGE: Bottom-Up Merging of Ontologies. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, IJCAI 2001, Seattle, Washington, USA, August 4-10, 2001*, pages 225–234, 2001.

[WWA+] Daniel S. Weld, Fei Wu, Eytan Adar, Saleema Amershi, James Fogarty, Raphael Hoffmann, Kayur Patel, and Michael Skinner. Intelligence in wikipedia. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008, Chicago, Illinois, USA, July 13-17, 2008*, pages 1609–1614.

[ZSC08] Katrin Zaiß, Tim Schlüter, and Stefan Conrad. Instance-Based Ontology Matching using Regular Expressions. In R. Meersman, Z. Tari, and P. Herrero, editors, *On the Move to Meaningful Internet Systems: OTM 2008 Workshops, ODBase 2008, LNCS 5333, 9-14. November 2008, Monterrey, Mexico*, pages 40–41. Springer-Verlag, 2008.



Ontologiespeicherung in Datenbanken im Kontext natürlichsprachlicher Dialogsysteme

Nils Weber
Christoph Eigenstetter
Markus Berg
Antje Düsterhöft

Hochschule Wismar
University of Applied Sciences
Technology, Business and Design

In vielen Anwendungen ist es notwendig komplexe Sachverhalte vorzuhalten um Eigenschaften, Zustände und Beziehungen zwischen den Objekten einer Domäne herzustellen und weiterführende Informationen abzuleiten. Hierfür eigenen sich besonders gut Ontologien und die Konzepte des Reasoning. Im Umfeld natürlichsprachlicher Dialogsysteme ist es von entscheidender Bedeutung möglichst umfassend Kontextinformationen und deren Zusammenwirken strukturiert zu verwalten, um daraus Informationen für die Dialoggenerierung zu gewinnen[9]. Der praktische Einsatz von Ontologien zeigte erhebliche Leistungsunterschiede zwischen den existierenden Knowlegde Base Systemen. Dieser Artikel stellt die Vorteile und Nachteile der wichtigsten Softwarelösungen gegenüber und gibt einen Überblick über die charakteristischen Eigenschaften der konkurierenden Produkte.

1 Einführung

Das „Natural Language Laboratory“[1] der Hochschule Wismar beschäftigt sich mit der computergestützten Verarbeitung von menschlicher Sprache. Dazu gehören u.a. Themengebiete wie Spracherkennung, Sprachsynthese, die semantische Interpretation von sprachlichen Äußerungen, das Führen von natürlichsprachlichen Dialogen sowie das Design von IVR-Systemen[2].
Der Schwerpunkt der aktuellen Forschungsprojekte liegt auf der Unterstützung des Anwenders mit Hilfe von flexiblen natürlichsprachlichen Dialogen in den verschiedensten Domänen. Um einen Sprachdialog zu führen, ist es notwendig eine Vielzahl unterschiedlicher Prozesse zu beherrschen, u.a. die semantische Interpretation von sprachlichen Äußerungen. Für diesen Zweck bieten Ontologien eine hervorragende Basis, da sie die Annotation von gespeicherten Daten mit Semantik unterstützen und durch logisches Folgern (Reasoning) die automatische Extraktion von implizit vorhandenen Daten ermöglichen.
Eines unserer aktuellen Forschungsprojekte, „TravelConsult im Dialog“[3], enthält Szenarien aus dem Bereich Tourismusmarketing. Ziel ist die Entwicklung einer zeitgemäßen Nutzerschnittstelle (natürlichsprachl. Dialog), die den Reisenden beim Auffinden und bei der Buchung von touristischen Angeboten komfortabel unterstützt. Da die Domäne „Tourismus“ im Vergleich zu anderen beschränkteren Szenarien vergleichsweise umfangreich ist und somit die zu verarbeitenden Ontologien ebenfalls eine hohe Komplexität erreichen, wurde es für uns notwendig, geeignete Speicherungsmöglichkeiten für große Ontologien zu finden, die den Projektanforderungen, besonders im Hinblick auf die Performance, genügen.

[1] http://www.et.hs-wismar.de/natlab
[2] IVR - **I**nteractive **V**oice **R**esponse Systems
[3] http://www.et.hs-wismar.de/natlab/travelconsult.html

Im Verlauf der Recherche wurden verschiedene Frameworks zur Speicherung von umfangreichen OWL-Ontologien getestet. Nahezu alle getesteten Lösungen benutzen im Hintergrund ein RDBMS für die eigentliche Speicherung und Indizierung der OWL-Daten. Zusätzlich müssen geeignete Anfragemechanismen, wie bswp. SPARQL[4], und adäquate Indizierungsmechanismen vorhanden sein. Zu den betrachteten Lösungen zählen: (a) Oracle 11g Semantic Technologies (b) OWLGres (c) Sesame (d) OWLIM (e) Jena.

2 Technologien

Technologien zur Speicherung und Verarbeitung von Semantik wurden in den letzten Jahren weiter entwickelt und werden immer ausgereifter. Dadurch werden sie in immer breiteren Anwendungsfeldern eingesetzt und die gespeicherten Ontologien und RDF-Graphen wachsen signifikant. Aus diesem Grund werden effiziente und skalierbare Technologien benötigt, die große Datenmengen verwalten können und nicht an die Größe des verfügbaren Arbeitsspeichers gebunden sind.
Eine Idee ist, bestehende relationale Datenbanksysteme zu nutzen, die ja bereits viele der oben genannten Anforderungen erfüllen, und diese um spezielle Funktionen für die Verarbeitung von Ontologien zu erweitern. Fast alle der in diesem Artikel vorgestellten Lösungen benutzen daher ein DBMS[5] entweder als reine Persistenzschicht, auf die über ein entsprechendes API[6] zugegriffen werden kann, oder die Funktionalität des API wird nativ in das DBMS integriert.
Ein weiterer Vorteil von Ontologien ist das automatische Folgern von implizitem Wissen mit Hilfe von Inferenzregeln. Das Reasoning wird bei vielen Lösungen direkt im RAM durchgeführt, so daß hier der Umfang des verwendeten Graphen durch die physikalische Größe des Arbeitsspeichers begrenzt ist. Gerade bei komplexen Ontologien, in der Größenordnung von mehreren Terrabytes, ist dies nicht akzeptabel. Zusätzlich gewinnt hier die Verarbeitungsgeschwindigkeit beim Laden, Anfragen und Schlussfolgern an Bedeutung.

2.1 Oracle 11g

Die Datenbank Oracle 11g stellt optional eine native RDF/RDFS/OWL-Unterstützung, die auf dem Oracle Spatial Network Data Model [2] basiert, zur Verfügung. Semantische Daten werden intern als RDF-Tripel auf das Network Data Model abgebildet. Dadurch wird das Laden, das Anfragen und das Inferencing von umfangreichen Graphen beschleunigt. Die Datenbank enthält eine native Inference-Engine, die OWL-DL, RDF, RDFS und benutzerdefinierte Regeln unterstützt [3]. Mit Hilfe von *Rulebases* können benutzerdefinierte Inferenzregeln definiert werden.
Semantische Daten werden mit Hilfe der *SEM_MATCH*-Funktion in einer SPARQL-ähnlichen Syntax abgefragt. SEM_MATCH kann mit regulären SQL-Statements kombiniert werden, so daß sich hier umfangreiche Möglichkeiten für die kombinierte Auswertung von semantischen und relationalen Daten ergeben (s. Abb. 1). Listing 1 findet bspw. Hotelnamen und Hotelort, die in Westmecklenburg liegen.

Listing 1: SPARQL-SQL Beispiel

```
SELECT x, h.ort
FROM TABLE(SEM_MATCH('(?x␣:in␣:Westmecklenburg)',
SEM_Models('owl'),
SEM_RULEBASES('OWLPRIME'),
SEM_ALIASES(SEM_ALIAS('','http://et.hs-wismar.de/natlab/owltest#')),null)),
Hotel h
WHERE h.name = x;
```

[4]SPARQL Protocol and RDF Query Language
[5]DBMS - **D**ata**b**ase **M**anagement **S**ystem
[6]API - **A**pplication **P**rograming **I**nterface

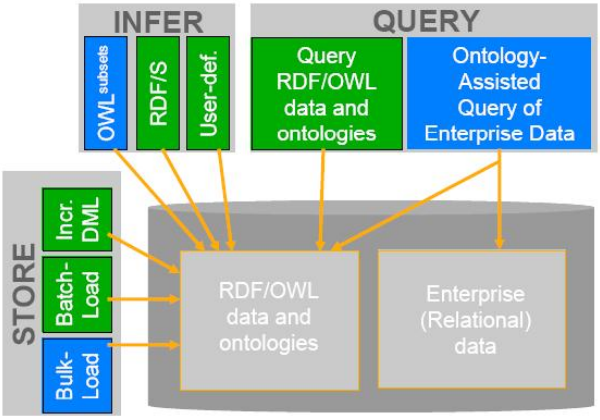


Abbildung 1: Oracle Database 11g RDF/OWL Semantic Data Store [1]

Es existieren Schnittstellen zu gängigen semantischen Design- und Analysewerkzeugen, wie TopBraid Composer[7] oder Protégé[8]. Einen weiteren Vorteil gerade beim Umgang mit umfangreichen Datenbeständen stellt die ausgereifte und skalierbare Oracle-Architektur dar, welche über das Oracle Network Data Model mitgenutzt wird.

2.2 OWLGres

OWLGres ist eine DL-Lite Reasoner Implementation, basierend auf dem DBMS PostgreSQL. Ziele von OWLGres sind das effiziente Abfragen einer skalierbaren, persistenten semantischen Datenbasis und das effiziente automatische Reasoning auf grossen RDF- und OWL-Datenbeständen. Derzeit werden Teile des OWL-DL Standards unterstützt. OWLGres ist als Java API implementiert und nutzt das DBMS PostgreSQL als Persistenzschicht.
Besonderen Wert wurde auf die Optimierung der Anfrage- und Reasoningperformance gelegt. Verbesserungen werden mit Hilfe von Query Simplification, Selectivity Optimization und der Zusammenfassung von mehreren SQL-SELECT-Statements mittels des UNION-Operators erreicht [4]. Für die Abfrage des Graphen wird SPARQL genutzt, wobei jedoch einige SPARQL-Konstrukte, wie bspw. OPTIONAL nicht unterstützt werden. Die Dokumentation ist noch lückenhaft, da sich das Projekt in einem frühen Stadium der Entwicklung befindet.

2.3 Sesame

Sesame[9] ist ein plattformunabhängiges Open Source Framework zum Speichern von RDF-Daten mit Unterstützung zur Anfrage und Inferencing von RDF Schema. Es kann im Zusammenspiel mit verschiedenen Speicherungskonzepten (Datenbanken, in-memory, Dateisystemen, Keyword Indexer) eingesetzt werden und stellt eine Vielzahl von Entwicklerwerkzeugen bereit. Verschiedene Anfragesprachen wie SeRQL und SPARQL sind über eine flexible API nutzbar.
Ein zentrales Konzept bei Sesame sind Repositories. Der Repostitory-Typ bestimmt die Art der Datenhaltung und hat Einfluss auf den Umfang des Inferencing. Über den sogenannten SAIL-Stack[10] werden alle relevanten Parameter gesetzt. Ohne Erweiterungen sind allerdings nur grundlegende Schlussfolgerungen möglich, die sich aus der Hierarchie ableiten lassen. Mit Hilfe des *Custom Inferencing* wird dem Nutzer eine Möglichkeit gegeben eigene transitive, symetrische oder inverse Properties zu definieren. Hierzu ist es notwendig den SAIL-Stack anzupassen und

[7] TopBraid Composer - `http://www.topquadrant.com`
[8] Protégé - `http://protege.stanford.edu/`
[9] Sesame - an Open Source RDF Framework, `http://www.openrdf.org/index.jsp`
[10] SAIL - **S**torage **A**nd **I**nference **L**ayer

ein gegeinetes Repository zu erstellen. Am Beispiel OWLIM soll der Zusammenhang zwischen Repositorytyp und Reasoning verdeutlicht werden.

2.4 OWLIM

OWLIM ist ein skalierbares semantisches Repository zur Speicherung, Integration und Analyse heterogener Daten. Dabei werden *Full RDFS*, eingeschränktes *OWL Lite* und *Horst* unterstützt. OWLIM ist als *Storage And Inference Layer* (SAIL) für Sesame implementiert und basiert auf der TREE-Engine[11]. Es werden die zwei Ausprägungen - *SwiftOWLIM* mit extrem schnellen Lade- und Inferenzzeiten und *BigOWLIM* für die Optimierung sehr großer Datenmengen unterschieden.
Bei der kommerziellen Version *BigOWLIM* ist die Skalierbarkeit das wesentlichen Kriterium. Hier werden die Daten nicht komplett im Speicher gehalten, sondern in Binärfiles abgelegt. Dieses Konzept bildet die Grundlage für das Handling von sehr großen Datenmengen. Anfrageoptimierung und spezielle Kriterien bei der Nutzung von *Equivalenzklassen* sollen verhindern, dass unnötige Inferenzen gebildet werden.
SwiftOWLIM dagegen setzt als in-memory-Version auf maximale Performanz. Basierend auf Hash-Table-Indizes werden Strategien zur Optimierung der Speicherung, Indizierung, Konsistenz und Integrität der Daten zur Verfügung gestellt. SwiftOWLIM gilt als schnellste RDF(S)- und OWL-Engine.

2.5 Jena

Jena[12] ist ein Open-Source Java Framework für die Entwicklung von semantischen Applikationen. Es stellt RDF und OWL APIs sowie eine SPARQL-Query-Engine zur Verfügung. Ursprünglich wurde Jena im Rahmen des „HP Labs Semantic Web Programme“[13] entwickelt. Es wird sowohl die in-memory-Speicherung von OWL Modellen als auch die persistente Speicherung in relationalen Datenbanken über das „Jena2 persistence subsystem“ (JPS) [7] unterstützt. JPS enthält einen Fastpath[14]-Algorithmus, welcher für Teile von SPARQL-Anfragen dynamisch SQL-Statements generiert, die dann direkt in der Datenbank bearbeitet werden. JPS unterstützt derzeit folgende DBMS: HSQLDB[15], MySQL, PostgreSQL, Apache Derby, Oracle und MS SQL-Server.
ARQ ist eine Implementierung der SPARQL-Sprache für das Jena Framework. Es werden die gängigen SPARQL-Funktionalitäten abgebildet und einige zusätzliche Features, wie „GROUP BY“-Klauseln, property paths, LET Variablen und Sub-Selects integriert [10].

3 Vergleich

Derzeit steht eine beachtliche Anzahl von Systemen zur Speicherung und zur Verarbeitung von semantischen Datenbeständen zur Verfügung. Die meisten unterscheiden sich in wesentlichen Punkten, so daß ein jedes System verschiedene Charakteristika aufweist und somit für unterschiedliche Anforderungen geeignet ist. Aufgrund der stetig wachsenden semantischen Datenmengen spielt die Performance und Stabilität bei der Verarbeitung von umfangreichen Graphen zunehmend eine besondere Rolle. Daher wurden Methoden entwickelt, um die Performance, Skalierbarkeit und Stabilität von semantikverarbeitenden Systemen standardisiert zu messen. Der bekannteste dieser sogenannten Benchmarks ist LUBM [8], welcher an der Lehigh University in Bethlehem, USA, entwickelt wurde.

[11] TREE - **T**riple **R**easoning and **R**ule **E**ntailment
[12] `http://jena.sourceforge.net`
[13] HP Labs Semantic Web Programme - `http://www.hpl.hp.com/semweb/`
[14] Jena Fastpath Query Processing - `http://jena.sourceforge.net/DB/fastpath.html`
[15] HSQLDB - **H**yper**SQL** **D**ata**b**ase

Tabelle 1: LUBM Loading (Time in hours)

Software	**LUBM 50**	**LUBM 500**	**LUBM 1000**	**LUBM 50k**
Oracle 11g	0:13:29	3:20:00	6:23:00	k.A.
OWLGres	k.A.	k.A.	k.A.	k.A.
OWLIM	0:02:00 (Swift)	03:00:00	k.A.	40:00:00 (BigOWLIM)
Sesame (Native Store)	0:11:15	3:22:00	k.A.	k.A.
Jena	k.A.	k.A.	k.A.	k.A.

LUBM stellt eine Ontologiestruktur, einen Datengenerator, 14 standardisierte SPARQL-Anfragen sowie verschiedene Messparameter, wie Ladezeit, Repositorygröße und Antwortzeiten, bereit.
Mit Hilfe des Datengenerators können vergleichbare Testdaten generiert werden. Dabei wird auf eine festgelegte OWL-Struktur zurückgegriffen, die eine Universitätsdomäne repräsentiert. Die Testdaten enthalten dann 1 .. n Universitäten (LUBM n), wobei jede Universität 15 - 25 departments mit den dazugehörigen Daten (Studenten, Vorlesungen, Mitarbeiter etc.) enthält. Somit ergibt sich bspw. für eine LUBM 50 Ontologie die Anzahl von 6,8 Mio. Tripeln.
Mit LUBM können die Zeit, die für das Laden einer Ontologie in das System benötigt wird sowie die physische Größe des belegten Speicherplatzes gemessen werden. Um die Anfrageeffizienz eines Knowledge Base Systems zu bestimmen, wurden 14 genormte SPARQL-Anfragen entwickelt, die sich hinsichtlich ihrer Selektivität, Komplexität und der Hierarchiekomplexität unterscheiden. Außerdem existiert für jede Query eine definierte Antwortmenge, anhand derer sich die Vollständigkeit einer Systemantwort und somit auch die Reasoningqualität ermitteln läßt.
Zum jetzigen Zeitpunkt existieren nur sporadische Angaben der Entwickler/Hersteller in Bezug auf LUBM-Ergebnisse ihrer Produkte. Die in der Tabelle 1 aufgeführten Werte sind somit nicht vollständig und nicht direkt vergleichbar und sollen lediglich als grober Anhaltspunkt für die Leistungsfähigkeit der Systeme dienen. Sie wurden im Rahmen unserer Recherche gewonnen. Tabelle 1 enthält die Ladezeiten für LUBM Ontologien verschiedener Grösse.

4 Zusammenfassung

Die in diesem Dokument beschriebenen Techniken geben einen Überblick über die wichtigsten Speicherungsmöglickeiten von Ontologien und über die Vorteile und Nachteile der jeweiligen Werkzeuge im Anforderungsksontext. Im Bereich der Datenhaltung unterscheidet man grundsätzlich Systeme, bei denen die Daten komplett im Hauptspeicher gehalten werden und Systeme mit zusätzlichen Datenbanken oder Binärdateien. Ein weiteres Schlüsselkriterium zur Bewertung von Knowlege Base Systemen ist der Umfang des Reasonings, also die Fähigkeit implizite Informationen aus den ursprünglichen Daten abzuleiten und nutzbar zu machen.
In Abhängigkeit von den Anforderungen der jeweiligen Anwendungen müssen sich die Entwickler bei der Wahl des Knowledge Base Systems zwischen möglichst guter Performanz oder der Fähigkeit, mit großen Datenmengen umzugehen, entscheiden. Gerade der Umgang mit großen Datenmengen kann unter Umständen dazu führen, dass einige Systeme bei denen die Daten im Speicher gehalten werden, abhängig von der zu Verfügung stehenden Hardware nicht nuztbar sind. Das Hochladen besonders umfangreicher Ontologien und Anfragen auf den Datenbestand dauern gegebenfalls extrem lange oder führen sogar zu Abstürzen der Systeme.
Auf der anderen Seite stehen Anforderungen an die Reaktionsgeschwindigkeit der Systeme, wobei hier in-memory Systeme wesentlich effektiver arbeiten. Einschränkungen in der Wahl der geeigneten Systeme sind im Umfang der Reasoningfähigkeiten zu suchen.
Über die Anforderungen an die Leistungsfähigkeit hinaus können auch die Implementierungssprachen der API's die Entscheidung für ein System beeinflussen. So sind die meisten der Sy-

steme als Javabibliotheken erhältlich. Mit *dotsesame*[16] existiert ein Versuch der Imlementierung einer C#-Schnittstelle, die jedoch zum gegenwärtigen Zeitpunkt nur über die Nutzung der Java-Klassen aus der C#-Umgebung heraus möglich ist.
Ein weiterer Aspekt bei der Integration von Knowledge Bases in komplexe Anwendungen ist die Stabilität unter realen Bedingungen. Hierbei sind beispielsweise Multiuserumgebungen mit Parallelzugriffen unter Umständen problematisch, da diese zu Inkonsistenzen innerhalb des Datenbestands führen können. Umfangreiche praktische Erfahrungen und Testergebnisse liegen diesbezüglich allerdings nur für das Jena-Framework vor und sind somit nicht nicht auf alle Systeme zu beziehen.
Abschließend bleibt noch der Kostenfaktor zu erwähnen, welcher bei kommerziellen Systemen wie Oracle nicht unerheblich ist. Das für den Umgang mit sehr großen Datenmengen optimierte BigOWLIM ist ebenfalls nur in einer Testversion kostenlos, stellt aber als einziges System Konzepte für das Reasoning von über 3.3 Milliarden Statements und den Upload von bis zu 6.7 Milliarden Triples zur Verfügung.

Literatur

[1] Semantic Data Integration for the Enterprise. Oracle Corporation, 2007

[2] Oracle Spatial Network Data Model. Oracle Corporation, May 2005

[3] Zhe Wu, George Eadon, Souripriya Das, Eugene Inseok Chong, Vladimir Kolovski, Melliyal Annamalai, Jagannathan Srinivasan: Implementing an Inference Engine for RDFS/OWL Constructs and User-Defined Rules in Oracle. In Proc. of 24th International Conference on Data Engineering (ICDE), April 7-12, 2008, Cancun, Mexico

[4] Markus Stocker, Mike Smith: Owlgres: A Scalable OWL Reasoner. In Proc. of OWLED2008. Fifth International Workshop, Karlsruhe, Germany, 2008

[5] Atanas Kiryakov, Damyan Ognyanov, Dimitar Manov: OWLIM a Pragmatic Semantic Repository for OWL. In Proc. of Int. Workshop on Scalable Semantic Web Knowledge Base Systems (SSWS 2005), WISE 2005, 20 Nov, New York City, USA.

[6] Brian McBride: Jena Implementing the RDF Model and Syntax Specification. In Proc. of The Second International Workshop on the Semantic Web (SemWeb'2001), Hongkong, May 1, 2001

[7] Wilkinson, Kevin; Sayers, Craig; Kuno, Harumi; Reynolds, Dave: Efficient RDF Storage and Retrieval in Jena2. HPL-2003-266

[8] Y. Guo, Z. Pan, and J. Heflin. LUBM: A Benchmark for OWL Knowledge Base Systems. In Journal of Web Semantics, Vol 3, Issue 2, 2005

[9] N. Weber, C. Eigenstetter, M. Berg, A. Düsterhöft: Natural language dialogs in home automation systems (eingereicht NLDB 2009)

[10] ARQ A SPARQL Processor for Jena. `http://jena.sourceforge.net/ARQ/`

[16] dotsesame-Projekt `http://sourceforge.net/projects/dotsesame`

Eine Link-Datenbank zur Integration von Virtual Engineering-Daten*

Ingolf Geist und Stephan Vornholt
Otto-von-Guericke-Universität Magdeburg
Universitätsplatz 2, 39106 Magdeburg
{ingolf.geist|stephan.vornholt}@ovgu.de

Zusammenfassung

Virtual Engineering beschreibt den kollaborativen Prozess der Produktentwicklung in den Phasen der Konstruktion und Simulation/Analyse mit Hilfe von computergestützten Methoden in unterschiedlichen Entwicklungsabteilungen. Dabei entsteht in verschiedenen Entwicklungsbereichen eine Vielzahl von heterogenen, komplex-strukturierten Dokumenten, Produktdaten und Wissen, welche voneinander abhängen. Da Produktmodelle zur Integration von Konstruktion und Verhaltensanalyse die Sichten der Entwickler in den verschiedenen Bereichen einschränken, soll in dieser Arbeit eine Link-Datenbank vorgestellt werden, die durch externe Links und Annotationen Abhängigkeiten verwaltet, ohne die lokalen Datenmodelle zu verändern. Die Integration erfolgt, ähnlich zu Social-Tagging-Systemen, durch die interaktive Annotation der Daten in ihrer jeweiligen Entwicklungsumgebung und der Erstellung von Abhängigkeiten zwischen den Bereichen.

1 Einleitung

Im globalen Wettbewerb ist es notwendig, die Entwicklungszyklen von Produkten zu verkürzen, um schneller auf den Markt reagieren und neue Produkte kostengünstig anbieten zu können. Die konsequente und durchgehende Nutzung von computergestützten Methoden und virtuellen Prototyping-Techniken [9] in den Phasen Konstruktion und Simulation/Analyse des Produktlebenszyklus [6] ist eine Möglichkeit diese Anforderung zu erfüllen. Die Entwicklungsprozesse, *Virtual Engineering* (VE) genannt, nutzen CAD-Systeme zur Konstruktion sowie verschiedenste Simulationswerkzeuge (CAE-Systeme) zur frühen Analyse des Verhaltens eines Produktes. Eine effiziente Unterstützung erfordert eine Integration von CAD- und CAE-Daten.

Abb. 1 skizziert drei Datenquellen und verschiedene Aktionen in einem VE-Prozess zur Entwicklung eines Industrieroboters. Ausgehend von dem Ergebnis des konzeptuellen Entwurfs (nicht gezeigt) wird in einem CAD-System der Roboter konstruiert. Um zu überprüfen, ob das initiale Design die gegebenen Anforderungen erfüllt, wird ein Simulationsmodell mit Hilfe der Informationen aus dem CAD-Modell erstellt und parametrisiert. Dieser Schritt erfolgt manuell oder automatisch durch angepasste Werkzeuge (z.B. [3]). Mit Hilfe zusätzlicher Produktdaten über Elektromotoren und Materialien wird das Simulationsmodell vervollständigt. Anschließend werden verschiedene Alternativen getestet. Die Ergebnisse der Tests müssen an die Konstrukteure zurückgegeben werden, um eventuelle Änderungen ausführen zu können. Diese Schritte werden zyklisch durchgeführt bis das Produktdesign den Anforderungen entspricht. Aus der Analyse verschiedener VE-Prozesse ergeben sich folgende Anforderungen für die Datenverwaltung:

Integration: In den verschiedenen Entwicklungsbereichen, *Domänen* genannt, liegen Daten über dieselben Artefakte, d.h. Real-World-Objekte, aus unterschiedlichen Blickwinkeln vor. Die

*Unterstützt durch die Europäische Kommission: EFRE COMO C1-3201201 und C3-3201201.

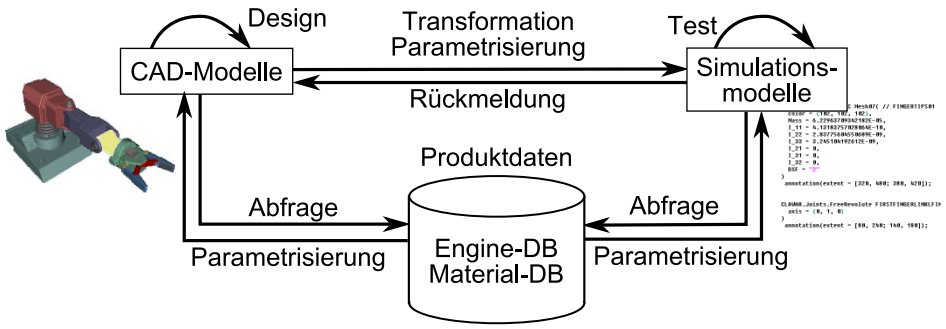


Abbildung 1: Beispielszenario

dabei auftretende Redundanz von Information muss verhindert oder verwaltet werden. Eine Möglichkeit ist die Nutzung von integrierten Produktmodellen, in denen versucht wird CAD- und CAE-Daten zu integrieren (eine Übersicht ist in [5] gegeben).

Unterstützung von Entwicklersichten: Jeder Ingenieur hat seine vertraute Entwicklungsumgebung, die eine bestimmte Datenverwaltung sowie ein bestimmtes Datenmodell und -schema voraussetzt, woraus sich eine domänen- und entwicklerspezifische Sicht auf die Produktentwicklung ergibt. Diese unterschiedlichen Sichten werden von integrierten Produktmodellen nur eingeschränkt unterstützt. Deswegen wurden *lightweight* Integrationsmodelle vorgeschlagen, welche die Daten in den lokalen Datenmodellen belassen, und nur die Abhängigkeiten verwalten [1].

Semantik/Beschreibung: Es ist notwendig die Ideen und Konzepte hinter einer Entwicklung dauerhaft zu speichern, um spätere Projekte von diesem Wissen durch bessere Dokumentation und Suchmöglichkeiten profitieren zu lassen. Diese Informationen sollen idealerweise direkt zu dem VE-Datenobjekt zugeordnet sein, welches sie beschreiben. Die Beschreibungen ihrerseits sollen ebenfalls organisiert sein, bspw. durch eine Ontologie. Durch Beachtung von Entwicklersichten stellt der Aufbau von domänen- und projektpezifischen Ontologien und deren Integration einen großen Aufwand dar.

Soziale Interaktion: In einem VE-Prozess arbeiten verschiedene Teams und Personen zusammen, die miteinander interagieren müssen. Somit ist es von Interesse, Kommentare und das Wissen von Personen über Ressourcen innerhalb des VE-Prozesses auszutauschen und zu nutzen.

Um diese Anforderungen zu erfüllen, wird in dieser Arbeit eine externe Link-Datenbank *LinkDB* vorgeschlagen, die nach dem Vorbild einer Folksonomy [7, 8] aufgebaut ist. Dabei stellen Datenobjekte und Beziehungen zwischen Datenobjekten die Menge der Ressourcen dar, die unabhängig von der Domäne definiert sind. Strukturierte Annotationen werden von Nutzern in einem Bottom-Up-Prozess angelegt und den Ressourcen zugeordnet. Desweiteren ermöglicht das System die explizite Definition von Abhängigkeiten zwischen Domänen, um die Integration zu ermöglichen. Im Folgenden wird der prinzipielle Aufbau des Systems vorgestellt, die Struktur der LinkDB beschrieben und der Integrationsprozess erläutert.

2 Architektur

Abb. 2 skizziert die Architektur des Systems und dessen Arbeitsweise. Die Entwickler benutzen domänenspezifische Werkzeuge, die ihre Daten in den jeweiligen Formaten in lokalen Repositories verwalten. Die Repositories haben die Aufgaben, eine lokale Versionsverwaltung anzubieten und den Zugriff auf Datenobjekte mittels eines Uniform Resource Identifiers über eine Webservice-Schnittstelle zu ermöglichen. Der *LinkDB-Server* implementiert die Integrations- und Annotationsfunktionalität. Alle dazu notwendigen Informationen verwaltet ein relationales Datenbanksystem. Darauf aufbauend wird der LinkDB-Server Dienste zur Registrierung und Überwachung von lokalen Datenquellen und zur Annotierung von Ressourcen anbieten. Der Server erlaubt

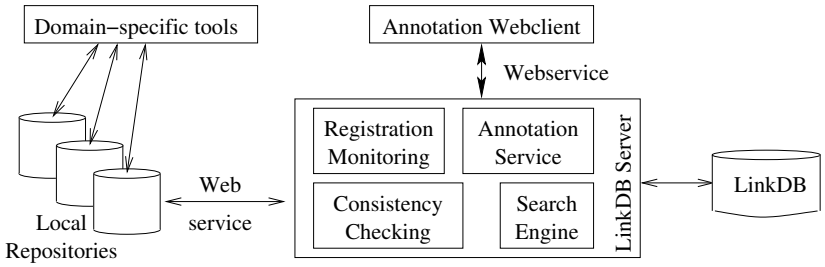


Abbildung 2: Architektur

die Suche nach Ressourcen und Experten und implementiert eine Abhängigkeitskontrolle zwischen Domänen. Diese Dienste sind über Webservice-Schnittstellen verfügbar und somit in anwendungsspezifische Werkzeuge integrierbar. Ein webbasierter Annotations-Client bietet einen domänenunabhängigen Zugriff auf die LinkDB-Dienste.

3 Link-Datenbank

Die Struktur der LinkDB folgt grundsätzlich der Struktur einer Folksonomy, d.h. es existieren Ressourcen, Nutzer, Annotationen und die Beziehungen zwischen diesen Mengen [8].

Werte in Annotationen können verschiedenen Datentypen angehören. Die *Datentypen* sind in der Menge $\mathcal{T} = String \cup Integer \cup Date \cup ...$ zusammengefasst. Darüber hinaus gibt es zwei spezielle Mengen: $\mathcal{URI} \subset String$ umfasst alle Uniform Resource Identifier (URI) und $Label \subset String$ beinhaltet alle gültigen Bezeichner.

Eine VE-Ressource repräsentiert ein Objekt in einem lokalen Daten-Repository, das durch eine URI eindeutig identifiziert ist. Die Koordinaten in einer 2D- oder 3D-Geometrie können in der URI kodiert werden, d.h. Ressourcen können auch mittels der Geometrie spezifiziert werden. Desweiteren wird der Ressource in der LinkDB ein Bezeichner zugeordnet. Somit wird die Menge der *VE-Ressourcen* als $\mathcal{R} = URI \times Label$ definiert. In der Beispiel-Datenbank in Abb. 3 werden Ressourcen aus drei Quellen (CAD-Modell, Mehrkörpersimulation (MBS), E-Motor-DB) als Boxen dargestellt. Auf die URIs wurde in der Darstellung verzichtet, aber jede Ressource stellt einen Link zu einem lokalen Objekt dar.

Ein Nutzer, z.B. $u2 = (user2, \text{Hans Müller})$ in Abb. 3, wird durch einen eindeutigen Nutzernamen sowie wie den realen Namen dargestellt. Reale Namen ermöglichen die direkte Kommunikation und werden deshalb in Enterprise-Folksonomies eingesetzt [4]. Im vorgestellten System ist die Menge der *VE-Nutzer* als $\mathcal{U} = Label \times String$ spezifiziert. Weitere Nutzerdaten wie Kontaktdaten können hinzugefügt werden.

VE-Ressourcen stehen in verschiedenen Beziehungen zueinander, die entweder domänenspezifisch sind (z.B. `partOf`-Beziehung in einem CAD-System), in allen Domänen vorkommen (z.B. `versionOf`-Beziehung) oder die Abhängigkeiten zwischen Domänen widerspiegeln. Weiterhin existieren Beziehungen zwischen Ressourcen und Nutzern. Die Menge der *VE-Beziehungen* ist als gerichtete Kantenmenge $\mathcal{E} = \mathcal{V} \times \mathcal{V} \times Label$ definiert, wobei $\mathcal{V} = \mathcal{U} \cup \mathcal{R}$ die Menge der Knoten ist. Jede Kante hat einen Bezeichner, wodurch mehrere Kanten pro Knotenpaar möglich sind. In Abb. 3 sind diese Beziehungen als durchgehende Linien dargestellt, auf die Bezeichner wurde in der Darstellung verzichtet.

Ressourcen und Beziehungen haben weder einen Typ noch eine Beschreibung. Diese Informationen sollen durch Annotationen geliefert werden. Eine Annotation ist eine bezeichnete Menge von Label-Wert-Paaren. Annotationen haben eine komplexe Struktur, sind aber wie in einem Social-Tagging-System frei durch den Nutzer definierbar. Die Menge aller Label-Wert-Paare ist als $LV = Label \times \mathcal{T}$ definiert und $\mathcal{P}(LV)$ bezeichnet die entsprechende Potenzmenge. Somit stellt die Menge $\mathcal{A} = Label \times \mathcal{P}(LV)$ die Menge der *Annotationen* dar. In Abb. 3 ist z.B. die Annotation $(manuell, \{(\text{description}, \text{manuelle Übernahme})\})$ dargestellt, welche eine Kante als

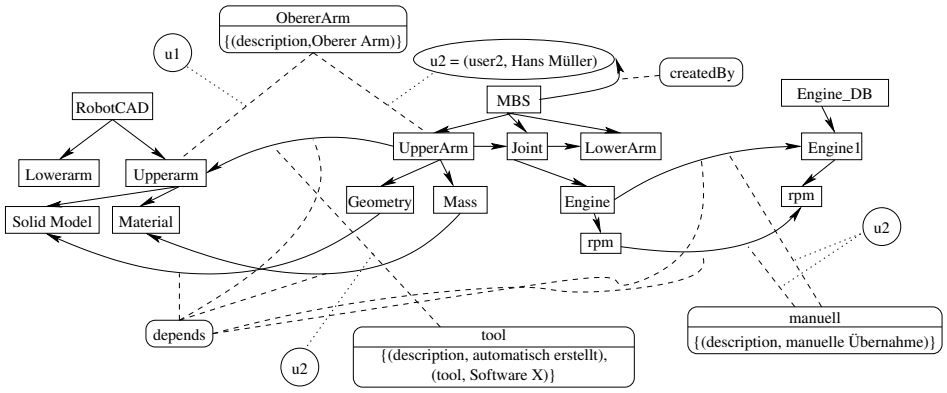


Abbildung 3: Beispiel-Datenbank

manuelle Übernahme von Parameterwerten beschreibt.

Der Zusammenhang zwischen Ressourcen $\mathcal{R}$ und Kanten $\mathcal{E}$, Nutzern $\mathcal{U}$ sowie Annotationen $\mathcal{A}$ wird durch Beziehung *Annotierung* ausgedrückt. Die Beziehung ist als Relation $\mathcal{AN} = (\mathcal{R} \cup \mathcal{E}) \times \mathcal{U} \times \mathcal{A} \times Date$ definiert. In Abb. 3 werden Annotierungen durch gestrichelte und gepunktete Linien dargestellt. Das Element $(UpperArm, u2, ObererArm, date) \in \mathcal{AN}$ bedeutet (siehe Abb. 3), dass der Nutzer *user2* der Ressource $UpperArm$ im Mehrkörpermodell (MBS) die Annotation $ObererArm$ zum Zeitpunkt *date* zugeordnet hat. Wie in der Abbildung zu sehen, kann eine Annotation verschiedenen Ressourcen zugeordnet werden. Der Zeitpunkt der Annotierung ist nötig, um den Status der Produktentwicklung mit der Annotation in Beziehung setzen zu können, welches gerade bei iterativen Produktentwicklungsprozessen wichtig ist [7]. Nachdem alle Komponenten beschrieben sind, ist die Link-Datenbank definiert als

$$LinkDB = (\mathcal{R}, \mathcal{U}, \mathcal{E}, \mathcal{A}, \mathcal{AN}).$$

In Abb. 3 sind Beziehungen von Ressourcen zwischen verschiedenen Domänen mit Hilfe der Annotation *depends* bezeichnet. Diese Annotation wird für alle interdomänen-Beziehungen verwendet und ist eine vordefinierte *Systemannotation*. Weitere Systemannotationen sind u.a. *versionOf* zur Darstellung einer Versionsabhängigkeit oder *createdBy* zur Darstellung, dass eine Ressource durch einen Nutzer erzeugt wurde.

4 Integrationsprozess und weitere Applikationen

Aufbauend auf der Architekturbeschreibung und der Datenbankstruktur wird in diesem Abschnitt der interaktive Prozess der Integration beschrieben, welcher sukzessiv die Datenbank aufbaut. Der Prozess besteht folgenden Schritten:

1. Registrierung von Datenquellen: Nutzer registrieren sich und die lokalen Repositories beim LinkDB-Server. Damit stehen die Daten für weitere Dienste zur Verfügung. Die lokalen Repositories stellen sicher, dass die Daten in einem XML-Format beschrieben werden, bspw. ModelicaXML für die Simulationssprache Modelica, und über eine URI zugreifbar sind.

2. Erstellung von Ressourcen: Mit Hilfe des Annotationsclients, der XML-Daten darstellen kann, exportiert der Nutzer seine lokalen Daten und deren Beziehungen als VE-Ressourcen und VE-Beziehungen. Dadurch werden diese Daten für die Annotatierung und die Nutzung durch andere User zugänglich.

3. Erstellung von Annotationen: Durch die Zuweisung von Annotationen an eigene und fremde Ressourcen beschreiben Nutzer die VE-Daten in einem Bottom-Up-Prozess. Somit wird sukzessive eine Beschreibung der Informationen aufgebaut.

4. Erstellung von Abhängigkeiten: Interdomänen-Beziehungen werden im Annotationsclient erstellt. Die Beziehungen erhalten jeweils die Annotation *depends*. In Zukunft sollen spezielle Transformationswerkzeuge(z.B.[3]) integriert werden, um die Aufgabe zu erleichtern.

5. Beschreibungen von Abhängigkeiten: Interdomänen-Beziehungen werden ebenfalls mittels Annotatierungen durch den Nutzer beschrieben.
Durch die Datenbank können die im Abschnitt 1 genannten Aufgaben wie folgt erfüllt werden:
Integration bei Beibehaltung der Entwicklersichten: Bei der Erstellung einer neuen Version eines Datenobjekts werden die Beziehungen der zugehörigen Ressource überprüft. Werden hierbei Abhängigkeiten durch Verfolgung von Kanten zu anderen Domänen erkannt, können die jeweiligen Nutzer durch das System über Änderungen benachrichtigt werden.
Dokumentation und Beschreibung: Annotationen beinhalten automatisch die Beschreibung der Daten. Durch die Annotierungen werden sie zu den Datenobjekten direkt zugeordnet. Somit wird es möglich, Beschreibungen zu speichern, im VE-Prozess allgemein zugänglich zu machen und erweiterte Suchmöglichkeiten zu implementieren.
Soziale Interaktion: Da Nutzer explizit modelliert sind, ist es möglich wie in Social-Tagging-Systemen [4] "Experten" für bestimmte Aufgaben zu finden, indem man deren annotierte Ressourcen untersucht.

5 Zusammenfassung

Ziel dieser Arbeit war die Entwicklung einer Struktur für eine Link-Datenbank, die Darstellung der Architektur und die zugehörigen Integrationsprozesse als Grundlage der interaktiven Integration und Beschreibung von heterogenen, komplex-strukturieren Daten in VE-Prozessen. Im Moment wird das System umgesetzt. Im Anschluss soll eine erste Akzeptanz bei Projektpartnern im Bereich des Maschinenbaus getestet werden. In einem weiteren Schritt sollen die Annotationen durch die Nutzung von Ontologien organisiert werden, wie bspw. in [2] für wissenschaftliche Daten vorgeschlagen. Durch die Informationen in den Ontologien sind weitere Konsistenzprüfungen möglich. Weiterhin wird daran gearbeitet, domänenspezifische Anwendungen mit dem LinkDB-Server zu koppeln, sowie automatische CAD-zu-CAE-Transformationsprozesse zur Erzeugung von Abhängigkeiten zu nutzen. Beides soll dazu beitragen, die Akzeptanz im Anwendungsbereich zu erhöhen.

Literatur

[1] Sabeur Bettaieb and Frederic Noel. A generic architecture to synchronise design models issued from heterogeneous business tools: towards more interoperability between design expertises. *Engineering with Computers*, 2007. online first.

[2] Michael Gertz and Kai-Uwe Sattler. Integrating Scientific Data through External, Concept-Based Annotations. In *VLDB 2002 Workshop EEXTT*, volume 2590 of *Lecture Notes in Computer Science*, pages 220–240. Springer, 2002.

[3] Tamas Juhasz and Ulrich Schmucker. Automatic Model Conversion to Modelica for Dymola-based Mechatronic Simulation. In *Modelica 2008, March 3rd-4th, 2008*, pages 719 – 726. The Modelica Association, 2008.

[4] David R. Millen, Jonathan Feinberg, and Bernard Kerr. Dogear: Social bookmarking in the enterprise. In *CHI '06: Proc. of the SIGCHI conference*, pages 111–120. ACM, 2006.

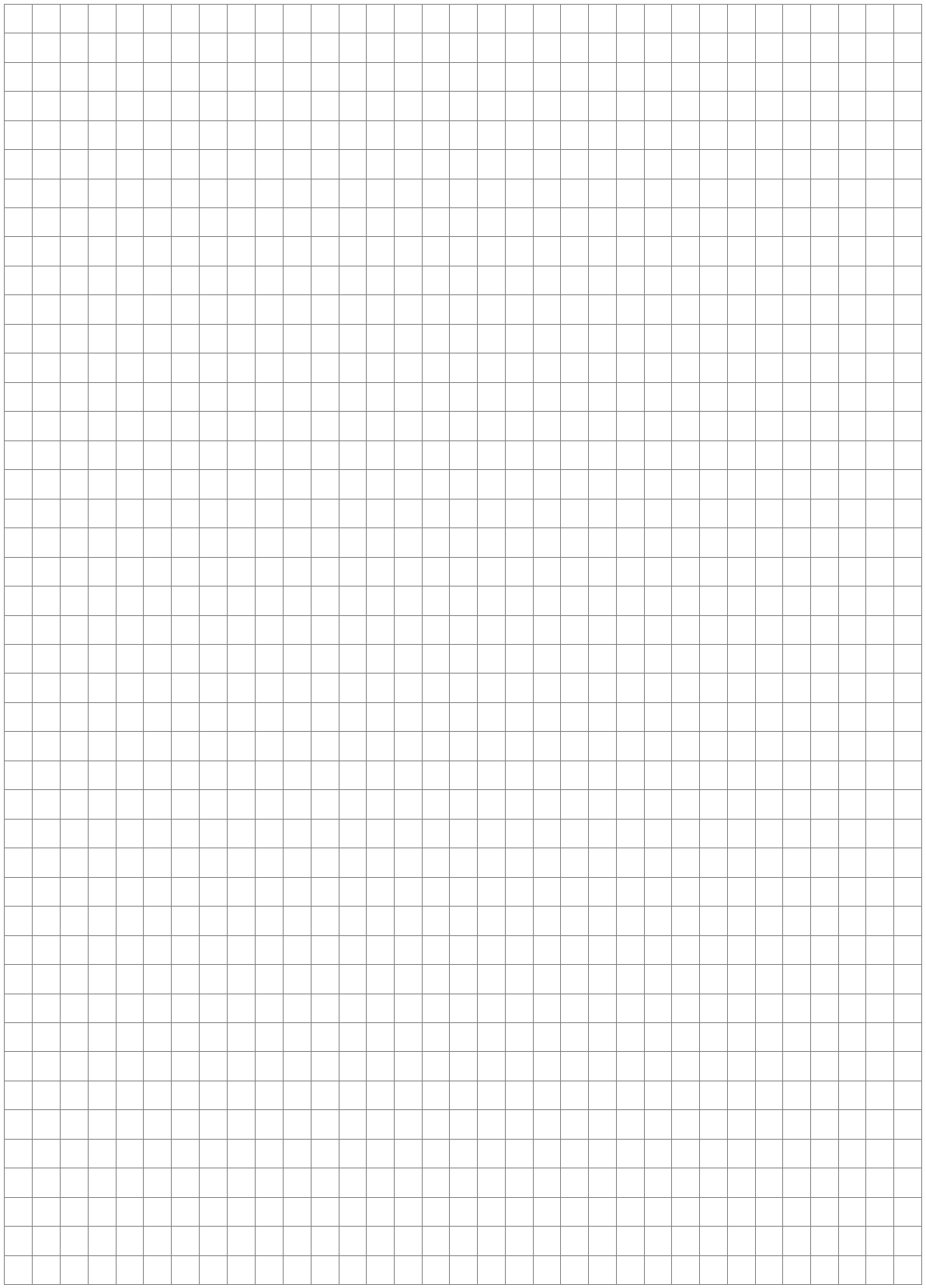
[5] Gregory M. Mocko and Steven J. Fenves. A Survey of Design – Analysis Integration Issues. Technical Report NISTIR 6996, National Institute Of Standards and Technology, May 2003.

[6] Gerhard Pahl, Wolfgang Beitz, Jörg Feldhusen, and K. H. Grote. *Konstruktionslehre : Grundlagen erfolgreicher Produktentwicklung. Methoden und Anwendung.* Springer, Berlin, 2007.

[7] Noel Titus, Eswaran Subrahmanian, and Karthik Ramani. Folksonomy and Designing: An Exploration. In *Proc. ASME IDETC/CIE 2007.* ASME, 2007.

[8] T. Vander Wal. Folksonomy Coinage and Definition. http://vanderal.net/folksonomy.html, February 2007.

[9] F. Zorriassatine, C. Wykes, R. Parkin, and N. Gindy. A survey of virtual prototyping techniques for mechanical product development. *Journal of Engineering Manufacture*, 217:513 – 530, 2003.



Towards Real-Time Data Integration and Analysis for Embedded Devices

Michael Soffner, Norbert Siegmund, Mario Pukall, Veit Köppen
Otto-von-Guericke University of Magdeburg
{soffner, nsiegmun, pukall, vkoeppen}@ovgu.de

In complex systems, e.g., in logistic hubs, cars or factories, there is a need for real time decision support. In present approaches the transfer and storage process and subsequent analysis of data in real time is not possible. One reason is that data sources can fail and thus the information flow is interrupted. Furthermore there is a large divergence of the data amount generated by data streams coming from different data sources, e.g. sensors, relational databases and mobile devices. We offer an architecture to eliminate these problems. Therefore we introduce a classification of the data sources that enables an appropriate handling of the specific data source's properties.

Keywords: data integration, embedded systems, real-time, sensor networks

1 Introduction

The increasing usage of embedded systems leads to more complex heterogeneous systems. In ViERforES[1] project's context we are working with the scenario logistic hub. The logistic hub's daily business comprehends the flow of goods. Thereby, goods pass scanners, conveyors and robots, that are collecting data and controlling their flow and transfer information into central data storage systems (further details see section 2). The information is monitored and evaluated through control stations. To get the state of such complex system, e.g., the transport status or the determination of anomalies of dangerous good's actual position. There has to be found an appropriate way to run analysis on the collected information. Data Warehouses [Kim96, Inm05, BG04] provide concepts and solutions, which allow an integration as well as a flexible and efficient analysis of data. Thereby queries are usually processed on aggregated data and prepared to fit the respective department needs. Data collection and transformation processes are time consuming and additionally the data aggregation usually has to perform huge amounts of data. In some application areas, this is not possible. The problem is that, in the scenario logistic hub, the data comes from heterogeneous and distributed sources, e.g., embedded systems, and has to be gathered, aggregated and analyzed in real time. For example the logistic hub's control station is faced with problems of availability and reliability of required data. Decisions have to be made to assure that goods are delivered in-time and in-place. To support those decisions, a guarantee for real time delivery including the availability of data and processing time of analyses has to be given.

There are already Data Warehouse approaches, that provide near real time capabilities (see section 3). However, these approaches don't regard heterogeneous, distributed, and unstable data sources. A special challenge is the varying amount of input data. For example if a catastrophe on an airport takes place. That's why at this airport no airplanes can land anymore. Thus the still flying airplanes have to land somewhere else. The other airport has to handle

[1] ViERforES project web page `http://vierfores.de`

those unexpected airplanes and their data as well. Another challenge is the unreliability of the data gathering systems. Unreliability of those systems is reasoned for example by limited energy supply, e.g., wireless sensors, or they are error-prone due to environmental reasons. This situation demands special algorithms to be implemented in the extract, transform, and load (ETL) process [BG04, Inm05] to warrant a stable data quality in order to derive trustworthy information. In this paper, we propose an architecture including a classification to reduce the data amount and to compensate failures.

2 Application Scenario Logistic Hub

Within the scope of the project ViERforES, we examine different domains including the logistic hub of an airport. Therefore, in this section we give answers to the questions, which kind of systems can provide information and what are the challenges in that environment. The main task on the airfield of logistic hubs is to load and unload the airplane's cargo. Therefore, different people are involved to manage this process, like ramp agents or chief supervisors in the control station. These people need information to decide whether the airplain is ready to take off or ready to be loaded. They also decide if it has to be checked in more detail due to possible airplane's system errors. This process is assisted by embedded systems. Those embedded systems are integrated in each of the process's supporting equipment, e.g., the unit load device (ULD) that sorts the cargo to make it more manageable. In future, cargo containers are supplied with Radio-frequency identification (RFID) chips that can transmit ID values and further data. Furthermore dollies can be equipped with sensors to signify their loading status. This means, if there is anything loaded and what is loaded. Again, this information is provided by the ULDs. To enable the tracking of airplanes and dollies, GPS sensors or RFID technology can be used. Therefore, they can be located on the ramp at any time. Airplanes can automatically deliver information about their loading status, how many ULD they have loaded and, during the unloading, how many of them have already been unloaded. The shelter where dollies and other unused equipment is stored delivers information about how many empty dollies are parked. The Warehouse provides data about which ULDs have arrived and how long the unloading of the ULD had taken. Another source are reports that are supported by PDAs and filled in by ramp agents. Those PDAs can be a source as well as a sink of information. The ramp agent gets help by proposals of the unloading status generated through the system, that basically got its information out of integrated sensors.

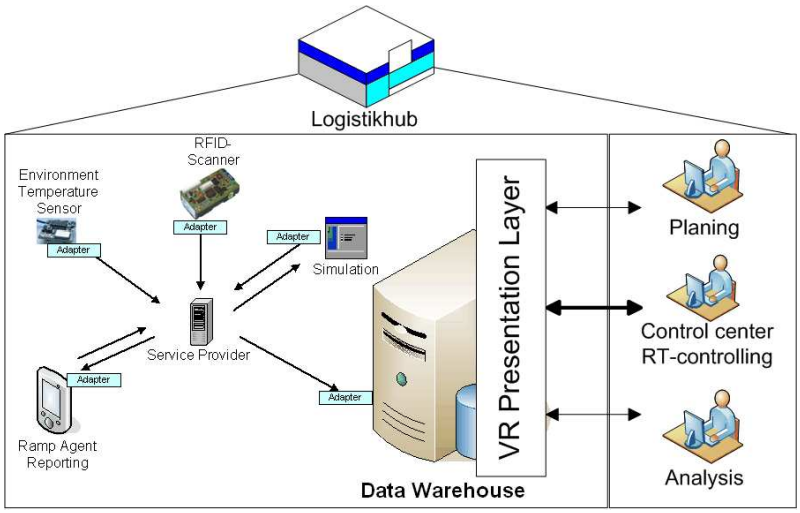


Figure 1: Sensor Network containing sensors of an airfields logistic hub

In the left part of figure 1 there are different embedded systems like those that have been described above. They are connected through a service-oriented architecture. Thereto, they have to register at a service provider, that administrates all services. The Data Warehouse is one of the registered services. In the figure's right part there are three different departments that use the Data Warehouse. One of them is the above mentioned control center.

All the before described data can be used to support the decisions made by ramp agents and the ramp control. Thereby, there are some challenges to retrieve the data. The first one is the real time challenge. On the one hand the data will be generated in time series, like position tracking, or the airplane's loading status. On the other hand data needs to be presented in real time. The ramp control needs to know the actual loading status to decide if it is ready to take off or not. Furthermore, if we consider the rising amount of data, a supporting system needs to handle a constant amount of data not only in common situations. Especially in critical situations where decisions are needed the system needs to work.

3 Real-Time Data Warehouse Architecture

This section will show problems and give an overview about existing ideas of connecting real time data with a Data Warehouse.

To be able to run analyses we assume that the data can be stored in an persistent way. Actually, this might be quite a challenge. Regarding that in our scenario the data is generated by sensors we have to take a look at what the actual amount of data might be. In [HB09] they benchmarked existing database systems. The benchmark was aimed at determining the read and write performance of existing database systems. The interesting part was and still is that just one of the tested database systems provided a sufficient performance. The others, e.g., Oracle, couldn't attain the needed performance. They assumed a data volume of 2500kb per second. If we imagine the raising usage and the potential amount of sensors on a logistic hub this data rate is easily achieved. Hence a wireless communication network is potentially used, we can assume higher data rates, e.g., the actual wireless LAN standard IEEE 802.11n.

One approach is called *trickle and flip* [Lan04]. Thereby, a copy of the original fact table is made. It is directly fed with the real time data. After a defined period of time the import of new data is stopped and this copy is copied again. The new copy is renamed to the original fact table and the old fact table is deleted. The process is called the flip. While flipping there should be no queries executed. The problem of this approach is that it is not scalable. Hence, we assume a huge amount of data this approach doesn't fit.

The idea of *table partitioning* [BG04, Rad03] is to use the feature of relational databases to create table partitions. Therefore a criteria is defined that decides about when a new partition will be created. With respect to the real time aspect this criteria will be the time. Thus the real time data will be stored in the latest partition. To prevent running queries on it, this has to be hided through the Data Warehouse. Otherwise the loading process would be disturbed. The problem in here is to find the right period of time. The aim should be that the real time partition fits in the memory. Due to the incoming sensor information the interval a day this interval can be short. Like the approach before, this approach will have a problem because the high amount of data. Thus an shorter interval must be chosen. This causes that there will be to many partitions that will decrease the performance.

Real Time partitions are introduced by Ralph Kimball [Kim02,KRT+08] to cope with the new requirement of fresh data. This approach is based on the idea to separate real time fact tables from the usual static Data Warehouse. Until the static Data Warehouse holds the historical data, the real time partition keeps the data, that was generated or changed starting at the time of the last warehouse update. Their are slightly differences in how the data is stored in the static and real time partition. It depends on what fact tables grain was chosen. The design goal should be that the real time fits into the memory. Thus the query and loading performance

can satisfy the refreshment needs demanded by the user. Although this approach is scalable the database management system hast to provide the functionality. As we have seen in the above example usual database systems have problems with the amount of data. Furthermore, the approach doesn't mention about how to load the huge amount of data and to handle unstable data sources.

The *real time data cache* [Lan04] is a completely separated database server that handles the real time data. The tables in the data cache database are modeled like the Data Warehouse tables. Additionally, just those tables that hold real time data will be created in the data cache. If analysis on the data cache's data need historical data, then this data is just in time queried from a static Data Warehouse. This approach basically has the same deficiencies like that one before.

Furthermore there is another critical aspect that needs to be considered. As usual, in real time scenarios the continuous data stream demands a way to continuously load the data. Additionally, the network load itself is another aspect. In [VS08] a workflow architecture of the ETL process is proposed. It offers a possibility to extract and load data that doesn't influence the source's and target's performance. But it doesn't regard the network load itself and it doesn't handle the information loss caused by crashed sources.

4 The Reliable Information Architecture

We have seen so far that current approaches cannot suitable handle requirements arising in complex systems that have to face reliability problems. Therefore, we propose in our architecture (see figure 2) a classification of the data providing sensors or embedded devices respectively. We define three classes:

- *Preprocessor class* contains data sources, e.g., embedded devices, database services, and sensors that locally preprocess the data, i.e., directly on the device. This is needed due to a too large amount of data that has to be transferred. Preprocessing methods include aggregation functionality, data cleaning, etc.
- *Intermediate Aggregation class* contains nodes, where data is aggregated. Furthermore, nodes that send data to be aggregated belong to this class as well. Using these aggregation nodes we can significantly reduce the network load. Additionally, an increased reliability of data is achieved due to a compensation of single node failures.
- *Direct Input class* are those data sources transferring their data directly to the Data Warehouse. Examples are time critical systems and traditional Data Warehouse sources. Thus, mission critical data can be provided straight forward to the control center.

As an example let us take embedded systems that are measuring temperature. We assume that basically the temperature sensors of the system provide data 5 times a second. Now, devices have different possibilities to handle the data. A device would be classified as preprocessor if it aggregates the temperature over a period of 5 minutes locally. This value is send to the Data Warehouse. In another case there are some devices that don't send their measured values to the Data Warehouse either preprocessed or directly. They transfer them to an intermediate node which gets this data and values from other sub devices too, e.g., a hall's temperature determination node that aggregates the values from different measures. Those measures take the temperature of heat sources in that hall. Both the node that sends data to the intermediate node and the intermediate node itself are members of the intermediate aggregation class. Another device, e.g., used in more critical area with for example temperature sensitive liquids transfers the data directly to the Data Warehouse as it comes. Those device are members of the direct input class.

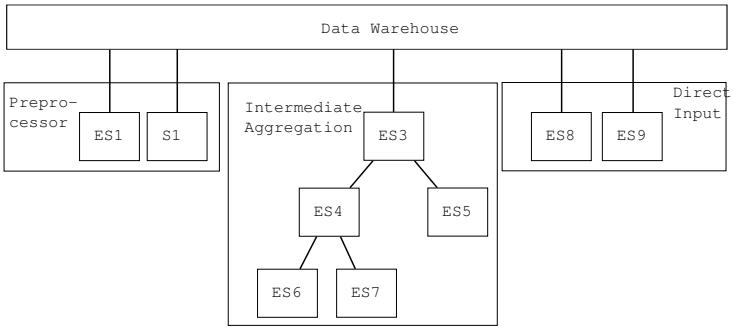


Figure 2: The Reliable Integration Architecture

This classification has two major advantages: (1) the reduced amount of incoming data and (2) the reduced network load. Additionally, it can be used to evaluate existing architectures. This information can help to make changes and extensions. Furthermore, based on the classification, we propose different storage and processing functions to overcome the mentioned problems.

5 Conclusion

In this paper we have shown problems of the real time decision support in today's complex systems. Therefore, we have taken the scenario logistic hub as an example. Afterwards, we have discussed existing approaches to store and transfer real time data and have shown their deficiencies regarding the mentioned problems. Hereupon, we have proposed a reliable information architecture that offers a classification of existing systems and eliminates the mentioned problems. The future work encloses an implementation and evaluation of our proposed architecture.

References

[BG04] Andreas Bauer and Holger Günzel. *Data Warehouse Systeme: Architektur, Entwicklung, Anwendung*. Heidelberg:dpunkt-Verlag, 2te, überarbeitete und aktualisierte auflage edition, 2004. ISBN 3-89864-251-8.

[HB09] Daniel Herold and Wolfgang Benn. Sensordatenablage in echtzeit. *Datenbank Spektrum*, 28:31–36, Feb 2009.

[Inm05] William H. Inmon. *Building the Data Warehouse*. New York:John Wiley & Sons, 4th edition, Oct 2005. ISBN 978-0-7645-9944-6.

[Kim96] Ralph Kimball. *The Data Warehouse Toolkit: Practical Techniques for Building Dimensional Data Warehouses*. Wiley & Sons, 1996. ISBN 0-471-15337-0.

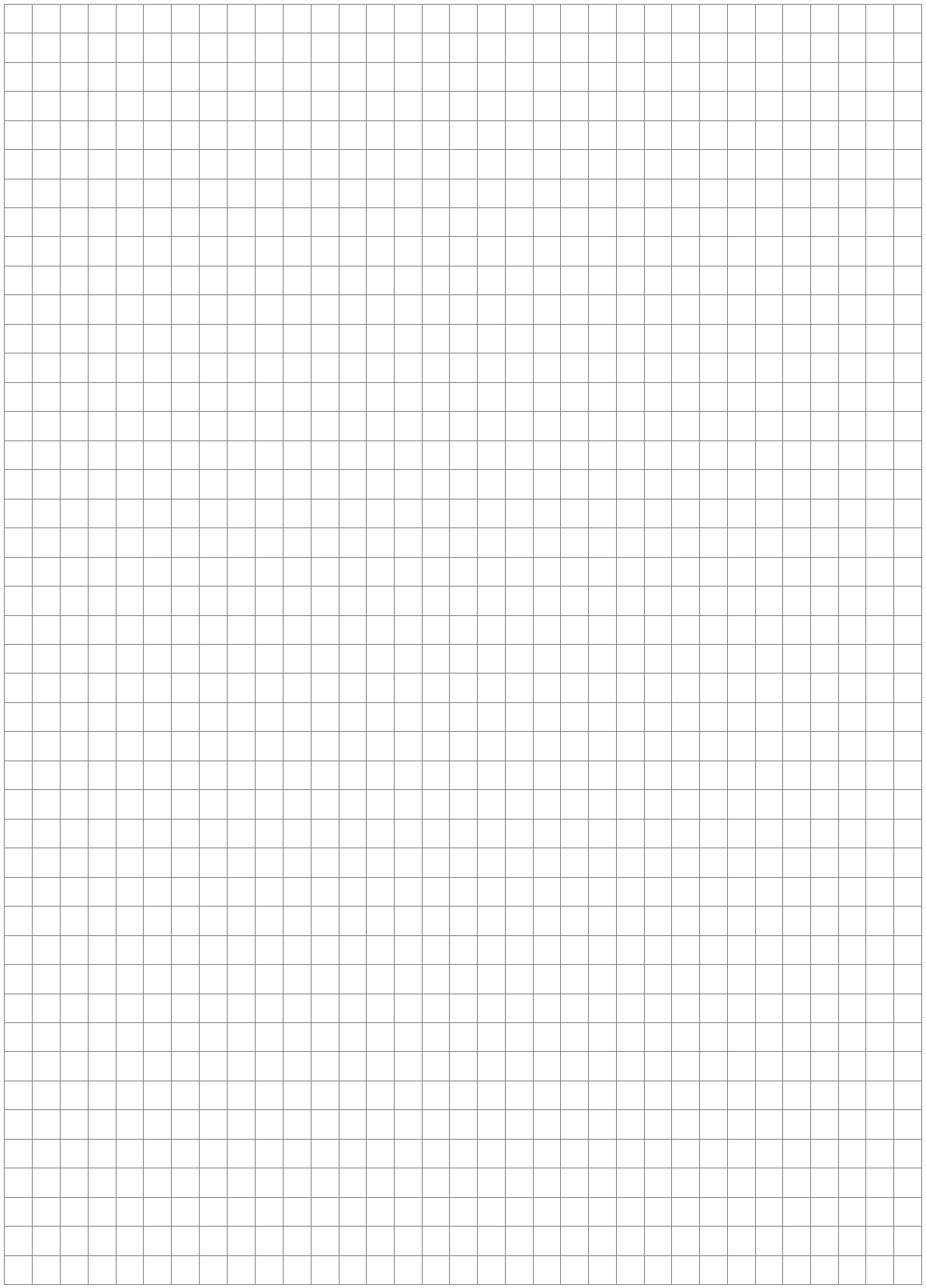
[Kim02] Ralph Kimball. Realtime partitions, February 2002. Intelligent Enterprise Magazine, `http://www.intelligententerprise.com/020201/503warehouse1_1.jhtml`.

[KRT+08] Ralph Kimball, Margy Ross, Warren Thornthwaite, Joy Mundy, and Bob Becker. *The Data Warehouse Lifecycle Toolkit*. Wiley & Sons, 2nd edition, January 2008. ISBN 978-0470149775.

[Lan04] Justin Langseth. Real-time data warehousing: Challenges and solutions, 2004. DSSResources.COM, `http://dssresources.com/papers/features/langseth/langseth02082004.html`.

[Rad03] Neil Raden. Real time: Get real, part ii, June 2003. Intelligent Enterprise Magazine, `http://www.intelligententerprise.com/030630/611warehouse1_1.jhtml`.

[VS08] Phano Vassiliadis and Alkis Simitsis. *Near Real Time ETL*, volume 3 of *Annals of Information Systems*, chapter 2. Springer US, 2008. ISBN 978-0-387-87430-2.



Ein Hybridansatz zur Interaktion mit Retrievalsystemen

David Zellhöfer
Institut für Informatik
Brandenburgische Techn. Universität Cottbus
david.zellhoefer@tu-cottbus.de

Zusammenfassung

Gängige Retrievalsysteme nutzen qualitative oder quantitative Ansätze, um das subjektive Ähnlichkeitsempfinden der Nutzer bzw. ihre Präferenzen bezüglich einer Anfrage zu formulieren. Die beiden Ansätze treten dabei meist getrennt auf. Vertreter qualitativer Ansätze sehen quantitative Ansätze, die Ergebnismengen auf Basis numerischer Score-Werte und Gewichtungen erzeugen und damit eine Totalordnung generieren, meist als Spezialisierung ihrer Methoden.

Die vorgestellte Arbeit versucht, die Vorzüge beider Methoden zu verbinden, um dem Nutzer eine vereinfachte Interaktion während des Relevance Feedbacks mit dem Retrievalsystem zu bieten, ohne auf die höhere Ausdifferenzierung der Ergebnismenge quantitativer Ansätze verzichten zu müssen. Die Interaktion mit dem System geschieht dabei allein auf Basis der Angabe von intuitiv verständlichen Präferenzpaaren, die wiederum eine Halbordnung bilden. Hierbei muss angemerkt werden, dass die Angabe einer Präferenz $\geq$ immer auf einzelnen Paaren der Ergebnismenge geschieht und nicht mittels *abstrakter* Präferenzen, deren konkrete Angabe gerade im Feld des Multimedia Retrievals schwierig erscheint. Diese nutzerdefinierte Halbordnung stellt die Grundlage für ein maschinelles Lernverfahren auf Basis einer nicht-linearen Optimierung dar, welches aus den speziellen Präferenzen allgemeingültige Gewichtungen für die Anfrage generiert.

Erste Ergebnisse experimenteller Untersuchungen zeigen, dass der vorgestellte Ansatz gut einsetzbar ist und weiterverfolgt werden sollte.

1 Motivation

Retrievalsysteme sind gegenüber traditionellen Datenbanksystemen dadurch gekennzeichnet, dass sie eine Anfrage seitens des Nutzer mit relevanten Ergebnisobjekten beantworten. Diese Ergebnisobjekte müssen dabei eine Anfrage nicht komplett erfüllen, wie es im Fall eines DBS, das in der Regel auf der Booleschen Logik basiert, notwendig ist. Wichtiger ist bei diesen Systemen die Ähnlichkeit eines Objekts zur Anfrage. Das Konzept der Ähnlichkeit ist dabei subjektiv und basiert im wesentlichen auf den Erwartungen des Nutzers. Psychologische Studien stützen dies [2, 11]. Experimente im Information-Retrieval zeigen, dass einzelne Bedingungen einer Anfrage subjektiv wichtiger wahrgenommen werden und damit das Ähnlichkeitsempfinden steuern. Die Möglichkeit einzelne Bedingungen zu gewichten erhöht außerdem die Nutzerzufriedenheit [8].

Folglich nutzen gängige Retrievalsysteme Methoden, um das subjektive Ähnlichkeitsempfinden der Nutzer bzw. ihre Präferenzen bezüglich einer Anfrage zu formulieren. Präferenzen ermöglichen es dem Nutzer, einzelne Bedingungen der Anfrage stärker zu gewichten oder oder als besonders relevant anzugeben. Üblicherweise werden die Möglichkeiten zur Angabe von Präferenzen in zwei Klassen unterteilt: die quantitativen [10, 5] und die qualitativen Ansätze [4, 6]. Diese Klassen treten stets getrennt auf. Eine Kombination ist in der Regel nicht vorgesehen.

Die qualitativen Ansätze setzen voraus, dass der Nutzer bereits zum Zeitpunkt der Anfrageformulierung seine Präferenzen kennt. Präferenzen und Ergebnismengen werden hierbei mittels Halbordnungen abgebildet. Präferenzen erweitern die Boolesche Anfragelogik und lassen sich deshalb gut in relationalen DBS umsetzen. Beim Skyline-Operator [1] handelt es sich ebenfalls um einen qualitativen Ansatz.

In Abgrenzung dazu wird beim quantitativen Ansatz eine Totalordnung der Ergebnisobjekte auf der Grundlage einer numerischen Scoring-Funktion erzeugt, wie dies in Retrievalsystemen üblich ist. Präferenzen werden bei diesen Ansätzen in der Regel als numerischer Gewichtungswert über Anfragebedingungen ausgedrückt. Anhänger qualitativer Ansätze betrachten deshalb quantitative Verfahren als Spezialfall der qualitativen Verfahren, da jede Totalordnung auch eine Halbordnung ist. Dem kann entgegengebracht werden, dass ebenfalls jede Halbordnung mit der Dushnik-Miller-Dimension d durch den Schnitt von d Totalordnungen ausgedrückt werden kann.

Beide Verfahren haben ihre Daseinsberechtigung in unterschiedlichen Einsatzszenarien. Qualitative Verfahren bieten sich vor allem für den Einsatz in relationalen DBS an, da sie sich dort ohne größere Hindernisse implementieren lassen [4, 6]. Sie setzen allerdings voraus, dass die Präferenzen des Nutzers bereits zum Zeitpunkt der Anfrageformulierung bekannt sind.

Beispiel 1.1 *Denkbar ist dies z.B. bei der Wohnungssuche. Hier kann der Benutzer klar angeben, dass er Wohnungen mit Balkon gegenüber solchen ohne bevorzugt und und wenn eine Wohnung die gleiche Lage hat, die günstigste gewählt wird.*

Tabelle 1: Beispieldatenbank Wohnungen

ID	Lage	Balkon	Etage	Miete
A	Kreuzberg	ja	1	510
B	Mitte	ja	2	490
C	Kreuzberg	ja	5	450
D	Kreuzberg	ja	4	500
E	Neukölln	ja	4	350

Die entstehende Halbordnung würde die Wohnungen B,C und E als relevant einstufen. Eine Aussage über eine Ordnung innerhalb der Kreuzberger Wohnungen kann nicht getroffen werden, da diese nicht spezifiziert wurde[1]. Die Interpretation der Präferenz folgt somit dem Ceteris-paribus-Prinzip. D.h. dass die die Präferenz erfüllenden Objekte vor allen anderen stehen und die übrigen als gleichwertig betrachtet werden.

Diese Form des Ergebnisses lässt sich nicht mit quantitativen Ansätzen erreichen. Für diesen Fall müsste eine Scoring-Funktion gefunden werden, welche die Präferenz aus Beispiel 1.1 ausdrückt. Da keine Präferenz zwischen den Kreuzberger Wohnungen und z.B. Wohnung B angegeben wurde, müsste der Score von B gleich dem der Kreuzberger Wohnungen sein. Daraus folgt, dass der Score aller Kreuzberger Wohnung ebenfalls gleich sein müsste, was einen Widerspruch darstellt, da gleichzeitig die günstigste Wohnung gewählt werden sollte [3]. Trotz dieser Einschränkung bieten quantitative Ansätze ein feiner ausdifferenziertes Ergebnis, was durch folgendes Beispiel verdeutlicht werden soll.

Beispiel 1.2 *Der Nutzer bevorzugt Wohnungen mit Balkon, wobei ein möglichst günstiger Preis wichtig ist. Tabelle 2 zeigt ein denkbares Ergebnis als Totalordnung, wobei die Teilbedingungen unterschiedlich mittels einer Scoring-Funktion gewichtet wurden, so dass der Preis den größten Einfluss hat.*

Die Wohnungen A und D erhalten hier die niedrigsten Score-Werte, da sie am teuersten sind. Über die Score-Werte ist es möglich anzugeben, inwiefern sich die einzelnen Objekte bzgl. ihrer Relevanz zur Anfrage unterscheiden. Qualitative Ansätze ermöglichen keine Unterscheidung zwischen „wesentlich“ und „geringfügig besser“. Die beiden Wohnungen wären als gleichwertig betrachtet worden (s.o.).
Da konkrete Präferenzen bei quantitativen Verfahren noch nicht zu Beginn der Anfrage feststehen müssen, ist es möglich, dass Präferenzen während der Interaktion mit dem System angepasst werden können. Dies

[1] Dies gilt natürlich auch für eine Ordnung zwischen B,C und E.

Tabelle 2: Ranking der Wohnungen in einem quantitativen Verfahren

Score	ID	Lage	Balkon	Etage	Miete
0.8	E	Neukölln	ja	4	350
0.7	C	Kreuzberg	ja	5	450
0.6	B	Mitte	ja	2	490
0.5	D	Kreuzberg	ja	4	500
0.4	A	Kreuzberg	ja	1	510

wird mit einer höheren Komplexität der Präferenzangabe erkauft, da Präferenzen als Gewichtswerte für Anfrageteile numerisch angegeben werden müssen, was gerade bei komplexen Anfragen oder schwer verständlichen Anfragebedingungen eine Hürde für den Nutzer darstellt. Ein ähnliches Problem während der Nutzerinteraktion tritt auch bei qualitativen Verfahren auf. So ist es in einem Retrievalsystem kaum denkbar, dass Anwender konkret angeben können, welche Feature-Werte, wie z.B. Textureigenschaften, ihnen gegenüber anderen wichtiger sind.

Durch geeignete Verfahren des maschinellen Lernens und die Nutzung eines speziellen Relevance-Feedback-Prozesses kann die Angabe konkreter Gewichte jedoch verborgen werden, wie im folgenden Abschnitt gezeigt wird.

2 Ein Hybridansatz für das Relevance Feedback

Die vorgestellte Arbeit verbindet die Vorzüge beider Ansätze, um dem Nutzer eine vereinfachte Interaktion während des Relevance Feedbacks mit dem Retrievalsystem zu bieten, ohne auf die höhere Ausdifferenzierung der Ergebnismenge quantitativer Ansätze verzichten zu müssen. Die Interaktion mit dem System geschieht dabei allein auf Basis der Angabe von intuitiv verständlichen Präferenzpaaren, die wiederum eine Halbordnung bilden, da die Angabe von Gewichtswerten eine zu große Hürde für den Anwender darstellt.

Abbildung 1 zeigt den konzeptionellen Interaktionsablauf. Das Verfahren ist intern quantitativ und nutzt zur Anfrageformulierung und Gewichtung die Anfragesprache CQQL [10]. Die Nutzerschnittstelle hingegen verwendet eine qualitative Präferenz-Metapher. Hierbei muss angemerkt werden, dass die Angabe einer Präferenz $\geq$ immer auf einzelnen Paaren der Ergebnismenge geschieht und nicht mittels *abstrakter* Präferenzen, wie sie bei traditionellen qualitativen Verfahren üblich sind. Die Formulierung abstrakter Präferenzen ist gerade bei komplexen Retrievalsystem schwierig, da sie eine Kenntnis über zugrundeliegende Mechanismen und die Anfrage voraussetzt. Bei der Ergebnismenge handelt es sich um eine Totalordnung (Rank), die anhand der Scores der Objekte sortiert ist. Der erste Rank basiert dabei auf einer gewichteten CQQL-Anfrage und einem initialen Gewichtungsschema, z.B. zufälligen oder Nutzerprofil-basierten Gewichtswerten. Weitere mögliche Gewichtungsschemata finden sich in [13].

Sämtliche Präferenzpaare bilden die Menge P, welche mitsamt der zugrundeliegende Anfrage die Eingabe eines Downhill-Simplex-Lernalgorithmus [7] bilden. P ist dabei frei von widersprüchlichen Präferenzen, z.B. Zyklen in der abgeleiteten Halbordnung. Details zu den verwendeten Algorithmen finden sich in [9]. Der Lernalgorithmus dient zum Finden von konkreten Gewichtswerten, die zusammen mit der Anfrage einen neuen Rank erzeugen, welcher den angegebenen Präferenzen des Nutzers entspricht. Diese Modifikation hat zur Folge, dass sich die Ergebnisobjekte mit der Zeit an die Erwartung des Nutzers (Finaler Rank) anpassen, da die Gewichte direkt aus den nutzerdefinierten Präferenzen P' abgeleitet werden.

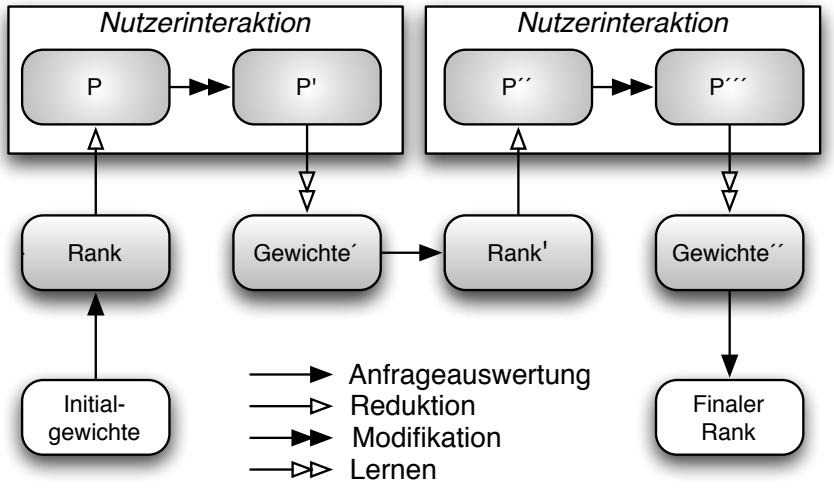


Abbildung 1: Verfeinerung von Gewichten durch Nutzerinteraktion mittels Präferenzen P

3 Fazit und Ausblick

Das vorgestellte Verfahren wurde experimentell anhand eines Szenarios zur Kamerasuche überprüft [9]. Dabei wurde gezeigt, dass das konzeptionelle Relevance-Feedback-Modell zu einer Annäherung an das vom Nutzer erwartete Ergebnis führt und ein Lernen von Gewichtswerten aus qualitativen Präferenzen möglich ist. Der zusätzliche Transformationsschritt und der Einsatz eines maschinellen Lernverfahrens zur Ermöglichung des Hybridverfahrens wirkt sich außerdem nicht kritisch auf die Laufzeit des Systems aus. Abb. 2 zeigt die Laufzeit der Lernalgorithmus in Abhängigkeit der Parameter Anzahl der Gewichtsvariablen und der nutzerdefinierten Präferenzen.

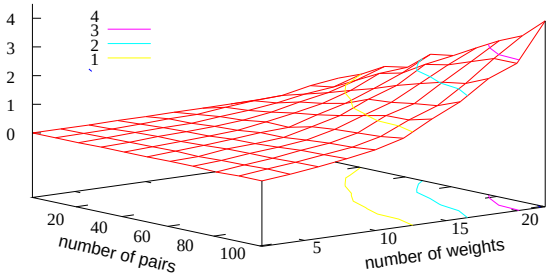


Abbildung 2: Laufzeit [s] in Abhängigkeit der Anzahl von Gewichten ($weights$) und Präferenzen ($pairs$)

Die Laufzeit für weniger als 10 Gewichtsvariablen innerhalb einer Anfrage und 40 Präferenzen liegt dabei deutlich unter einer Sekunde. Mit einer höheren Anzahl von Präferenzen ist nicht zu rechnen, da Nutzer in der Regel vor umfangreichen Interaktionen zurückschrecken [12]. Trotzdem erhöht eine große Anzahl von Gewichten und Präferenzen die Laufzeit nur langsam.

Abschließend kann festgestellt werden, dass sich qualitative und quantitative Verfahren kombinieren lassen. Im Bereich der Nutzerinteraktion sind qualitative Verfahren klar überlegen, da es Nutzern leichter fällt, Vergleiche mittels „besser als“ anzugeben als numerische Werte zu verwenden. Durch maschinelle Lernverfahren kann intern jedoch weiter ein quantitatives Verfahren genutzt werden, um eine möglichst große Ausdifferenzierung innerhalb der Ergebnismenge zu gewährleisten. Dies ist gerade im Retrievalbereich wünschenswert – bietet aber auch in anderen Szenarien bessere Möglichkeiten um beispielsweise die Ergebnismenge schrittweise zu vergrößern.

Literatur

[1] BÖRZSÖNYI, STEPHAN, DONALD KOSSMANN und KONRAD STOCKER: *The Skyline Operator*. In: *Proceedings of the 17th International Conference on Data Engineering*, Seiten 421–430, Washington, DC, USA, 2001. IEEE Computer Society.

[2] BRUCE, VICKI und PATRICK R. GREEN: *Visual Perception -physiology, psychology and ecology (2nd ed., reprinted)*. Lawrence Erlbaum Associates, Publishers, Hove and London, UK, 1993.

[3] CHOMICKI, JAN: *Querying with Intrinsic Preferences*. In: *EDBT '02: Proceedings of the 8th International Conference on Extending Database Technology*, Seiten 34–51, London, UK, 2002. Springer-Verlag.

[4] CHOMICKI, JAN: *Preference formulas in relational queries*. ACM Trans. Database Syst., 28(4):427–466, 2003.

[5] FAGIN, R. und E. L. WIMMERS: *A Formula for Incorporating Weights into Scoring Rules*. Theoretical Computer Science, 239(2):309–338, 2000.

[6] KIESSLING, W.: *Foundations of Preferences in Database Systems*. In: *Proc. of the 28th Int. Conf. on Very Large Data Bases, VLDB'02, Hong Kong, China, August, 2002*, Seiten 311–322. Morgan Kaufmann Publishers, 2002.

[7] NELDER, J. A. und R. MEAD: *A Simplex Method for Function Minimization*. Computer Journal, 7:308–313, 1965.

[8] SALTON, GERARD, EDWARD A. FOX und HARRY WU: *Extended Boolean Information Retrieval*. Commun. ACM, 26(11):1022–1036, 1983.

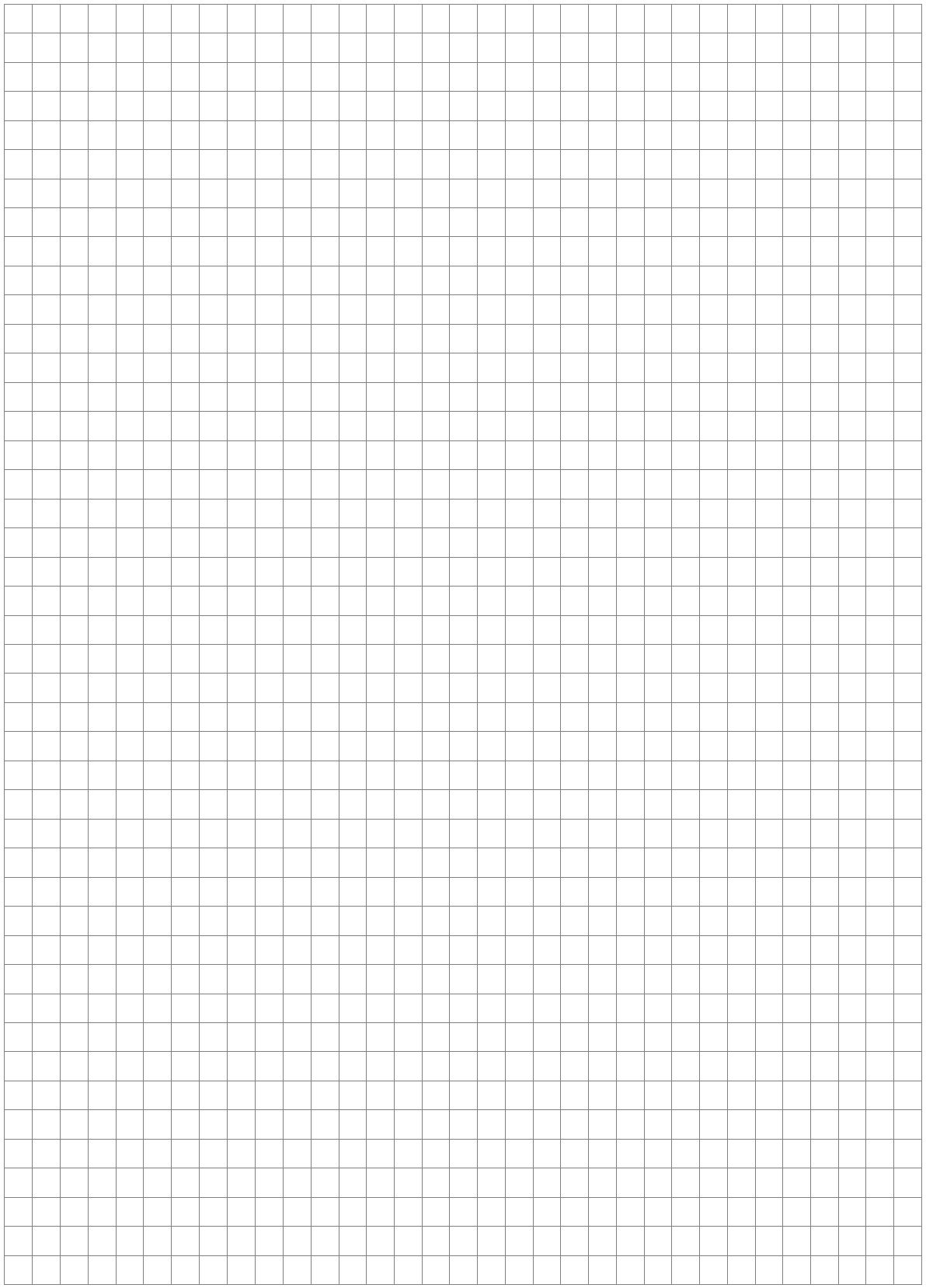
[9] SCHMITT, I. und D. ZELLHÖFER: *Lernen nutzerspezifischer Gewichte innerhalb einer logikbasierten Anfragesprache*. In: *Datenbanksysteme in Business, Technologie und Web, BTW'09, Lecture Notes in Informatics P-144*, Seiten 137–156. GI-Edition, 2009.

[10] SCHMITT, INGO: *QQL: A DB&IR Query Language*. The VLDB Journal, 17(1):39–56, 2008.

[11] SELFRIDGE, O. G.: *Pandemonium. A paradigm for learning*. The mechanics of thought processes, 1959.

[12] SHNEIDERMAN, BEN und CATHERINE PLAISANT: *Designing the user interface: Strategies for effective human–computer interaction*. Pearson, Boston, 4. ed. Auflage, 2005.

[13] ZELLHÖFER, D. und I. SCHMITT: *A Poset Based Approach for Condition Weighting*. In: *6th International Workshop on Adaptive Multimedia Retrieval, to appear*, 2009.



A quantum logic-based query processing approach for extending relational query languages

Sebastian Lehrack
Institut für Informatik
Brandenburgische Techn. Universität Cottbus
slehrack@informatik.tu-cottbus.de

May 1, 2009

Abstract

Evaluating a traditional database query against a data tuple returns a `true` on match and a `false` on mismatch. Unfortunately, there are many application scenarios where such an evaluation is not possible or does not adequately meet users needs. A further problematic application area is text retrieval where in general finding a complete match is impossible. Thus, there is a need for incorporating impreciseness and proximity into a logic-based query language: Objects fulfill such a *similarity condition* to a certain degree which is expressed by a result value out of the interval [0, 1]. In this work we will sketch a quantum logic-based approach, which provides the combination of classical Boolean predicates and similarity conditions into one integrating formalism.

1 Introduction

To motivate our query processing approach we want to consider an example, which is dealing with the assessment of TV sets. So, in an online shop a user may look for a very comfortable but inexpensive TV set by means of a query expressing these requirements. In a database following attributes are saved for a certain TV set: `name, price, status, handling` and `image_quality`. The two last attributes contain a rating of the respective properties decoded as marks from 1 to 6. Thereby, the mark 1 stands for an *excellent* test result and the mark 6 confirms an *inadequate* quality for the tested feature. The Table 1 gives an extracted part of the entire data spreadsheet. The user defines his query more precisely as: I want to find a device, which can be handled as easy as possible and its price does not exceed 1.000 Euro. If otherwise the cost of a certain TV set is more than 1.000 Euro, it should provide the best possible quality of image at least. The vagueness of the subconditions *handling as easy as possible* and *best possible quality of image* cannot be mapped to Boolean truth values. In Table 1 the truth values for a Boolean evaluation are given in parentheses, when the threshold value for an acceptable mark is assumed to 2.

TV sets				
`name`	`price`	`handling`	`image_quality`	...
TV1	500 (1)	3 (0)	4 (0)	...
TV2	800 (1)	2 (1)	2 (1)	...
TV3	900 (1)	1 (1)	4 (0)	...
TV4	2000 (0)	2 (1)	1 (1)	...
...	...	...	...	...

Table 1: Spreadsheet of tested TV sets

Obviously, important informations are getting lost by the usage of classic Boolean logic. Thus, the provided result items are not distinguishable at all. For instance, the TV sets `TV2, TV3` and `TV4` return all the *same* positive

query result `true`, in spite of the fact that TV set `TV2` has to be acknowledged as the best choice, when all three subconditions are taken into account.

Next section gives a short overview to the theoretical model behind the quantum-logic query processing and introduces similarity calculus CQQL: *Commuting Quantum Query Language*. Section 3 will sketch the structure of the similarity language family based on CQQL.

2 The Quantum Query Language CQQL

In the following section we present an introduction to the theoretical model behind CQQL. Especially, we will sketch the structure and the evaluation of a CQQL query. A more detailed description of theory can be found in [1].

In general, CQQL enables the logic-based construction of queries from traditional Boolean and similarity conditions. The underlying idea is to apply the theory of vector spaces, also known from quantum mechanics and quantum logic, for query processing. Table 2 gives the correspondences between query processing concepts and the adapted vector space model of CQQL.

query processing		**vector space model of CQQL**	
query system	-	vector space	**H**
tuple to be queried	t_i	vector	$v[t_i]$
query	q	vector subspace	$vs[q]$
evaluation	$eval(t_i, q)$	squared cosine of the angle between $v[t_i]$ and $vs[q]$	$cos^2(\alpha(v[t_i], vs[q]))$

Table 2: Correspondences between query processing and the model of CQQL

Before we go in more detail, we want to summarise the basic idea of evaluating a given tuple t_i against a given CQQL query q. We start by considering a vector space **H** containing all encoded elements of the query processing (see Table 2). All attribute values of a tuple t_i are embodied by the direction of a normalized tuple vector $v[t_i]$. The query q itself corresponds to a vector subspace $vs[q]$ located in **H**. The position and the expansion of the subspace $vs[q]$ bijectively correlates to the semantic of q. To distinguish the subspace $vs[q]$ from the containing vector space **H** we will denote $vs[q]$ as *query space*.

Furthermore, the evaluation result of a tuple t_i against the query q is determined by the minimal angle, denoted as $\alpha(v[t_i], vs[q])$, between the tuple vector $v[t_i]$ and the query space $vs[q]$ in **H**. The squared cosine of this angle is a value out of the interval $[0, 1]$ and can therefore be interpreted as a similarity measure as well as a score-value. If the tuple vector belongs to the query space, i.e. $\alpha(v[t_i], vs[q]) = 0°$, then we interpret the query outcome as a complete match: $cos^2(0°) = 1$. Contrarily, a right angle of 90° between $v[t_i]$ and $vs[q]$ leads to a complete mismatch: $cos^2(90°) = 0$.

Based on this main idea we want to further discuss the two central issues: (1) the construction of the tuple vector $v[t_i]$ and the query space $vs[q]$ and (2) the fast computation of the evaluation result as $cos^2(\alpha(v[t_i], vs[q]))$. To study these both topics we assume the following example query, which is associated with the already known TV set scenario:

$$q = \{(name, status, handling, \ldots) \mid TV(name, status, handling, \ldots) \wedge (status = a \vee status = o) \wedge handling \approx 1\}$$

The domain of the attribute *status* contains the three values *available*, *sold* and *ordered*, whereby the underlined abbreviations are used for the sake of convenience, i.e. $Dom(status) = \{a, s, o\}$. Thus, the query determines all TV sets, which are available or ordered and own a handling mark as good as possible. This query will be validated against the two tuple $t_1 = (TV1, o, 3, \ldots)$ and $t_2 = (TV2, s, 2, \ldots)$.

2.1 Construction of the tuple vector $v[t_i]$ and the query space $vs[q]$

To construct the elements $v[t_i]$ and $vs[q]$ we employ a typical bottom-up strategy based on the logical composition of q. Thus, atomic conditions, also called *predicates*, can be considered as smallest evaluable entities, which are getting combined by the logical connectors $\wedge, \vee$ and $\neg$ to form the final query q. We will exploit this construction principle (going from single predicates over combined subconditions to the final query) to deploy the intended elements.

According to their semantic atomic conditions can be divided in two different main types: Boolean and similarity predicates. For instance, the introduced example query includes three atomic conditions '$handling \approx 1$', '$status = a$' and '$status = s$', whereby the first one can be classified as a similarity predicate and the last two conditions are typical Boolean predicates. The left relation predicate $TV(name, status, handling, \ldots)$ is also a Boolean predicate, but for the discussion of this special case we refer to [1]. At the first construction step we set up a *separate* vector space $\mathbf{H_{p_j}}$ for each Boolean and similarity predicate p_j.

These single vector spaces possess the character of basic modules for the construction of the final vector space $\mathbf{H}$. This behaviour is similar to the structural meaning which is owned by predicates according to the query q. In the following we simple name this kind of a separate vector space as a *basic module*.

There are two information entities, which have to be encoded in such a basic module. On one hand, we have the value of the queried tuple attribute. For example, taking the predicate '$status = a$' we must encode the value o for the attribute $status$ of t_1, which has to be tested against the predicate condition: Is o equal to a? So, the value o forms the tuple vector $v[t_1^{status}]$. On the other hand, the comparison constant of the predicate condition, e.g. the constant 1 for $handling \approx 1$, must be integrated into $\mathbf{H_{p_j}}$ as the query space $vs[handling \approx 1]$.

The applied method for the encoding directly depends on the predicate type, whereby several implications must be considered. Primarily, the encoding must guarantee a valid outcome for the minimal angle between $v[t_i^{attr}]$ and $vs[p_j]$. So, the angle $\alpha(v[t_i^{attr}], vs[p_j])$ has to be either $0°$ *or* $90°$ for a Boolean predicate. In contrast, for a similarity predicate the angle $\alpha(v[t_i^{attr}], vs[p_j])$ can be evaluated as a value *between* $0°$ *and* $90°$. These constraints determine the structure of a basic module $\mathbf{H_{p_j}}$ as well the integration of $v[t_i^{attr}]$ and $vs(p_j)$ into $\mathbf{H_{p_j}}$.

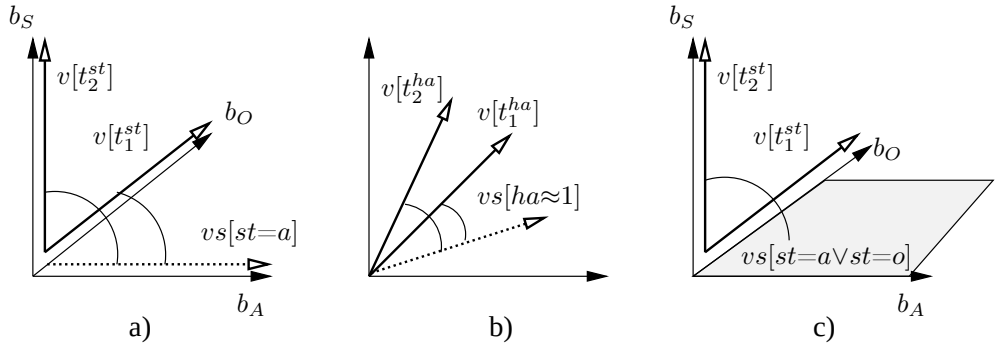


Figure 1: Basic modules

Basic modules for Boolean predicates For Boolean predicates each domain value of the queried attribute constitutes a orthonormal basis vector for $\mathbf{H_{p_j}}$. So, for instance a 3-dimensional basis module is built for the Boolean predicate '$status = a$', whereby the basic vectors b_A, b_S and b_O represent the domain values available, sold and ordered.

To encode the attribute value of t_i^{attr} and the condition constant of p_j we map the corresponding elements $v[t_i^{attr}]$ and $vs[p_j]$ to basis vectors embodying the same domain value. Figure 1a) depicts the basic module for the Boolean predicate '$status = a$' including the query space $vs[st = a]$, and the tuple vectors $v[t_1^{st}]$ and $v[t_2^{st}]$. Please notice that in a basic module the query space $vs[p_i]$ only spans one dimension and is therefore equivalent to a single vector. By studying

Figure 1a) we can confirm that all angles between tuple vectors and query spaces must be either $0°$ or $90°$ depending on the considered attribute values and condition constants. In this example both attribute values o (for t_1) and s (for t_2) leads to a mismatch: $\alpha(v[t_i^{st}], vs[st = a]) = 90°$ for $i = 1, 2$.

Basic modules for similarity predicates A basic module for an arbitrary similarity predicate has always two dimensions. So, the attribute value and the condition constant must be represented by non-orthogonal vectors embedded in the 2-dimensional vector space $\mathbf{H_{p_j}}$. In Figure 1b) the basic module for the similarity predicate '$handling \approx 1$' is given. We conclude that the angles between the attribute vectors $v[t_1^{ha}]$, $v[_2^{ha}]$ and the query space $vs[ha \approx 1]$ express the similarity between these elements in an interval from 0 ($cos^2(90°)$) to 1 ($cos^2(0°)$).

Combining basic modules For the combining of basic modules to the final vector space $\mathbf{H}$ we use the tensor product of two vector spaces [1]. By the usage of this algebraic operation two vector spaces $\mathbf{H_{p_j}}$ and $\mathbf{H_{p_k}}$ are getting entangled in a new generated vector space: $\mathbf{H_{p_j}} \otimes \mathbf{H_{p_k}} = \mathbf{H_{c_l}}$. The dimensionality of the product space $\mathbf{H_{c_l}}$ is in general higher than the dimensionalities of both input spaces.

The attribute tuples and the query spaces within $\mathbf{H_{p_j}}$ and $\mathbf{H_{p_k}}$ are also getting transfered into the product vector space, whereby the two attribute vectors merge to a single multi-attribute vector and the query spaces express a combined condition.

The specific combining operation for query spaces corresponds to the logical operators $\wedge, \vee$ and $\neg$ of the underlying formula q. For example, two query spaces $vs[c_1]$ and $vs[c_2]$ are getting intersected, when the respective subconditions c_1 and c_2 are connected conjunctively. Following operations for the merging of query spaces are defined:

$$\begin{aligned} vs(c_1 \wedge c_2) &\stackrel{def}{=} vs(c_1) \cap vs(c_2), \\ vs(c_1 \vee c_2) &\stackrel{def}{=} \text{closure}(vs(c_1) \cup vs(c_2)), \\ vs(\neg c_1) &\stackrel{def}{=} \mathbf{H_{c_l}} \setminus vs(c_1). \end{aligned}$$

The closure operation generates the set of all possible vector linear combinations. By applying these rules the final vector space $\mathbf{H}$, the multi-attribute tuple vector $v[t_i]$ and the combined query space $vs[q]$ can be constructed recursively.

The Figure 1c) gives an idea for the combination of query spaces. The condition '$status = a \vee status = o$' generates a combined query space as a plane spanned by the vectors b_A and b_O. Since the considered condition only queries the attribute $status$, the query space $vs[st = a \vee st = o]$ is still placed in a single basic module instead of a product space. The measurement of the angle between $v[t_1^{st}]$ and $vs[st = a \vee st = o]$ returns $0°$. It correlates to the fact that $t_1^{st} = o$ fulfill the subcondition '$st = a \vee st = o$'. On the other side, there is an angle of $90°$ between $vs[t_2^{st}]$ and the query space $vs[st = a \vee st = o]$, which is affirming a mismatch between $t_2^{st} = s$ and '$st = a \vee st = o$'.

2.2 Fast computation of $cos^2(\alpha(v[t_i^{attr}], vs[p]))$

Generally, the squared cosine of the minimum angle between $v[t_i]$ and $vs[q]$ can be computed by using a set of basis vectors for the constructed query space $vs[q]$. From a computational view point such a method is quite inefficient, because in general the cardinality of a basis vector set increases tremendously.

An alternative method is developed in [1]. It allows to evaluate a tuple t_i against a complex CQQL query q by using of simple arithmetic operations recursively:

$$
\begin{array}{lcl}
eval(t_i, p) & := & \varphi_{attr}(t_i^{attr}, p), \\
eval(t_i, c_1 \wedge c_2) & := & eval(t_i, c_1) * eval(t_i, c_2), \\
eval(t_i, c_1 \vee c_2) & := & eval(t_i, c_1) + eval(t_i, c_2) - eval(t_i, c_1 \wedge c_2), \\
eval(t_i, \neg c) & := & 1 - eval(t_i, c),
\end{array}
$$

whereby p is an atomic condition (predicate) and c, c_1 and c_2 are arbitrary subconditions. The function $\varphi_{attr}(t_i^{attr}, p)$ returns $cos^2(\alpha(v[t_i^{attr}], vs[p]))$ for the predicate p and the respective attribute value of t_i.

For the correct application of the defined operations a specific *syntactical form* of the CQQL query q is needed. Since our language obeys the same transformation rules as held by the Boolean logic we can convert every CQQL query into the required syntactical form. An algorithm performing this transformation is presented in [1].

3 Language family

Since Codd has introduced the relational data model in the 70s a lot of relational query languages have been proposed. Especially, the relational calculuses, the relational algebra and the database query language SQL have grown a strong theoretical and practical significance. Under certain circumstances all three languages can be considered as equivalent expressive. However, they distinguish from each other by their areas of application and their user groups.

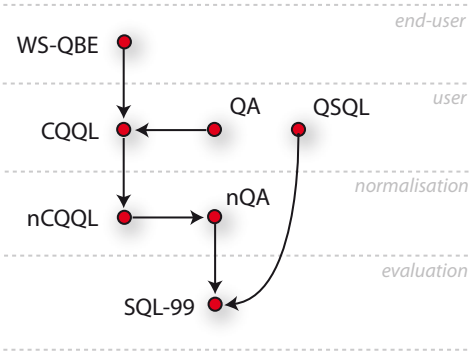


Figure 2: Similarity query languages

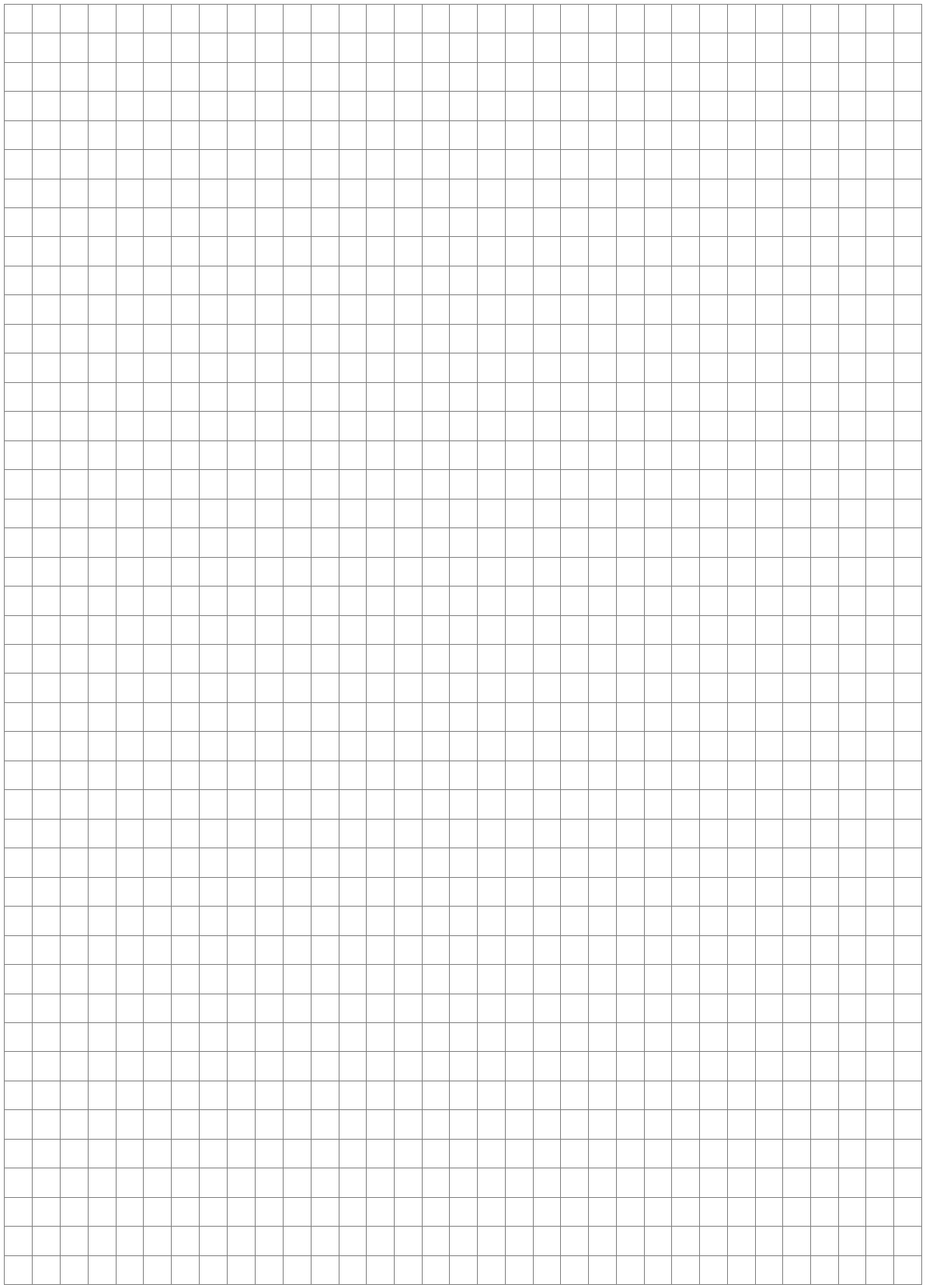
In the previous section we presented the basic ideas of our query processing approach by an informal introduction to the calculus language CQQL. The underlying concepts and basic constraints, which have been derived from the vector space model, are encapsulated in the CQQL semantic.

The challenge is now to incorporate these principles into further relational query languages. For this purpose a whole quantum logic-based language family has been developed. To preserve the compatibility to existing object-relational data base systems we also evolved several mappings between the languages (Fig. 3), which are all ending up to the SQL-99 standard. Following languages belongs to our group of similarity languages:

- WS-QBE: QBE-based extension for the input of multimedia and similarity conditions
- CQQL: extension of the relational domain calculus
- QA: extension of the relational algebra
- QSQL: special SQL dialect based on SQL-92
- nCQQL: normalised CQQL-version
- nQA: normalised QA-version

References

[1] Schmitt, Ingo: *QQL: A DB&IR Query Language*. The VLDB Journal, 17(1):39–56, 2008.



Towards an Energy Aware DBMS – Energy Consumptions of Sorting and Join Algorithms

Hagen Höpfner
International University in Germany
Campus 3; 76646 Bruchsal; Germany
hoepfner@acm.org

Christian Bunse
International University in Germany
Campus 3; 76646 Bruchsal; Germany
chistian.bunse@i-u.de

Abstract

Database management systems comprise various algorithms for efficiently retrieving and managing data. Typically, algorithm efficiency or performance is correlated with execution speed. However, the uptime of battery-powered mobile- and embedded systems strongly depends on the energy consumption of the involved components. This paper reports our results concerning the energy consumption of different implementations of sorting and join algorithms. We demonstrate that high performance algorithms often require more energy than slower ones. Furthermore, we show that dynamically exchanging algorithms at runtime results in a better throughput.

1 Introduction and Motivation

Database management systems (DBMS) are software systems that are in widespread use for managing data independent of applications and underlying hardware. Hence, application developers can utilize DBMS in order to efficiently store and retrieve data without knowing their exact implementation (aka. black-box view). DBMS provide various access strategies (indexes) and retrieval algorithms for handling and manipulating data. These are typically optimized regarding their execution performance. Most existing DBMS are designed to run on desktop computers and powerful servers. However, due to the recent developments in the area of embedded and mobile devices, the need for lightweight and mobile DBMS appeared. In contrast to other (more powerful) platforms such devices are often battery powered. Therefore, their uptime strongly depends on the efficient usage of the resource energy. Whereas energy optimization regarding hardware (sleep mode, speed reduction, etc.) is state of the art, only little research has been performed regarding the energy consumption of software. In this paper we present first ideas on how to minimize the energy consumption of DBMS. Since, DBMS are complex software systems, we focus on the energy consumption of basic algorithms such as sort and join operations.

The remainder of the paper is structured as follows: Section 2 introduces the examined algorithms. Section 3 briefly presents the evaluation environment and discusses the underlying measurement theory. Section 4 discusses the results obtained in various empirical studies. Section 5 contains first ideas on how to adapt algorithm usage to reduce energy consumptions. Section 6 concludes the paper and gives an outlook on future research. Due to the strict space limitations we decided not to include a related work section into this paper but refer to [2].

2 Sorting and Join Algorithms

A first step towards the development of lightweight and energy-efficient DBMS is the analysis of the basic or ”foundation level” algorithms regarding their energy consumption. This paper presents first, preliminary results for sort and join algorithms. In the following we briefly describe these algorithms based on [5] (sort) and [3] (join):

Bubblesort belongs to the family of comparison sorting. It works by repeatedly stepping through the list to be sorted, comparing two items at a time and swapping them if they are in the wrong order. Bubblesort has a worst-case complexity of $O(n^2)$ and best case of $O(n)$. Its memory complexity is $O(1)$.

Heapsort is comparison-based and part of the Selectionsort family. In practice it is slower on most machines than an efficient implementation of Quicksort, but it has the advantage of a worst-case complexity of $O(n \log n)$.

Insertionsort is a “naive” sorting algorithm belonging to the family of comparison sorting, too. It has a worst case complexity of $O(n^2)$, but it is known to be efficient on substantially sorted data sets. Its average complexity is $O(n^2/4)$ and $O(n)$ in the best case. As an in-place algorithm it requires a constant amount of memory space.

Mergesort belongs to the family of comparison-based sorting. It has an average and worst case complexity of $O(n \log n)$. Mergesort requires three times the memory of in-place algorithms such as Insertionsort.

Quicksort [4] belongs to the family of exchange sorting. On average, Quicksort makes $O(n \log n)$ comparisons to sort n items, but in its worst case it requires $O(n^2)$ comparisons. Typically, Quicksort is regarded as one of the most efficient algorithms. Its memory usage depends on factors such as choosing the right Pivot-Element. On average, having a recursion depth of $O(\log n)$, the memory complexity of Quicksort is $O(\log n)$ as well.

Selectionsort belongs to the family of in-place comparison sorting. It typically searches for the minimum value, exchanges it with the value in the first position and repeats the first two steps for the remaining list. On average Selectionsort has a $O(n^2)$ complexity that makes it inefficient on large lists. Selectionsort typically outperforms Bubblesort but is generally outperformed by Insertionsort.

Shakersort [1] is a variant of Shellsort that compares each adjacent pair of items in a list in turn, swapping them if necessary, and alternately passes through the list from the beginning to the end then from the end to the beginning. It stops when a pass does not executes any swaps. Its complexity is $O(n^2)$ for arbitrary data, but approaches $O(n)$ if the list is substantially sorted at the beginning.

Shellsort is a generalization of Insertionsort. The algorithm belongs to the family of in-place sorting but is regarded to be unstable. The algorithm performs $O(n^2)$ comparisons and exchanges in the worst case, but can be improved to $O(n \log_2 n)$. This is worse than the optimal comparison sorts, which are $O(n \log n)$. Shellsort improves Insertionsort by comparing elements separated by a gap of several positions. This lets an element take "bigger steps" toward its expected position. Multiple passes over the data are taken with smaller and smaller gap sizes. The last step of Shell sort is a plain Insertionsort, but by then, the array of data almost sorted.

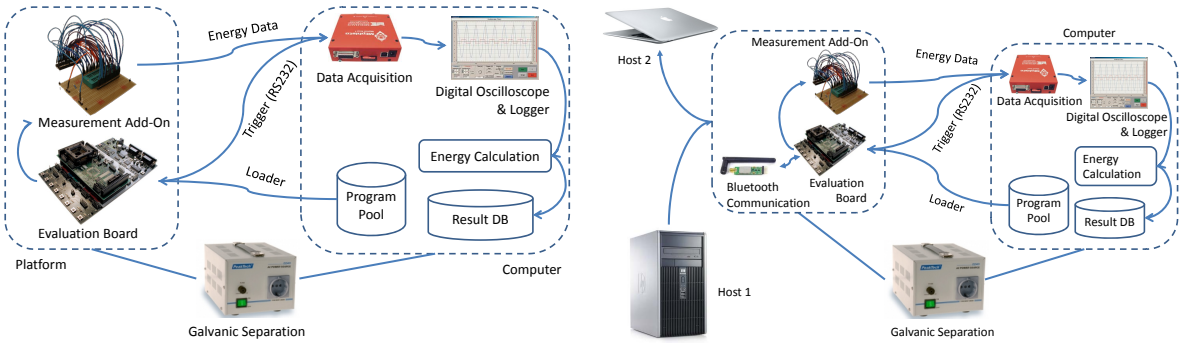
Nested-Loop-Join (NLJ) compares all tuples of one relation R to all tuples of the other relation S by using a nested loop. The complexity is $O(|R| \cdot |S|)$, where $|R|$ and $|S|$ represent the number of tuples per relation.

Sort-Merge-Join (SMJ) requires that both relations must be sorted on the joint attributes. If they are not physically or implicitly (index) sorted, an external sorting is performed first. The merge part then scans the input relations alternately using the monotony of sorted relations. In the worst case (all tuple have the same join attribute value), the complexity is $O(|R| \cdot |S|)$. In the best case only the smaller relation must be scanned ($O(\min(|R|, |S|))$.

Hash-Join (HJ) uses a hash function to find matching tuples. In a partitioning phase, all records of the smaller relation are hashed using the join attributes. In the subsequent probing phase the hash values for the tuples of the other relation are calculated. Hence, join candidates are hashed to the same bucket. In the worst case where only one hash bucket exists, the efficiency is equal to the Nested-Loop-Join. In the best case, assuming constant access time for values of the hash table, the complexity is $O(|R| + |S|)$.

3 Measurement Environment

The researched algorithms vary in their memory and CPU (e.g., no. of cycles) usage. In order to explicitly measure the energy consumption of single algorithm executions, we used a specific evaluation platform (Figure 1). In detail, the algorithms where executed on an AVR micro controller (i.e., ATMega 128 based on a STK501 board). Every program (algorithm) sent a TTL level signal [6] at the start and end of its run in order to trigger measurement and logging by a digital oscilloscope. The collected data was externally processed in order to calculate the consumed energy values and stored.



(a) without communication interface (ES_1)

(b) including communication via Bluetooth (ES_2)

Figure 1: Experimental Setup

Unfortunately, energy consumption cannot be directly measured since it is a function over time. However, using a sense resistor with a known value $R_{SENSE} = 100\Omega$ and a stabilized voltage $U = 4.93V$, we could calculate the needed energy or Joule value as follows: We measured the voltage drop U_{SENSE} at the sense resistor embedded

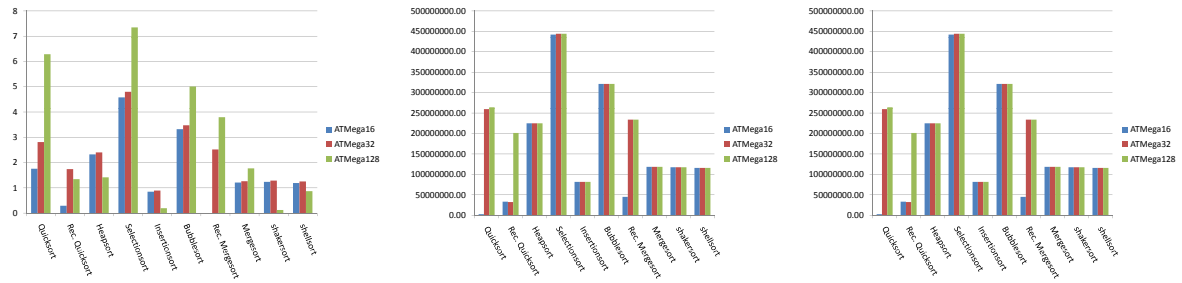
into an add-on board[1] and then calculated the current power consumed by the processor core. Using Kirchhoff's, Ohm's and some mathematical laws the resulting formula is $E = \frac{1}{100\Omega}\Delta t \cdot \sum_{n=0}^{\frac{t}{\Delta t}} 4.93V \cdot U_{SENSE}(n \cdot \Delta t) - U_{SENSE}^2(n \cdot \Delta t)$, whereby n represents the number of collected samples per second and $\Delta t = \frac{1s}{n}$ the interval for one sample. For details on this calculation we refer to [2].

Figure 1(a) illustrates the measurement setup without any network communications. Especially mobile devices very often use wireless networks to communicate with an information service provider. Therefore, we decided to extend the basic environment by a Bluetooth interface (see Figure 1(b)). The embedded node now wirelessly receives data-sets, processes them and sends the sorted set to another recipient.

4 Experimental Results

This section discusses the results of different experiments to investigate the energy consumption of sorting and join algorithms. All figures in this section use accumulated, non-normalized values. The following results are sums of measurement results of random, sorted and reverse-sorted data and are accumulated for 1,000 cycles.

We first examined whether software energy consumption, as widely believed, is strongly correlated to software performance. Therefore, we executed the algorithms on different processors of the AVR processor family, whereby the consumed energy (Figure 2(a)), the execution time (Figure 2(b)) as well as the number of cycles (Figure 2(c)) were measured. The latter two were obtained by using the AVR simulator of AVRStudio. "Empty" bars represent missing results. It was, e.g., not possible to use the recursive Mergesort for sorting 1,000 integer values on the ATMega16 since the system ran out of memory.



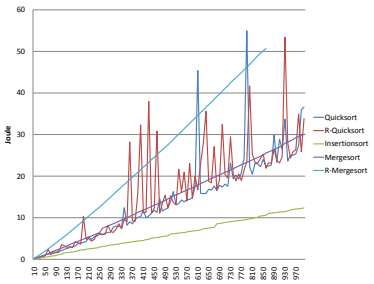
(a) Energy Consumption in Joule (b) Execution Time in ms (c) Used CPU Cycles

Figure 2: Experimental Results (sorting)

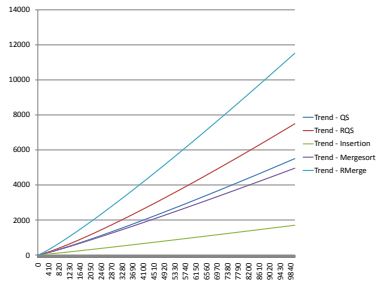
Initially, the results regarding the energy consumption of different sorting algorithms reveal that, in contrast to the initial assumption, sorting algorithms such as Insertionsort ($O(n^2)$) require significant less energy than high-performance algorithms such as Quicksort ($O(n \log n)$). The results show that this observation is valid and visible across different processors of the same family. Furthermore, energy consumption grows with the inbuilt flash memory size (15 KByte, 32 KByte, and 128 KByte). Interestingly, recursive Quicksort outperforms Insertionsort when running on the ATMega16 but is clearly behind at all other platforms. The reason is that the figure represents accumulated values (random, sorted, reverse sorted), and recursive Quicksort failed (out of memory) to sort the sorted and reverse sorted data. Therefore, the presented value is much lower.

The 2nd experiment series (500 execution cycles) had two goals: First, to measure the energy consumption of sorting algorithms for growing data sizes. Second, to evaluate the impact of using external memory on energy consumption. Therefore, we concentrated on Quicksort (standard and refined), Mergesort (standard and refined), and Insertionsort with respect to random, sorted and reverse sorted data and for increasing lengths of data (0 to 1,000 elements). The obtained measurement results (Figure 3(a)) in general confirm the results of the previous experiment series by showing that Insertionsort consumes significantly less energy than other algorithms, although it is slower. In fact, sorting 1,000 randomized elements with Insertionsort took 71.3 ms, whereas Quicksort needed 8.8 ms. Regarding the second goal, we extended the microprocessor's inbuilt SRAM memory from 4 to 132 KByte using an external memory chip. While comparing Figure 2 and Figure 3 it becomes obvious that using external memory requires significantly more energy. For Insertionsort, e.g., the energy consumption for sorting 1,000 random elements raised from 0.03 to 4.11 Joule. The difference cannot be explained by the standard energy the additional chip requires since the differences between both curves strongly diverge with a growing data size. We assume that this is caused by moving data to/from external memory and addressing/managing these additional memory cells.

[1] used to ease the change of processors, provide measurement points, etc.



(a) Comparison of Sums (rand, sort, rev)



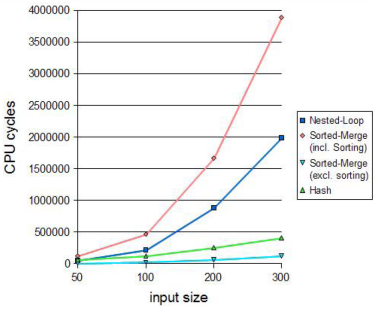
(b) Trends

Algorithm	trend function
Quicksort	$F(n) = n^{1.1388} \cdot 0.1533$
RQuicksort	$F(n) = n^{1.1771} \cdot 0.1467$
Insertionsort	$F(n) = n^{1.0679} \cdot 0.09121467$
Mergesort	$F(n) = n^{1.1035} \cdot 0.1912$
RMergesort	$F(n) = n^{1.1384} \cdot 0.3221$
Heapsort	$F(n) = n \cdot 0.2324 + 0.0286$
Shellsort	$F(n) = n^2 \cdot 0.0071 + n \cdot 0.0047 + 0.0939$
Selectionsort	$F(n) = n^2 \cdot 0.013 - n \cdot 0.0236 + 0.1908$

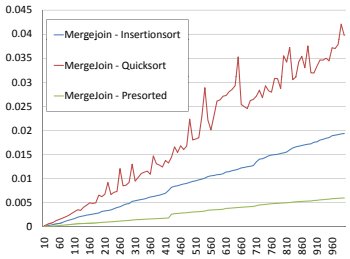
(c) Trend functions

Figure 3: Experimental Results – 2nd Measurement Series

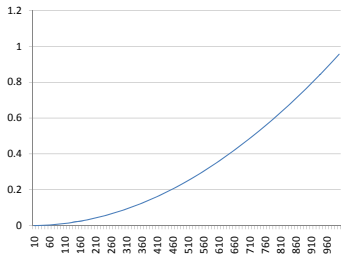
The differences between algorithms become clearer by watching the interpolated trend functions in Figures 3(b) and 3(c). Here, n is the number of processed data items and the R^2 value that represents the goodness of fit was 1.



(a) Various Join Implementations



(b) Merge-Join



(c) Nested-Loop-Join

Figure 4: Energy consumption comparison – normalized to one execution

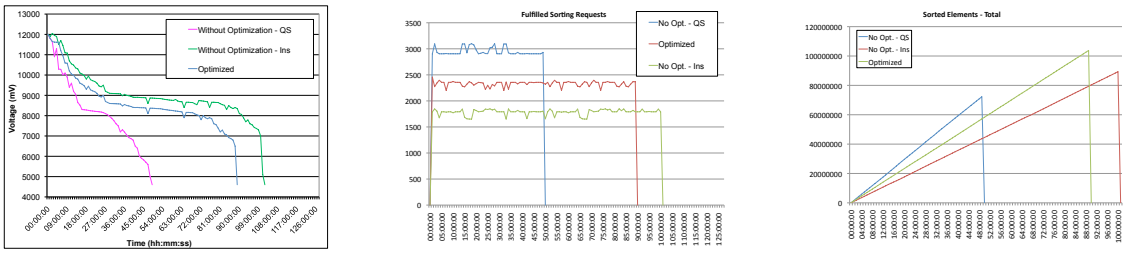
The final experimental run investigated the energy consumption of standard join operations. In detail, the joins were executed on the Atmel ATMega128 MCU. Each time two integer arrays were joined, whereas the join condition was the equality of both attributes. Both arrays contained random numbers between 0 and 100, generated with C's `rand()` function. As the generated number varied and such the execution of the different algorithms, the performance measurement was perform several times. The number of execution cycles each algorithm required to finish its task was measured. Note that SMJ used Insertionsort. Figure 4(a) shows results for array sizes of 50, 100, 200 and 300. For a small input size NLJ is most efficient, but the complexity rapidly increases as input gets bigger. By contrast, the number of HJ cycles takes grow only almost linearly, making it five times faster for input size of 300. The SMJ algorithms has two facets: for presorted data it is by far the fastest algorithm of the three, regardless of the input size. If the data has to be sorted first, the performance of the overall operation is highly slowed down by the sorting algorithm. Hence, HJ seems to be the most efficient join algorithm for unsorted input data.

From our sorting algorithm experiments we learned, that memory usage is important for energy consumptions. Whereas NLJ and SMJ (depending on the sorting algorithm used) need almost no additional memory space, HJ uses a considerable amount of memory to store the hash table. Thus, it is justifiable to assume that if considering the energy consumption instead of the performance, the outcome will be appreciably different. In order to determine the amount of energy in Joule, we measured the SMJ algorithm using Insertionsort and Quicksort. The results in Figure 4(b) conform to our previous finding that Insertionsort consumes much less energy in comparison to Quicksort. The energy consumption of a NLJ see Figure 4(c) grows quadratic. Thus, although NLJ require less execution cycles than MJ, its energy consumption is significantly higher. This supports our assumption that the energy consumption related to the execution of a specific algorithm mostly depends on its memory requirements and that the algorithms complexity class (i.e. number of execution cycles) plays only a minor role.

5 Optimized Algorithm Usage

The optimization of algorithmic energy-consumption requires a model (i.e., cost model) that can be used to predict the cost (i.e., *J* or energy) for executing a specific algorithm on a specific input set. Based on the measurement results concerning the energy consumption of sorting algorithms we therefore extrapolated trend functions that

calculate an estimation of the required energy for 1,000 executions [2] of an algorithm, based on the input size n. Since our goal was to find the optimal balance between sorting performance and energy consumption it is not sufficient to solely use the trend functions as cost function. Due to the nearly linear nature of the trend function the result would always indicate Insertionsort as the most energy-efficient algorithm but neglect algorithmic complexity and performance. Therefore we applied the following strategy: 1) By using the size n of the set as an input the energy-related costs for all algorithms are calculated and stored. 2) The minimum result and thus the most energy-efficient algorithm is identified. 3) Based on the algorithmic complexity, the minimum value is compared to those values that are related to algorithms of "lower" complexity classes. 4) If the difference between the energy requests is below a predefined threshold or delta the "faster" algorithm is chosen.



(a) Battery Lifetime (b) Request Performance (c) Total Number of Sorted Elements

Figure 5: Adaptive Algorithm Selection

The measurements are based on the networked experimental setup (cf. Section 3). A look at the battery level V over time supports the initial assumption that the uptime of a systems is directly correlates with the energy consumption related to the executed software system. However, a closer look at Figure 5(a) shows that a non-adaptive approach (i.e., using a fixed algorithm) either results in an excellent or a poor energy efficiency. Interestingly, the results for the adaptive version are close to those of the non-optimized Insertionsort variant. Figure 5(b) supports the initial assumption concerning the trade-off between energy efficiency and performance. Fast variants like Quicksort handle more sorting requests in a shorter period of time but result in a very limited V. Energy-efficient variants like Insertionsort result in an optimal V but handle significantly less sorting requests. Only adaptive systems provide a good balance of energy-efficiency and performance. This is also supported by Figure 5(c) that shows the total number of elements that were sorted over time.

6 Summary, Conclusions and Outlook

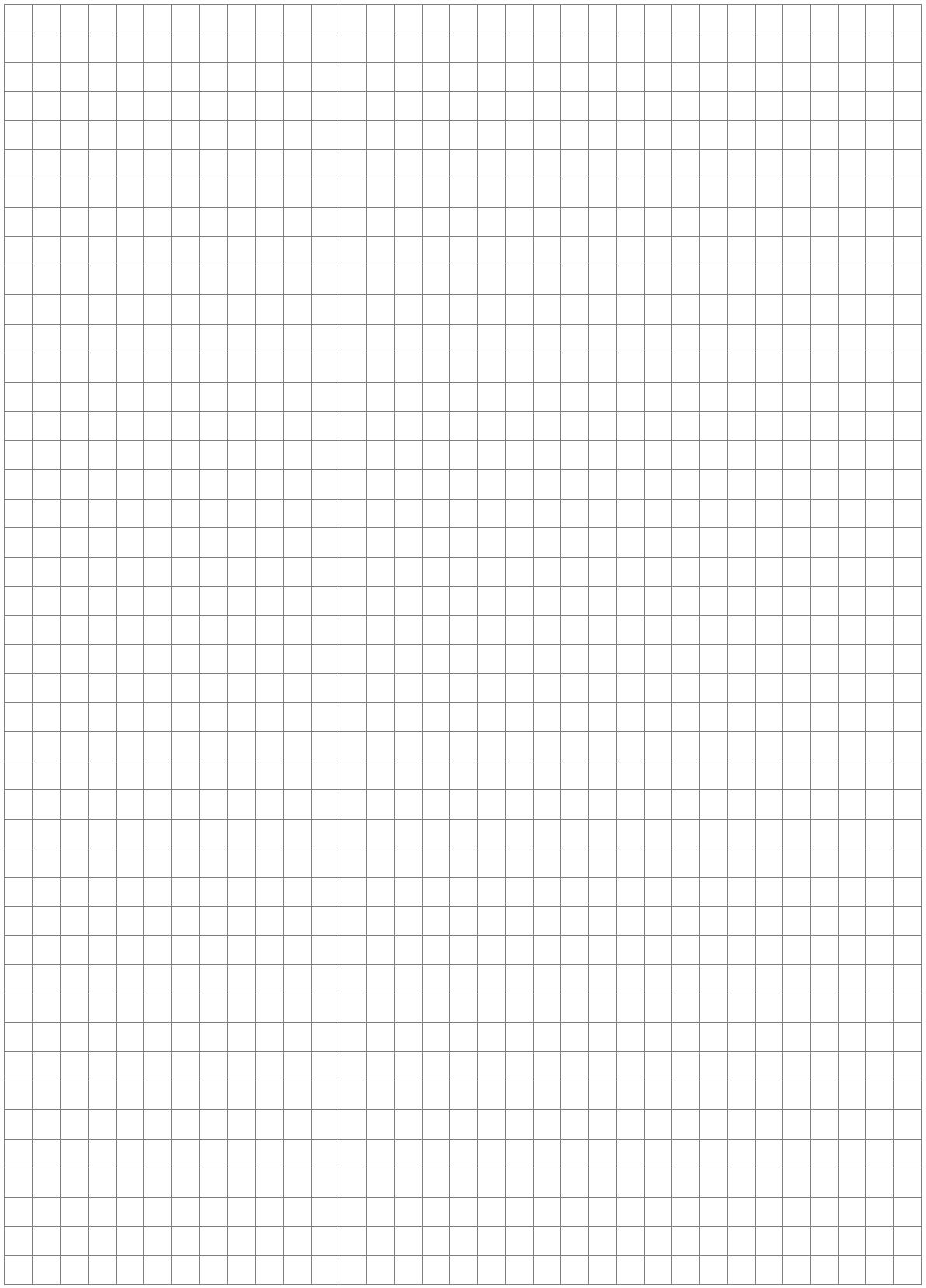
We presented first results towards realizing energy aware database management systems. We concentrated on sorting and join algorithms as those are essential for query processing. We introduced our measurement setup and discussed first, preliminary results. We highlighted that memory intensive implementations consume more energy then CPU intensive ones. Interestingly, we found out that energy consumption is not solely correlated with the complexity class (performance) of an algorithm. Furthermore, we discussed first ideas on how to optimize the algorithm usage based on trend functions reflecting the energy consumption of certain algorithm implementations.

The next step on our agenda is to replicate the experiments with other platforms (i.e., PIC, ARM, and PSoC) in order to get a more generalized "cost model". Furthermore, we plan to examine other DBMS algorithms for query processing (set operations, projection, selection, etc.) and for indexing data (B-Tree, tries, etc.). Afterwards, we plan to define an overall optimization strategy based on the user requirements (i.e., fast results vs. long up-time).

References

[1] B. Brejová. Analyzing variants of Shellsort. *Information Processing Letters*, 79(5):223–227, 2001.
[2] C. Bunse, H. Höpfner, E. Mansour, and S. Roychoudhury. Exploring the Energy Consumption of Data Sorting Algorithms in Embedded and Mobile Environments. In *ROSOC-M 2009 Proceedings*, 2009. forthcoming.
[3] R. Elmasri and S. B. Navathe. *Fundamentals of Database Systems*. Addison Wesley, 5th edition, 2006.
[4] C. A. R. Hoare. Quicksort. *Computer Journal*, 5(1):10–15, 1962.
[5] R. Lafore. *Data Structures and Algorithms in Java*. SAMS Publishing, Indianapolis, Indiana, USA, 2nd edition, 2002.
[6] D. E. Lancaster. *TTL Cookbook*. Sams, May 1974.

[2] trend functions are build upon measurements for 1,000 execution cycles



A Self-Adaptive Insert Strategy for Content-Based Multidimensional Database Storage

Sebastian Leuoth, Wolfgang Benn

Department of Computer Science
Chemnitz University of Technology
09107 Chemnitz, Germany

{lese, benn}@cs.tu-chemnitz.de

Abstract

In this paper, we present the current development progress of our dynamic insert strategy based on the Intelligent Cluster Index (ICIx), which is a new type of multidimensional database storage. Opposite to purely value-based interval methods, ICIx performs a semantic clustering of the data objects in a database and keeps the clustering results as basis for storing in a special tree structure (V-Tree). Our paper aims at the quality problem caused by a trade-off between the static clustering that results from the initial training data set and the continuous insertion of data into a database which requires a continuous classification.
The strategy that we propose will solve this problem through a continuous and efficient content-based growing of the initially static clustering. We have developed an additional structure — the C-Tree — which stores the knowlege of the hierarchical clustering component, i.e. hierarchical Growing Neural Gas (GNG), for unsupervised content based classification.
In contrast to other methods (e.g. dynamic versions of R-Trees) we use the C-Tree to process the new tuple. Furthermore, we use a Bayesian approach to determine the degree of adaptation of the knowledge base. Using this value, we update the knowlege base and propagate the resulting changes to the V-Tree. As a result, we obtain a continuous content-based growing.

1 Introduction

Since the emergence of non-standard database systems and their applications, the demand for an efficient processing of multidimensional data increases. Important requests are similarity-based queries such as set queries, range queries, and nearest neighbor queries. Traditional database systems can manage large amounts of data very efficiently, as long as they are ordered and selected by unique attributes in a one-dimensional search space. To close this gap, the Intelligent Cluster Index (ICIx)[3] has been developed.

ICIx is a multidimensional indexing and storage method. Its core idea is to combine methods from the field of Artificial Intelligence — in particular nonparametric classification — with those from the field of database accessing. ICIx uses specific Artificial Neural Networks to classify the data objects in a database, which is considered the multidimensional search space itself [7]. This is done initially by a bulk load step, where the hierarchical clusters of semantically similar data objects are constructed. The clustering procedure uses representative parts or (in the worst case) the complete database as training set. Finally, the resulting hierarchical clustering will be exploited and particular cluster-ids with their associated tuple are transferred into a persistent management tree — the so-called V-Tree. This V-Tree is the actual storage structure. It is used for all supported retrieval operations like point and range queries as well as nearest neighbor queries on the basis of a semantic similarity between the requested data objects. Investigations

have shown that our method is characterized by a fast response time and very few I/O operations in comparison to other methods, such as R*-tree, SS-tree or SR-Tree (for detailed experiments see [4]).

Today, several applications have been explored to prove our approach. The most prominent applications of the ICIx are its use as a database storage engine or as a kind of secondary database index in the form of a set-top box. The database storage engine utilizes our method as primary data organization. The usage as a set-top box is characterized by a loose coupling between our system and an arbitrary existing database system. The query is first sent to the ICIx that processes the multidimensional as well as semantic tasks. Afterwards, the query is transformed in a single primary key selection for which the ICIx returns the key values as response.

Currently, we focus on a flexible space allocation. The hierarchical cluster structure depends heavily on the quality of the training set and may be influenced by the ordering of the tuple. This addiction is tolerable mainly for static data, as it appears in data warehouses or data mining applications. In an online environment or in an environment with frequent input, the efficiency depends on the continuous and stable quality of the clustering — with the problem of its relatively static boundaries. Here, the insertion of a lot of new records can result in very large overlapping regions and a degeneration of the index structure — this problem is known in literature as concept drift [10]. Our current solution uses a hierarchical new-learning-process to compensate such degeneration. In the worst case, this can affect the complete data set. Thus, the aim of this paper is to present an idea for an insert strategy which enables the ICIx to do a continuous, knowledge based self-adaption.

The organization of our paper is as follows: In Section 2, we present an overview of related work in multidimensional data partitioning indexing techniques and their insert strategies. The central Section 3 describes our approach to continuous adaptation. Finally, we conclude with a summary and a description of further research.

2 Related Work

Existing approaches propose a variety of solutions for managing multidimensional data efficiently. A detailed review of these methods can be found in [5]. We focus on the field of data partitioning techniques basing on the R-Tree [6]. These methods propose data distribution as the fundamental basis for a partitioning and can be divided in two classes characterized by their creation method.

1. Dynamic versions are build up by inserting tuples one-by-one. The methods differ in their inserting and splitting strategies. The advantage arises from their continually growing feature. The quality of the resulting structure depends heavily on the order of data records. This methods are susceptible to the concept drift.

2. Static versions are build up during a bulk load step. This enables the methods to create information about the data records so that they can use this information to optimize there structure. The disadvantage of this lies in its read-only usage.

Thus, we were looking for a method to merge these two different streams. We are building an initial version of the ICIx using our Growing Neural Gas (GNG) [2] approach and then switch to a continuous inserting strategy. In contrast to other methods, we rest on the knowledge which was generated in the bulk load step. If we used well known techniques like Linear Node Splitting [1] or Branch Grafting [9], we would destroy the contend-based information and return to a simple rule based acting.

3 The Concept of a Self-Adaptive Insert Strategy

Our approach bases on emphasizing the separation in the construction and insertion phase. We build a new component – the C-Tree (C for Cluster or Control). Its task is to build, adapt, and manage the cluster. During the initial bulk load it applies the data records to the GNG component. After the GNG has finished the growing and classification, the C-Tree stores the internal state and all relevant information. In particular the count and position of the neurons or the edges with their weights are saved. In the next step, the C-Tree propagates the classified tuple to the V-Tree which is used to store the data records. Figure 1 shows this process in a simplified structure of our model. The figure emphasizes the partition in data and query processing component.

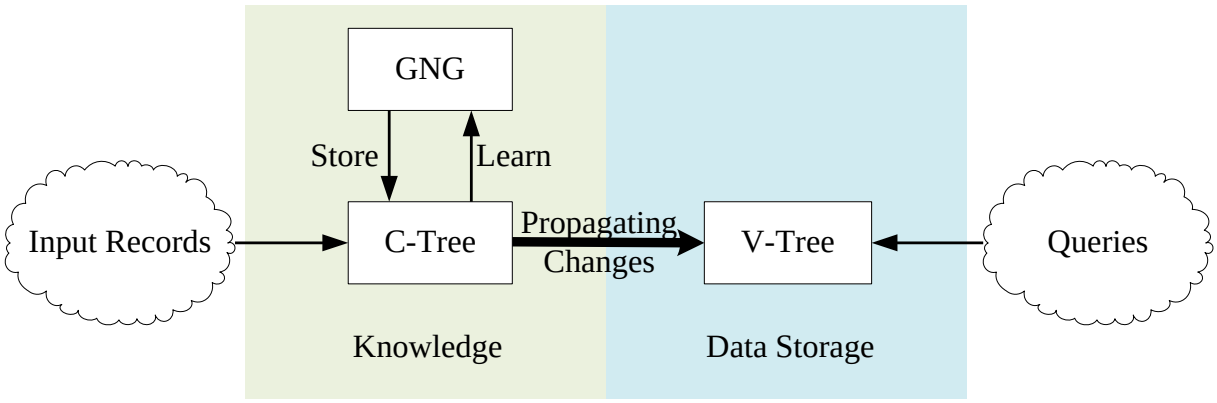


Figure 1: The Figure shows our separation between knowledge and data storage. The growth of the V-Tree is controlled by the C-Tree. Every change in the C-Tree is propagated to the V-Tree.

When a new tuple should be inserted, we can use the C-Tree and its knowledge about the data and their distribution. We use a Bayesian approach to determine the degree of adaptation of the knowledge base. Our insert algorithm (shown in Algorithm 1) starts on top level of the C-Tree and traverses this tree downwards. For each level it uses the neurons to classify the tuple. As in the original GNG procedure, we adapt the neurons. Then we rate the distance between the new tuple and the nearest neuron. If it is greater than a special threshold value, we append a new neuron, if the distance is less we adapt the existent neurons. This strategy corresponds to the Parzen-Windows approach [8]. The size of the volume is determined from the threshold.

The challenge of this method is to handle the resulting changes in a hierarchical environment. If we adapt a neuron on top level, we have to ensure consistency. All classified data records in all following levels still have to be correct. To meet this requirement we use the structure of the C-Tree. Owing to the fact that we have a hierarchy of clusters, we know that the next level again consists of clusters. If we change the position of the neurons in level i, we only have to reclassify the neurons of the all associated levels $i + 1$. There, the adaption effects only a small part of the C-Tree.

Another difficult problem is the overflow handling of nodes in the various levels. As mentioned above, dynamic methods use split strategies to solve this. If we detect an overflow in cluster C on level i, we have to reduce the count of neurons by using an agglomerative clustering method. It selects a group of nearest neurons, e.g. two neurons, and merges it to a new cluster. The resulting cluster will be represented by one new neuron which replaces the others. The position of this new point can be calculated by a centroid or by a center of gravity method which respects the weight of the original neurons. In the next step, we add a new level $i + 1$ into which the

Algorithm 1 Inserting a New Element (`NE`) into the ICIx

```
1: Initialize actual Level of Processing [LP] with root level of the C-Tree
2: Set default value of new Inserting Position [IPos] with blank
3: while [IPos] is blank do
4:   Determine the Minimal Distance [MD] to the Neuron [N] ∈ [LP]
5:   if [MD] ≥ threshold value then
6:     Insert New Neuron [NN]
7:     [IPos] ← New Neuron [NN]
8:   else
9:     Adapt Neuron [N]
10:    if neuron [N] has no child then
11:      Insert NE into value list of neuron [N]
12:      [IPos] ← neuron [N]
13:    else
14:      [LP] ← child level of neuron [N]
15:    end if
16:  end if
17: end while
18: Propagate the changes to the V-Tree
19: Recalculation of the minimum bounding boxes of the V-Tree
```

replaced neurons are moved. The direction of growth of the C-Tree is downwards. If we moved the new neuron up into level $i-1$, we could get two problems. First, in this level another overflow could appear. This could draw up to the root level. If the root level is satisfied, a new root would have to be created. The second problem is the problem of overlapping regions. It is not easy to exclude this possibility. Our proposed direction — down — avoids this problems and the classification of level i will not have to be changed. We only insert one additional level. This situation is shown in Figure 2.

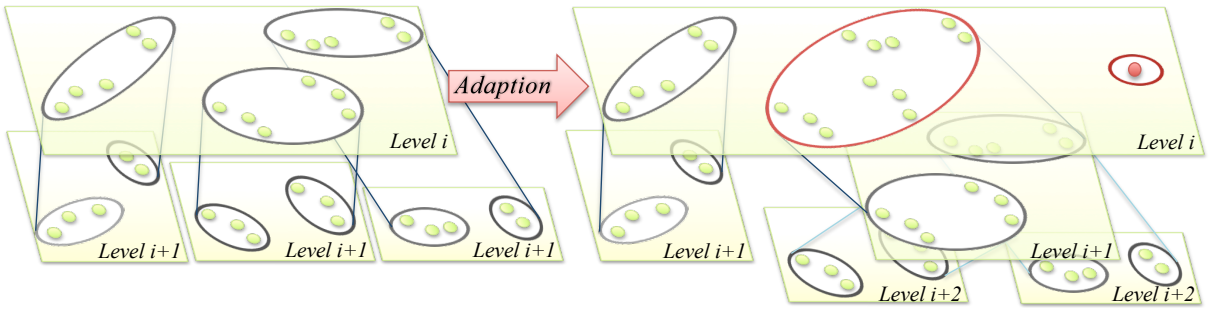


Figure 2: The Figure shows the cluster structure of level i before and after an adaption (highlighted with red). On the right side, a new neuron has been inserted. The number of neurons is limited to 3, so that the two clusters had to be merged.

In all the cases described above, we had to propagate the changes to the V-Tree. However, this operation is not very difficult. We use a link between the clusters of the C-Tree and the corresponding V-Tree nodes to get a fast access. We only have to log the C-Tree changes and finally transmit it to the V-Tree. The only changing effort worth mentioning results from the hierarchical recalculation of the minimum bounding boxes — although this task can occur in every dynamic indexing method.

4 Conclusion and Further Work

We have described a novel self-adaptive insert strategy for content-based multidimensional database storage that rests on a knowledge base. Our strategy introduces a separation between knowledge and data storage. It enables a primarily static method to become a continuous learning method. This is important to reduce the demand for a periodical rebuilding.

In the next step we have to implement the presented algorithm and evaluate our solution. Only then we can compare the different merge, insert and the threshold value strategies by using real and artificial data records. We want to find out the performance of our method and how robust it is against degeneration. In this case, we have to define degeneration and how it can be measured. Another topic is to find out which neurons and classicized tuples are effected by an adaption. We are thinking about a kind of filter which creates a voronoi region. The only neurons that have to be checked are those lying in the same cell.

References

[1] Chuan-Heng Ang and T. C. Tan. New Linear Node Splitting Algorithm for R-trees. In *SSD '97: Proceedings of the 5th International Symposium on Advances in Spatial Databases*, pages 339–349, London, UK, 1997. Springer-Verlag.

[2] Bernd Fritzke. A Growing Neural Gas Network Learns Topologies. In *NIPS*, pages 625–632, 1994.

[3] Sven Gild and Ralf Neubert. Semantische Indexierung mittels dynamische-hierarchischer Neuronaler Netze. Master's thesis, Technische Universität Chemnitz, 1999.

[4] Otmar Görlitz. *Inhaltsorientierte Indexierung auf Basis künstlicher neuronaler Netze.* Shaker, 1st edition, 2005.

[5] Sebastian Leuoth and Wolfgang Benn. Towards SISI – a Self Adaptive Insert Strategy for the Intelligent Cluster Index (ICIx). In *International Conference on Machine Learning and Data Mining MLDM, Leipzig*, 2009.

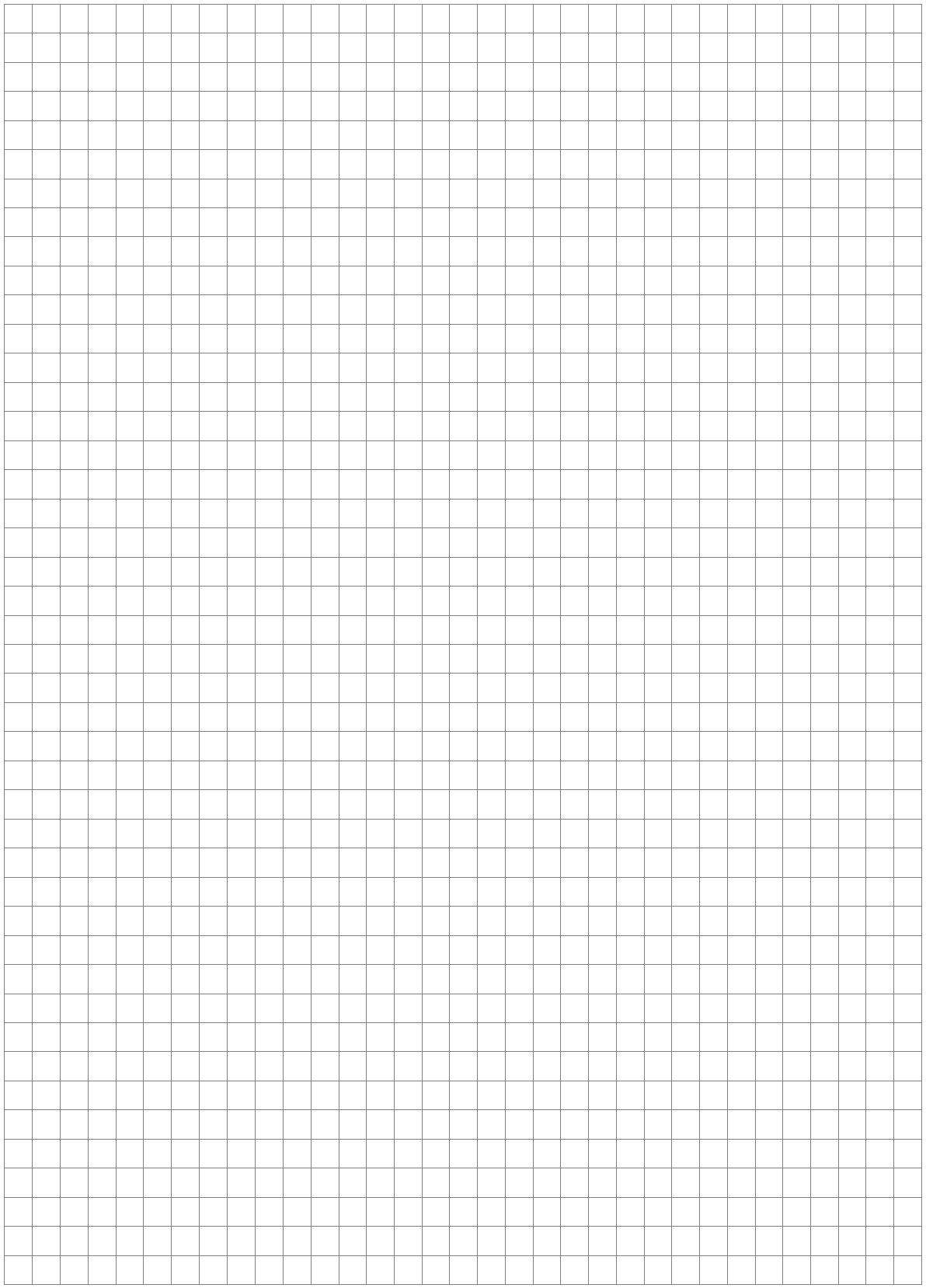
[6] Yannis Manolopoulos, Alexandros Nanopoulos, Apostolos N. Papadopoulos, and Y. Theodoridis. *R-Trees: Theory and Applications (Advanced Information and Knowledge Processing).* Springer, 1 edition, September 2005.

[7] Ralf Neubert, Otmar Görlitz, and Wolfgang Benn. Towards Content-Related Indexing in Databases. In *Datenbanksysteme in Büro, Technik und Wissenschaft (BTW), 9. GI-Fachtagung, Oldenburg*, pages 305–321. Springer-Verlag, 2001.

[8] Emanuel Parzen. On Estimation of a Probability Density Function and Mode. *The Annals of Mathematical Statistics*, 33(3):1065–1076, 1962.

[9] Thomas Schreck and Zhengxin Chen. R-Tree Implementation Using Branch-Grafting Method. In *SAC '00: Proceedings of the 2000 ACM symposium on Applied computing*, pages 328–332, New York, NY, USA, 2000. ACM.

[10] Gerhard Widmer and Miroslav Kubat. Learning in the Presence of Concept Drift and Hidden Contexts. *Mach. Learn.*, 23(1):69–101, 1996.



Anforderungen an die Anwendungsprotokollierung mittels XML

Daniel Schall
AG Datenbanken und Informationssysteme
Technische Universität Kaiserslautern
Email: d_schall@cs.uni-kl.de

Henrik Loeser
IBM Deutschland Research & Development
D-71032 Böblingen
Email: hloeser@de.ibm.com

21. April 2009

1 Abstract

Anwendungen setzen zunehmend auf Protokollierung ihrer Aktivitäten, sei es aus Gründen der Nachverfolgbarkeit, zu Abrechnungszwecken oder zur Fehleranalyse. Die entstehenden Dokumente sind geschäftskritisch und müssen daher sicher gespeichert werden. Aufgrund seiner Flexibilität hat sich XML in vielen Anwendungsbereichen als bevorzugtes Protokollformat durchgesetzt. In diesem Artikel werden wir zeigen, woher der wachsende Bedarf an Protokollierung stammt und welche Anforderungen an eine Infrastruktur zur Verarbeitung dieser Protokolle bestehen. Wir werden die Bedeutung von XML für diesen Kontext beschreiben und die Vorteile des Formats erläutern. Ein exemplarisches Szenario wird eine einfache Infrastruktur aufzeigen, die sich zur anschaulichen Erläuterung der prinzipiellen Anforderungen nutzen lässt. Abschließend werden einige neue Anforderungen angesprochen, die bislang von Standardsoftware noch nicht hinreichend unterstützt werden und bei denen noch Verbesserungsbedarf besteht.

2 Einleitung

Moderne IT-Infrastrukturen bewegen sich seit Jahren weg von einem zentralisierten Mainframe hin zu einer verteilten Client-Server-Architektur. Zunehmend werden Aufgaben und Prozesse auf mehrere Rechner verteilt. Serviceorientierte Architekturen ermöglichen rechnerübergreifende Prozesse, aber auch Abteilungs-, Standort- oder sogar Unternehmensgrenzen werden überschritten. Diese Trends fördern den Bedarf an Protokolldaten, um geschäftskritische Ereignisse verfolgen zu können.

2.1 Motivation

Viele Unternehmen verfügen über ein verteiltes Netz von Rechner, die Protokolldaten erzeugen. Ein neues und in letzter Zeit häufiger beobachtetes Problem besteht für viele Unternehmen darin, diese Protokolle zentral zu sammeln und auszuwerten (Application Logging). Außerdem müssen die Daten über mehrere Monate sicher gespeichert und jederzeit verfügbar sein. Da die Unternehmen bisher nicht über die notwendige Infrastruktur verfügen, um diese Aufgaben zu erfüllen, musse eine Lösung erarbeitet werden. Zunächst soll jedoch erläutert werden, woher der zunehmende Bedarf an Protokollen stammt.

2.2 Application Logging

Anwendungsprotokolle werden aus vielerlei Gründen erzeugt. Zum einen zur Nachvollziehbarkeit von Vorgängen sowie auf Grund von gesetzlicher Auflagen, zum anderen aus technischen Gründen um den Ablauf von Anwendungen verfolgen zu können.

2.3 Nachvollziehbarkeit

Die Abrechnung von Diensten erfolgt zunehmend nach On-Demand-Regeln und nach tatsächlicher Nutzung. Eine pauschale Abrechnung von Komplettleistung weicht dabei der feingranularen Bereitstellung und entsprechenden Abrechnung von kleinen Service-Einheiten. Ein Beispiel ist der Trend weg vom Komplettkauf eines Server zum flexiblen Einkauf von Rechenzeiten und Speicherplatz in Cloud-Computing-Szenarien. [1] Dadurch werden viele kleine Leistungseinheiten beansprucht, die exakt und nachvollziehbar abgerechnet werden müssen. Gesetzliche Regelungen wie der Sarbanes-Oxley-Act, die Unternehmen zu detaillierten Protokollierung aller finanziellen Aktivitäten zwingen, und die zunehmende beleglose Buchung von Transaktionen fördern ebenso den Bedarf an detaillierten Protokollen. [2] Allein die Finanzindustrie hat über ein Dutzend Standards geschaffen, um den Informationsaustausch zu vereinheitlichen und die vorgeschriebenen Daten zu erfassen. FIXML[1] (Financial Information eXchange Markup Language) oder FpML[2] (Financial Products Markup Language) sind nur zwei Beispiele für neu entstandene Protokollformate. Der Inhalt der Protokollen orientiert sich dabei am Prozess und den zur Durchführung der Aktivität benötigten Daten wie etwa Buchungsnummern, Beträgen und workflow-spezifische Angaben. [3]

2.3.1 Technische Motive

Auf der anderen Seite erfordern verteilte Anwendungen eine Verfolgung ihrer Aktivitäten zur Ablaufanalyse und Fehlerbehebung. Hier sind technische Daten, wie etwa Bearbeitungszeit, Serverauslastung oder beanspruchter Speicherplatz, von hohem Interesse. Für diese Art der Protokolldaten gibt es keine gesetzlich vorgegebenen Aufbewahrungsfristen. Sicherheitskritische Daten wie etwa Zugriffsprotokolle oder Fehleranalysen können jedoch ebenfalls über einen längeren Zeitraum von Interesse sein. Ein Standardformat für technische Protokolle hat sich noch nicht durchgesetzt.

3 Eigenschaften der Protokolle

Als Formatsprache der Protokolldateien hat sich XML durchgesetzt, da es flexible Dateiformate mit optionalen und wiederholenden Elementen erlaubt und zahlreiche Erweiterungen zur Auswertung und Transformation der Daten bietet. Zu nennen sind hier XML Schema zur Definition von XML-Datenstrukturen, die XQuery-Abfragesprache zur Auswertung von XML-Dokumenten, sowie XSLT zur Umformung von XML-Dokumenten in andere Formate. Trotz verbreiteter XML-Standards wie das erwähnte FIXML, sind die entstehenden Protokolldateien sehr heterogen. So umfasst die Version 4.4 von FIXML bereits 1310 verschiedene XML Typen und 41 XSD Dateien.
Das Headerformat der Nachrichten ist dabei meist identisch, der Inhalt variiert.
Anwendungen generierten meist hybride Protokolldateien, die sowohl geschäftskritische, als auch technische Daten in einer Protokolldatei vereinen.
Spitzenlasten von bis zu 500 Nachrichten pro Sekunde sind ebenso möglich wie ein Gesamtaufkommen von bis zu 20 Millionen Nachrichten pro Geschäftstag.
Bei einer durchschnittlichen Größe von 4 - 20 kByte pro Nachricht können also monatlich ca. 7

[1] `http://www.fixprotocol.org/`
[2] `http://www.fpml.org/`

Terabyte an Protokolldateien anfallen. [4]
Die Lebensdauer der Protokolle ist je nach Typ sehr unterschiedlich. Teils ist eine Mindestvorhaltedauer gesetzlich geregelt, teils bestimmt die Verwendung der Protokolle deren Lebensdauer. So sind technische Protokolldateien im Normalfall eher kurzlebig im Bereich von Stunden bis einigen Tagen, geschäftliche Protokolldaten erfordern teilweise eine Lebensdauer von Jahren oder sogar Jahrzehnten. In Deutschland sind beispielsweise alle Rechnungen, gestellt oder erhalten, zehn Jahre lang aufzubewahren. Darüber hinaus müssen diese den Finanzbehörden auf Verlangen unverzüglich zur Verfügung gestellt werden. [5]

3.1 Log Shipping

Da die Nachrichten zum Großteil von Anwendungen generiert werden, die auf speziell angepasster Hardware ausgeführt werden, wie etwa Geldautomaten oder integrierte Schaltungen, können diese weder ausreichend Speicherplatz noch Verarbeitungskapazitäten bereitstellen, um die Nachrichten direkt am Ort ihrer Entstehung verarbeiten zu können. Die Nachrichten müssen daher über ein Netzwerk an einen (zentralisierten) Speicherort weitergeleitet werden, bevor die Auswertung und Archivierung der Daten erfolgen kann. Da sich die Netzwerklandschaft der Kunden über ein Netz aus Filialen, Tochterfirmen und ausgelagerten Betriebsstätten erstreckt, besteht der Wunsch, alle anfallenden Protokolldaten vorab zu filtern und die geschäftskritischen Anteile in die Hauptverwaltung zur Auswertung und Sicherung zu leiten. Der Verlust von Nachrichten muss dabei ausgeschlossen werden.

4 Infrastruktur

Eine Infrastruktur, die Funktionen für das Sammeln der Protokolle bereitstellt und in einen Backend-Speicher leitet, wird benötigt. Die Anforderungen an diese und der Entwurf einer Infrastruktur basierend auf IBM Technologien wird im Folgenden vorgestellt.

4.1 Anforderungen an die Infrastruktur

Basierend auf den Eigenschaften der Protkolle und den Bedürfnissen der Unternehmen, ergeben sich für die Infrastruktur zur Anwendungsprotokollierung folgende Anforderungen: Die Infrastruktur muss eine **Schnittstelle für die Annahme von Protokolldateien** bieten, sodass Anwendungen Daten übergeben können. Alle Arbeitsschritte sollen wenn möglich **Unterstützung für verteilte Transaktionen** bieten um einen Verlust von Nachrichten ausschließen zu können. Ein **Caching** von Nachrichten auf Clientseite ist hilfreich um Netzwerke zu entlasten und Clients von der Infrastruktur zu entkoppeln. Ebenso kann **Routing** helfen, die Protokolle zu verschiedenen Back-End-Speichern zu leiten. Die Infrastruktur kann durch **Transformation** die Nachrichten in ein einheitliches Format bringen, bevor diese gespeichert werden. Insbesondere können dadurch beliebige Protokollformate in XML überführt werden um eine einheitliche Verarbeitung zu ermöglichen. Gleichzeitig muss die Infrastruktur eine **hohe Durchsatzrate** erreichen, um alle anfallenden Protokolle verarbeiten zu können. Eine **effiziente Auswertung** und **persistente Speicherung** der Protokolle muss trotz der großen Datenmenge gewährleistet werden, die Speicherung auf Bandlaufwerken oder in Dateiordnern scheidet daher für aktuelle Protokolldateien aus. Weiterhin soll die Infrastruktur eine gute **Verteilbarkeit** aufweisen und mit wachsender Anzahl der Clients **Skalierbarkeit** bringen. Da die Protokolldaten im XML-Format vorliegen, soll die Infrastruktur eine gute Unterstützung dieses Formats bieten. Daher sind XQuery-Unterstützung, XSLT und die Unterstützung von XML-Schemata wünschenswert.

4.2 Entwurf einer Infrastruktur

Die vorgestellten Anforderungen an die Infrastruktur sind bisher von keiner eigenständigen Softwarelösung zu erbringen. Durch eine Kombination mehrerer bekannter Technologien lässt sich ein grundlegendes Szenario für das Application Logging jedoch einfach entwerfen. Abbildung 1 zeigt den Systementwurf, der im Folgenden vorgestellt wird.

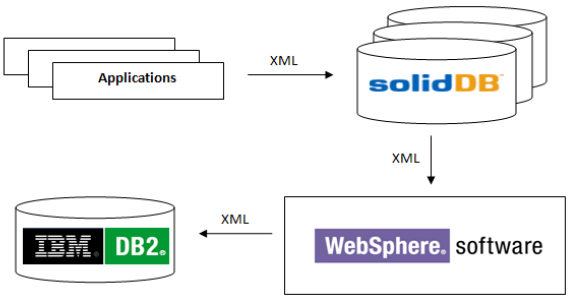


Abbildung 1: Szenario

Als Cache und zur Entkopplung der Anwendungen vom Protokollversand wird IBMs solidDB Datenbank eingesetzt. Diese In-Memory-Datenbanken dienen als Zwischenspeicher zur Vermeidung von Lastspitzen. WebSphere Message Queue dient als persistente Warteschlange, die Nachrichten speichert und über große Distanzen leiten kann. WebSphere Message Broker (WMB) wird eingesetzt, um die Protokolle zu unterschiedlichen Zielen routen, den Inhalt analysieren und Nachrichten in andere Formate überführen. WMB unterstützt XML-Daten und bietet dneben dem Import von XML-Schemata, XSLT-Umformungen und XQuery außerdem eine native Unterstützung für XML-Daten. WebSphere Message Broker wird auch eingesetzt, um die Nachrichten in die Back-End-Datenbank zu schreiben. Da der Message Broker verteilte Transaktionen (XA) unterstützt, ist ein Verlust von Nachrichten ausgeschlossen. Als persistenter Back-End-Datenspeicher wird IBMs DB2 for Linux, Unix and Windows eingesetzt. Dieses Datenbanksystem ist einerseits durch die pureXML-Technologie in der Lage, XML-Nachrichten nativ zu speichern und andererseits die aufkommende Datenlast durch Partitionierung der Datenbank aufzuteilen. Die Partitionierung von XML-Daten ist in der neuesten Version der DB2 hinzugekommen und erlaubt es nun, auch Tabellen, die XML-Spalten enthalten, zu partitionieren. Da die DB2 auch das Erstellen von XML-Indices erlaubt, ist eine effiziente Analyse der Datenmengen möglich.

5 Fazit

Es wurden die Anforderungen an eine Application Logging Infrastruktur vorgestellt und es konnte gezeigt werden, dass der Aufbau einer Infrastruktur möglich ist. Die vorgestellte Infrastruktur erfüllt die prinzipiellen Anforderungen an die Anwendungsprotokollierung, allerdings bleiben noch viele Möglichkeiten für Verbesserungen. Eine detaillierte Ausarbeitung und Anpassung auf kundenspezifische Gegebenheiten steht bei diesem generischen Ansatz noch aus, ebenso die Auswertung der XML-Daten im Backend. Auch wurden Sicherheitsaspekte in diesem Entwurf nicht betrachtet.

6 Ausblick

Die professionelle Anwendungsprotokollierung ist ein bisher kaum beachtetes Arbeitsfeld. Zwar bieten einzelne Produkte (SAP, DB2, Microsoft Windows, etc.) Tools zur Auswertung anwendungseigener Protokolle, eine allgemeine Lösung für die Auswertung von beliebigen Protokol-

len ist allerdings noch nicht verfügbar. Auch beim Log Shipping setzen Hersteller auf eigene Lösungen. Eine integrierte Infrastruktur, die definierte Schnittstellen für Clientanwendungen bietet ist derzeit ebenfalls noch nicht absehbar. Wünschenswert wäre eine Standardisierung der Middleware für die Anwendungsprotokollierung, wie schon in einigen Arbeiten angedacht. [6] Einheitliche Benchmarks wie TPoX, ein Benchmark für XML-Datenbanken, die speziell auf die Probleme und Anforderungen der Protokollierung verteilter Anwendungen abzielen, sind ebenso wie ein einheitliches, domänenübergreifendes Protokolldateiformat, das geschäftlichen und technische Informationsbedürfnisse vereinen kann, noch zu entwickeln. [7] Die Speicherung großer relationaler Datenmengen ist zwar von DBM-Systemen unterstützt und praxiserprobt, zur Speicherung großer Mengen von XML-Daten gibt es jedoch weit weniger Erfahrung. Speziell XML-Warehousing und die Echtzeitanalyse von XML-Daten sind weit weniger ausgereift als ihre relationalen Entsprechungen. [8]

7 Zusammenfassung

Es wurde gezeigt, woher der wachsende Bedarf an Anwendungsprotokollen stammt, welche Anforderungen an die Anwendungsprotokollierung bestehen und wie eine unterstützende Infrastruktur basierend auf Standardsoftware aufgebaut werden kann. Die Bedeutung von XML in diesem Kontext wurde erläutert und gezeigt, wie die Infrastruktur die Vorteile von XML nutzen kann.

Deutlich wurde, dass das Application Logging noch Potential bietet, insbesondere die Vereinheitlichung der Infrastruktur, sowie verbesserte Auswertungsmöglichkeiten sind wünschenswert.

Literatur

[1] Hayes and Brian. Cloud computing. *Commun. ACM*, 51(7):9–11, 2008.

[2] Rakesh Agrawal, Christopher Johnson, Jerry Kiernan, and Frank Leymann. Taming Compliance with Sarbanes-Oxley Internal Controls Using Database Technology. In *ICDE '06: Proceedings of the 22nd International Conference on Data Engineering*, page 92, Washington, DC, USA, 2006. IEEE Computer Society.

[3] A. Dan, D. M. Dias, R. Kearney, T. C. Lau, T. N. Nguyen, F. N. Parr, M. W. Sachs, and H. H. Shaikh. Business-to-business integration with tpaml and a business-to-business protocol framework. *IBM Syst. J.*, 40(1):68–90, 2001.

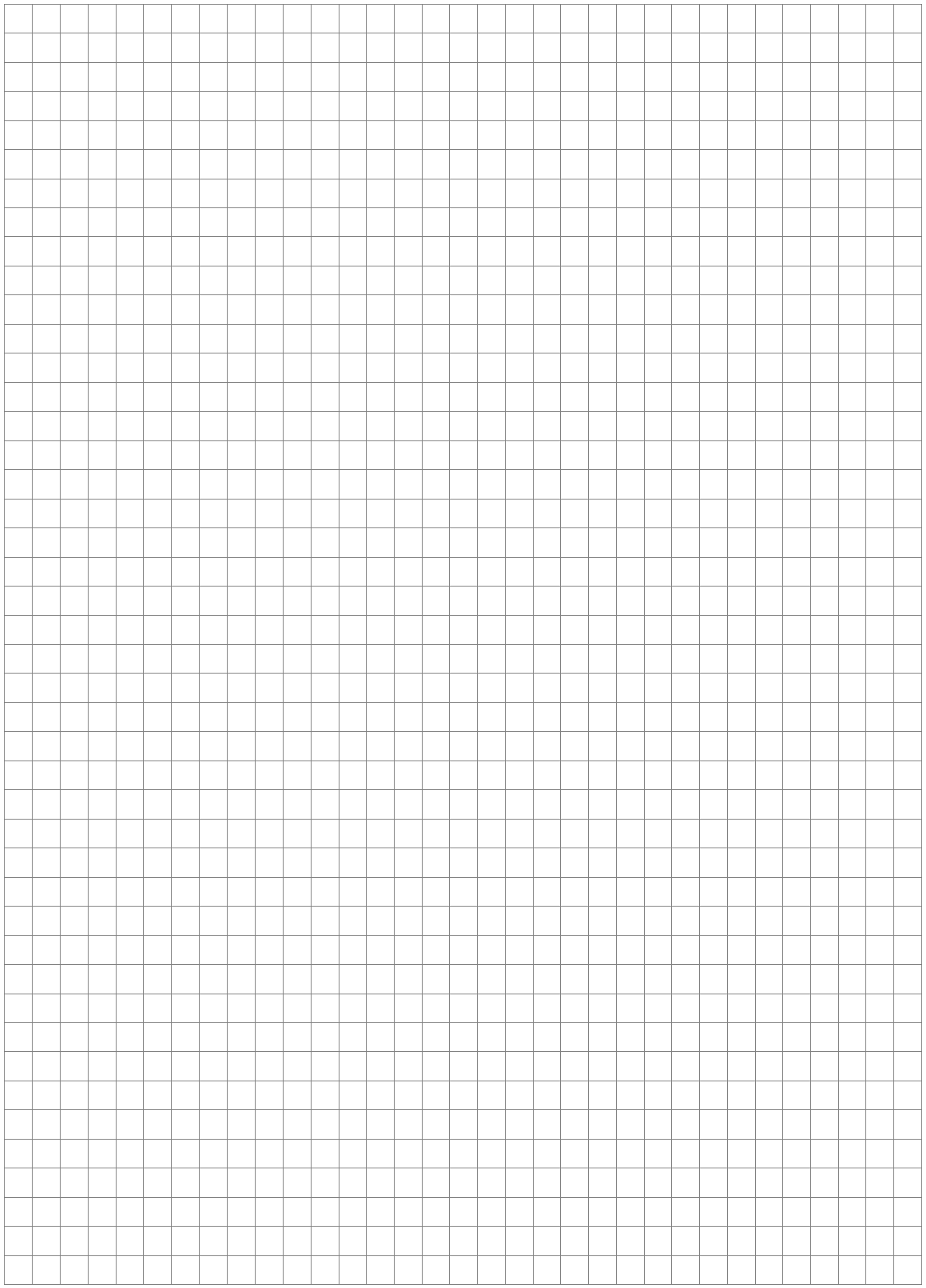
[4] Henrik Loeser, Matthias Nicola, and Jana Fitzgerald. Index Challenges in Native XML Database Systems. *BTW 2009*, März 2009.

[5] Deutsches Umsatzsteuergesetz, §14b, „Aufbewahrung von Rechnungen“.

[6] Marcelo Pitanga Alves, Paulo F. Pires, Flávia Coimbra Delicato, and Maria Luiza Machado Campos. Middlog: A web service approach for application logging. In Kurt Bauknecht, Birgit Pröll, and Hannes Werthner, editors, *EC-Web*, volume 3590 of *Lecture Notes in Computer Science*, pages 337–347. Springer, 2005.

[7] Matthias Nicola, Irina Kogan, and Berni Schiefer. An xml transaction processing benchmark. In *SIGMOD '07: Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 937–948, New York, NY, USA, 2007. ACM.

[8] Byung-Kwon Park, Hyoil Han, and Il-Yeol Song. *XML-OLAP: A Multidimensional Analysis Framework for XML Warehouses*, volume 3589. 2005.



Kooperatives XML-Authoring: Integration von Workgroups in Workspace-Szenarien

Francis Gropengießer
francis.gropengiesser@tu-ilmenau.de

Zusammenfassung

Kooperatives XML-Authoring, wie es u.a. in Sounddesign-Prozessen erfolgt, findet in unterschiedlichen Systemumgebungen statt. Workgroups stellen dabei eng gekoppelte, Workspace-Szenarien hingegen lose gekoppelte Systemumgebungen dar. Für das Workgroup-Szenario haben wir in vergangenen Arbeiten bereits eine Lösung vorgeschlagen und implementiert, die einen hohen Grad an Kooperation ermöglicht. In dieser Arbeit stellen wir einen ersten Ansatz vor, wie bestehende Workgroups in Workspace-Szenarien integriert werden können. Hauptbestandteil dieses Integrationskonzepts ist ein maßgeschneidertes Versionierungssystem, welches die Synchronisierung konkurrierender Updates ermöglicht. Dieses arbeitet nicht, wie z.B. Subversion, auf der textuellen Repräsentation der XML-Daten, sondern betrachtet die Daten als Baum. Deltas zwischen verschiedenen Versionen werden aus den innerhalb einzelner Workgroups gespeicherten Transaktions-Recovery-Informationen gewonnen. Es ist daher kein aufwendiger Diff-Algorithmus notwendig. Weiterhin ermöglicht uns die Wiederverwendung der Recovery-Informationen auch Verschiebeoperationen zu unterstützen. Für die serverseitige Versionsverwaltung untersuchen wir einige ausgewählte Verfahren auf ihre Einsatztauglichkeit im Workspace-Szenario. Dabei handelt es sich um die Versionsverwaltung mit „Completed Deltas“, ein referenzbasiertes Versionsmodell sowie ein Versionsstempel-Verfahren.

1 Einleitung

Kooperatives XML-Authoring ist in vielen Medienproduktionsprozessen eine zentrale Aufgabenstellung. Ein Beispiel ist die Postproduktion von Filmton mit dem räumlichen Soundsystem IOSONO (*http://www.iosono-sound.com/*). Dieses System ermöglicht die Positionierung virtueller Soundquellen im dreidimensionalen Raum, wodurch ein realistischer Raumklangeindruck entsteht. Die bei den Arbeiten der Autoren entstehenden Metainformationen werden in XML-Dateien (Szenengraphen) abgelegt. Ziel unserer Bemühungen ist es, ein kooperatives Bearbeiten gleicher Szenengraphen durch mehrere Autoren zu ermöglichen. Dabei müssen jedoch zwei unterschiedliche Systemumgebungen unterschieden werden, in denen das XML-Authoring abläuft:

- Das *Workgroup-Szenario* beschreibt eine eng gekoppelte Systemumgebung. Hier arbeitet ein kleines Autorenteam gemeinsam an einer Soundszene. Dabei ist die räumliche Trennung der Autoren relativ gering. Es kann davon ausgegangen werden, dass sie sich in einem Gebäude befinden. In [GS08, GHS09] haben wir eine Lösung für das Workgroup-Szenario vorgeschlagen und implementiert. Diese basiert auf einem Client-Server-Modell, in dem alle Autoren (Clients) permanent mit dem zentralen Datenbestand (Server) verbunden sind. Die Arbeitsschritte der Autoren werden in maßgeschneiderten Transaktionen gekapselt und über ein Sperrprotokoll, welches ihre Semantik ausnutzt, synchronisiert. Die Systemlösung berücksichtigt die besonderen Anforderungen kooperativer Designprozesse und ermöglicht daher eine unmittelbare Propagierung von Änderungen, einen multi-direktionalen Informationsfluss sowie das Verwerfen einzelner Arbeitsschritte.
- Im Gegensatz zum Workgroup-Szenario beschreibt das *Workspace-Szenario* eine lose gekoppelte Systemumgebung. Hier arbeiten mehrere Entwicklerteams, z.B. verschiedene Tonstudios, u.U. weltweit verteilt an der Postproduktion des Tons eines kompletten Films.

Jedes Team arbeitet vorwiegend entkoppelt von allen anderen Teams an einem Teil des gesamten Szenendatenbestands. Es kann nicht davon ausgegangen werden, dass alle Autoren permanent mit einer zentralen Datenbasis verbunden sind. Die Synchronisierung konkurrierender Änderungen unterschiedlicher Teams an gleichen Szenendaten erfordert daher spezielle, entkoppelte Techniken.

Die besondere Herausforderung bei der Realisierung des Workspace-Szenarios liegt in der fehlenden permanenten Verbindung aller Teams mit einer zentralen Datenbasis. Diese Einschränkung verhindert die unmittelbare Propagierung von Änderungen, wodurch es nicht möglich ist, dass alle Teams immer auf dem aktuellen Stand des Projekts sind. Somit lässt sich das Kooperationsprinzip nicht vollständig realisieren. Um zumindest eine teilweise Umsetzung des Prinzips zu erzielen, haben wir uns für den Einsatz eines Versionierungssystems entschieden. Dieses bietet gegenüber traditionellen Transaktionsmodellen den Vorteil, dass es auf Serialisierbarkeit als Korrektheitskriterium verzichtet. Dieses Kriterium verhindert multi-direktionale Informationsflüsse und widerspricht somit gänzlich dem kooperativen Grundgedanken.

In dieser Arbeit stellen wir einen ersten Ansatz für die Realisierung eines geeigneten Versionierungssystems vor. Dieses arbeitet nicht, wie z.B. Subversion (*http://subversion.tigris.org/*), auf der textuellen Repräsentation der XML-Daten, sondern betrachtet die Daten als Baum. Im nächsten Abschnitt stellen wir dazu zunächst das Integrationskonzept vor und gehen dabei konkret auf die Ermittlung der Deltas aus den Transaktions-Recovery-Informationen einzelner Workgroups ein. In Abschnitt 3 werden anschließend vier Versionsverwaltungssysteme vorgestellt und hinsichtlich ihrer Einsatztauglichkeit im Workspace-Szenario bewertet. Im letzten Abschnitt geben wir eine kurze Zusammenfassung und einen Ausblick auf zukünftige Arbeiten.

2 Integrationskonzept

Ein mögliches Konzept zur Integration von Workgroups in Workspace-Umgebungen ist in Abbildung 1 dargestellt. Es beruht auf einer Client-Server-Architektur mit einem globalen Server und den lokalen Servern der einzelnen Workgroups als Clients. Der globale Server verwaltet dabei den gesamten Szenendatenbestand, wohingegen jeder Client lediglich Teile dieses Bestands besitzt. Wie bereits in der Einleitung erwähnt, besteht keine permanente Verbindung zwischen den Clients und dem globalen Server. Der typische Arbeitsablauf in dieser Workspace-Umgebung sieht wie folgt aus:

1. Jedes Entwicklerteam (Workgroup) lädt sich einen Teil des gesamten Szenendatenbestands vom globalen Server auf den lokalen Server. Danach wird die Verbindung zum globalen Server geschlossen.
2. Jedes Team arbeitet entkoppelt von allen anderen Teams am lokalen Szenendatenbestand. Diese Arbeit erfolgt kooperativ mit den in [GS08] beschriebenen Modellen und Methoden.
3. Hat sich das Team auf einen finalen Zustand des lokalen Szenendatenbestands geeinigt, wird eine Verbindung zum globalen Server aufgebaut und die finale Version hochgeladen.

Punkt 3 wirft dabei zwei entscheidende Fragen auf:

1. Wie werden die Änderungen eines Teams erkannt?
2. Wie lässt sich das Mergen der Änderungen verschiedener Teams gestalten? Dabei ist besonders die Behandlung von Konfliktfällen interessant.

In dieser Arbeit gehen wir lediglich auf die erste Frage genauer ein. Die Ermittlung von Änderungen (des Deltas) zwischen zwei Versionen eines XML-Dokumentes erfordert typischerweise die Anwendung aufwendiger Diff-Algorithmen, wie sie z.B. in [CRGMW96, CAM01] beschrieben sind. Einen besonders kritischen Fall stellt dabei das Erkennen von Move-Operationen dar [CAHB00]. In unserem System kann jedoch auf einen derartigen Diff-Algorithmus verzichtet werden. Dies ist möglich, da wir die Transaktions-Recovery-Informationen, die in den

einzelnen Workgroups in Form von Logs vorliegen, wiederverwenden. Diese beinhalten alle Änderungen in Form einer Historie von Änderungsoperationen. Derzeit werden in unserem System die Änderungsoperationen *Insert*, *Edit*, *Move* und *Delete* unterstützt. Voraussetzung für die Funktionsweise dieses Verfahrens ist, dass alle Datenobjekte (Knoten und Kanten) global eindeutig identifiziert werden können. Dies ist durch das in [GS08] beschriebene Datenmodell, in dem IDs zur eindeutigen Identifikation von Knoten und Kanten verwendet werden, sichergestellt. Ein Beispiel für ein derartiges Log ist in Abbildung 2 dargestellt. Für jeden Knoten

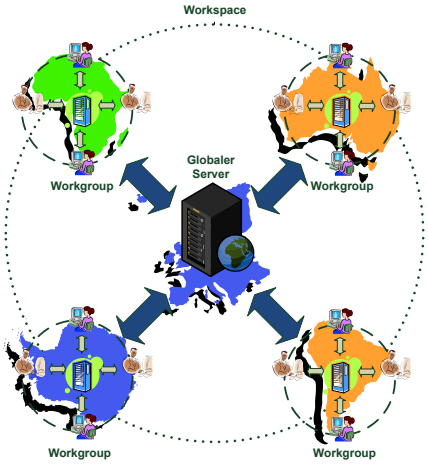


Abbildung 1: Workspace-Szenario

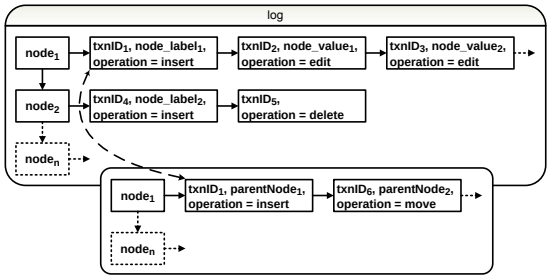


Abbildung 2: Beispiel für ein Log

werden in diesem Log folgende Informationen festgehalten:

- Wurde ein Knoten neu eingefügt, wird ein entsprechender Eintrag im Log erstellt.
- Für jede Editieroperation, die den Text eines Elements oder den Attributwert eines Attributes ändert, wird ein neuer Eintrag mit dem neuen Text oder Attributwert angelegt. Dabei wird angenommen, dass der alte Text oder Attributwert vollständig ersetzt wird.
- Das Löschen eines Knotens führt zu einem Eintrag.
- Die Kanten werden in Form von Vater-Kind-Beziehungen repräsentiert. So wird beim Einfügen eines Knotens ein zusätzlicher Eintrag geschrieben, der die Vater-ID beinhaltet. Jede Verschiebeoperation führt zu einem Eintrag im Log, der den neuen Vater des verschobenen Knotens enthält.

Wurden die Arbeiten innerhalb der Workgroup abgeschlossen, kann die finale Version der Szenendaten zum globalen Server übertragen werden. Dabei ist es ausreichend, den letzten Eintrag aus dem Log für jeden Knoten zu übertragen. Der Text oder Attributwert ergibt sich aus dem letzten Editiereintrag. Die Kanten ergeben sich aus den aktuellen Vater-Kind-Beziehungen. Lediglich bei neu eingefügten und danach verschobenen und editierten Knoten muss eine zusätzliche Operation $Insert(node_id, parent_id, node_label, node_value)$ generiert und übertragen werden. Diese Knoten existieren noch nicht auf dem globalen Server, und somit haben einfache Verschiebe- und Editieroperationen dort keinen Bezug. Wurde ein Knoten innerhalb der Arbeiten einer Workgroup eingefügt und wieder gelöscht, ist keine Übermittlung eines Eintrags für diesen Knoten erforderlich.

3 Versionsverwaltung

Die Versionsverwaltung von XML-Dokumenten ist ein altbekanntes Problem. Viele verschiedene Verfahren, wie z.B. [CTZ00, CTZZ01], wurden bereits entwickelt und kommen auch zum Einsatz. In dieser Arbeit untersuchen wir vier ausgewählte Verfahren, welche die Versionsverwaltung auf logischer Ebene, also auf Dokument-Ebene, betrachten. Diese Verfahren werden

hinsichtlich ihrer Einsatztauglichkeit im Workspace-Szenario bewertet und verglichen. Ein Versionsverwaltungssystem ist als tauglich für unseren speziellen Anwendungsfall anzusehen, wenn es mindestens die folgenden drei Kriterien erfüllt:

- Es sollten alle im letzten Abschnitt aufgeführten Änderungsoperationen unterstützt werden. Insbesondere sollte eine Verschiebeoperation nicht als Lösch- und Einfügeoperation betrachtet werden, sondern als „Umhängen“ eines Teilbaums unter einen anderen Vaterknoten.
- Der Aufwand zur Ermittlung einer bestimmten Version sollte möglichst gering sein. Dies garantiert kurze Antwortzeiten.
- Es sollte „Branched-Versioning“ unterstützt werden. Dies ist in Designprozessen besonders wichtig, da viele verschiedene kreative Ideen archiviert werden müssen.

Das Versionsverwaltungssystem des Web-Warehouses im Xyleme-Projekt [MACM01] setzt das Prinzip der „Completed Deltas“ ein. Diese ermöglichen es, in der Versionshistorie sowohl vorwärts als auch rückwärts zu gehen. Die Deltas werden als XML-Dokumente gespeichert. Dadurch ist es möglich, Änderungen direkt mit Hilfe von XPath/XQuery anzufragen. Das System unterstützt alle Änderungsoperationen, insbesondere wird eine Verschiebeoperation auch als „Umhängen“ eines Teilbaums betrachtet. Im System wird immer die aktuelle Version der Daten sowie alle „Completed Deltas“ gespeichert. Somit kann die aktuelle Version sofort angefragt werden. Ältere Zustände können mit Hilfe der Deltas berechnet werden. Dies führt jedoch zu hohen Berechnungskosten. Die Verwaltung von Branches ist möglich.

In [WL02] wird ein weiteres Versionsverwaltungssystem auf Basis von „Completed Deltas“ beschrieben. Das Grundprinzip ist ähnlich dem des eben beschriebenen Web-Warehouses im Xyleme-Projekt. Auch hier wird die aktuelle Version eines Dokuments und alle „Completed Deltas“ gespeichert. Allerdings werden zusätzlich Zwischenversionen des Dokuments materialisiert, um den Rekonstruktionsaufwand für eine Version zu verringern. Der Materialisierungsalgorithmus basiert auf dem Verhältnis der Anzahl notwendiger Operationen zur Rekonstruktion einer Version und dem Speicherbedarf für das Dokument.

Das „Reference-Based Version Model“(RBVM) [CTZ01] arbeitet mit Objektreferenzen. Teilbäume vorhergehender Versionen, die in der aktuellen Version unverändert sind, werden referenziert und nicht kopiert. Somit werden Redundanzen verringert, und es wird Speicherplatz gespart. Ein Vorteil ist, dass Verschiebeoperationen als „Umhängen“ betrachtet werden. Ein Nachteil dieses Systems ist, dass bei der Rekonstruktion einer Version eine Rückverfolgung von Objektreferenzen stattfinden muss, was zusätzlichen Arbeitsaufwand bedeutet und die Rekonstruktion einer Version verlangsamt. Die Verwaltung von Branches ist möglich.

Ein weiteres, vielversprechendes Modell zur Versionsverwaltung ist V-XML [RMG06]. Es ermöglicht die Repräsentation verschiedener Versionen eines XML-Dokumentes in einem einzigen Dokument. Dazu werden alle Elemente im Dokument mit Versionsstempeln versehen. Diese geben an, für welche Versionen das betroffene Element Gültigkeit besitzt. Vorteil dieses Modells ist, dass sehr leicht eine bestimmte Version eines Dokumentes angefragt werden kann. Dabei kann auf XQuery zurückgegriffen werden. Auch das „Branched-Versioning“ wird unterstützt. Ein Nachteil des Systems ist, dass Verschiebeoperationen als Löschen und Einfügen betrachtet werden.

In Tabelle 1 ist die Bewertung der vier Systeme noch einmal kurz zusammengefasst. Es kann festgehalten werden, dass lediglich das System der „Completed Deltas“ mit dem zusätzlichen Materialisierungsalgorithmus alle Anforderungen vollständig erfüllt. Dieses System ist daher für den Einsatz im Workspace-Szenario geeignet.

4 Zusammenfassung und Ausblick

In dieser Arbeit haben wir einen ersten Ansatz vorgestellt, wie bestehende Workgroups in ein Workspace-Szenario integriert werden können. Hauptbestandteil dieses Integrationskonzepts ist

System	Move	Versionsrekonstruktionskosten	„Branched-Versioning“
Web-Warehouse	unterstützt	hoch	unterstützt
Compl. Deltas + Mat.-Algo.	unterstützt	gering	unterstützt
RBVM	unterstützt	hoch	unterstützt
V-XML	nicht unterstützt	gering	unterstützt

Tabelle 1: Vergleich der Systeme

ein maßgeschneidertes Versionierungssystem. Eine Besonderheit dieses Systems ist, dass auf aufwendige Diff-Algorithmen verzichtet werden kann, da Deltas aus den Transaktions-Recovery-Informationen einzelner Workgroups gewonnen werden können. Für die serverseitige Verwaltung der Szenendaten und der Änderungen haben wir vier Systeme kurz vorgestellt und bewertet. Diese Bewertung hat ergeben, dass das System der „Completed Deltas“ mit dem dazugehörigen Materialisierungsalgorithmus alle Anforderungen erfüllt und daher für den Einsatz im Workspace-Szenario geeignet ist. In zukünftigen Arbeiten werden wir uns folgenden drei Problemen zuwenden: Zunächst werden wir uns mit dem Merge-Prozess beschäftigen. Hierbei geht es primär um die Definition von Konfliktfällen. Ein weiteres interessantes Gebiet ist die Anfrageverarbeitung auf versionierten XML-Dokumenten. Des Weiteren werden wir versuchen, den gleichzeitigen Betrieb von Workgroup und Workspace auf dem globalen Server zu realisieren. Das bedeutet, dass stark verteilte Entwicklerteams mit Hilfe von Versionierung und einzelne Autoren mit Hilfe kooperativer Transaktionen auf dem zentralen Szenendatenbestand des globalen Servers gleichzeitig arbeiten können.

Literatur

[CAHB00] Grégory Cobéna, Talel Abdessalem, Yassine Hinnach, and Domaine De Voluceau Rocquencourt Bp. A comparative study for xml change detection. Technical report, 2000.

[CAM01] Gregory Cobena, Serge Abiteboul, and Amelie Marian. Detecting changes in xml documents. In *ICDE*, pages 41–52, 2001.

[CRGMW96] Sudarshan S. Chawathe, Anand Rajaraman, Hector Garcia-Molina, and Jennifer Widom. Change detection in hierarchically structured information. In *SIGMOD '96*, pages 493–504, 1996.

[CTZ00] Shu-Yao Chien, Vassilis J. Tsotras, and Carlo Zaniolo. Version management of xml documents. In *WebDB (Informal Proceedings)*, pages 75–80, 2000.

[CTZ01] Shu-Yao Chien, Vassilis J. Tsotras, and Carlo Zaniolo. Efficient management of multiversion documents by object referencing. In *VLDB '01*, pages 291–300, 2001.

[CTZZ01] Shu-Yao Chien, Vassilis J. Tsotras, Carlo Zaniolo, and Donghui Zhang. Storing and querying multiversion xml documents using durable node numbers. In *Proc. of WISE*, pages 270–279, 2001.

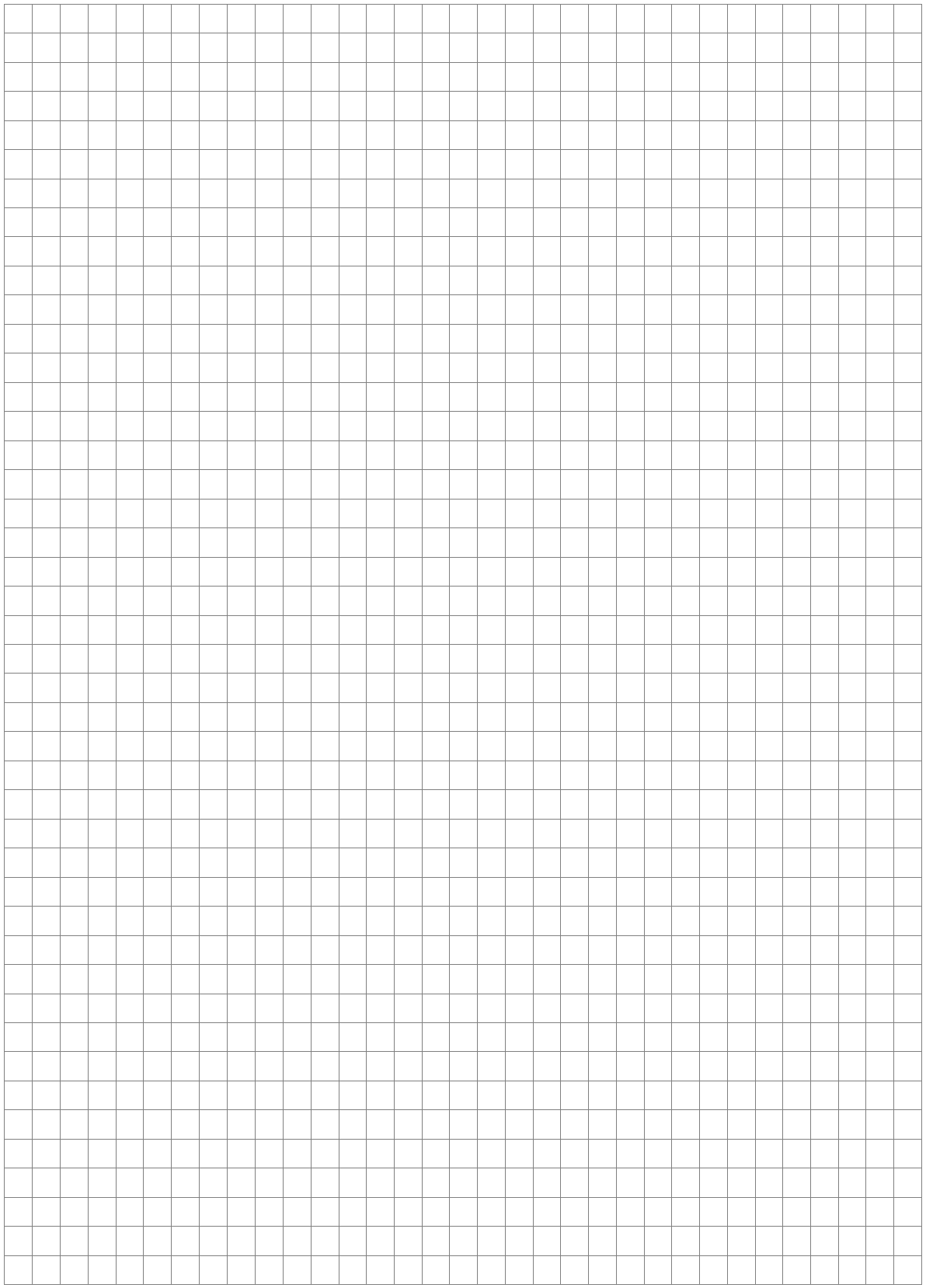
[GHS09] Francis Gropengießer, Katja Hose, and Kai-Uwe Sattler. Ein kooperativer XML-Editor für Workgroups. In *BTW*, 2009.

[GS08] Francis Gropengießer and Kai-Uwe Sattler. An Extended Cooperative Transaction Model for XML. In *PIKM'08 icw CIKM'08*, pages 41–48, 2008.

[MACM01] Amélie Marian, Serge Abiteboul, Gregory Cobena, and Laurent Mignet. Change-centric management of versions in an xml warehouse. In *VLDB '01*, pages 581–590, 2001.

[RMG06] Luis Arévalo Rosado, Antonio Polo Márquez, and Juan María Fernández González. Representing versions in xml documents using versionstamp. In *ER (Workshops)*, pages 257–267, 2006.

[WL02] Raymond K. Wong and Nicole Lam. Managing and querying multi-version xml data with update logging. In *DocEng '02*, pages 74–81, 2002.



XQuery & OLAP

Michael Seiferle, michael.seiferle@uni-konstanz.de

2009-04-09

Abstract

Online Analytical Processing on XML became a topic in the database research community[1; 2; 3; 4; 5], which is mostly due to the possibilities of XML to overcome the constraints of the relational multidimensional model[6; 7].

This paper gives an overview of chances and challenges in terms of storing multidimensional heterogenous data in XML and how to query it. We use BaseX, a native XML database developed at the chair for Database and Information Systems (U Konstanz) as a back-end to store and query the data.

The first part of this paper briefly deals with modeling aspects and provides an overview on how to store metadata for non-conventional (i.e. text, compound measures, irregular dimensional hierarchies) data in XML. While we do propose a metaschema for this multidimensional model, we try to allow a certain degree of freedom with respect to the characteristics of the actual data to be stored inside the data warehouse. Throughout this paper XML data from an example Bookstore will be used to support our concepts.

The second part focusses on querying heterogenous and imprecise data. As XQuery 1.0[8], at the moment, only offers very basic support for multidimensional querying, we first explore what enhancements are needed to meet up with Relational OLAP in terms of querying. In contrast to Relational OLAP where missing, incomplete and de-normalized data is avoided at the schema layer, *OLAP-enabled* XQuery offers a whole new perspective on querying non conventional data. We will show how heterogenous schemata, possibly derived from numerous data sources may, to some extent, be queried without prior normalization. We achieve this with the help of relaxing the group by operation. We add an optional granularity argument such that summarizability requirements are still met. Although our research on OLAP-enabled XQuery in BaseX has just begun, we think that our approach is promising for real world applications.

The last section offers some very basic ideas on integrating Full-Text features in multidimensional queries. We will give an future prospect on how XQuery's Full-Text abilities in conjunction with OLAP may be used to analyze textual contents.

1 Introduction

XML has become one of the premier formats for data interchange and information representation on the web[1]. The main reasons that led to this success are mostly due to XMLs simplicity and expressiveness: In fact XML allows us to store arbitrary information with respect to some simple rules. Wether it is relational data, semi-structured data or even binary objects – they are all represented in a uniform manner and syntax. Lots of XML datasets are available today that would not have to converted to the relational model while on the other hand the ETL process could easily be changed to create XML instead of relational data.

In accordance to [6; 7] there are challenges that data warehouses for real world data still face today, among them are:

1. Warehousing imperfect and unstructured data
2. Querying imperfect and unstructured data

3. Full Text Capabilities

Researchers propose new concepts[5; 6; 7] where data and data models are in fact two views upon the same entity and should not be constrained by the strict rules of multidimensional model. We think that XML might be a huge step in overcoming these constraints. The increased complexity of data representation in XML over relational storage in terms of hierarchical modeling and compound data-types leads to a much more flexible data-model.

2 Application Scenario

2.1 Modeling & Storage Aspects

> The fundamental goals of a data warehouse can be developed by walking around the halls of any large organization and listening to the management talk. [...]
> These concerns are so universal that they drive the bedrock requirements for the data warehouse. [9, S. XXV]

Kimball's quote elucidates how versatile, and thus universal, both possible queries and data structures inside the data warehouse are. Up until today developing a suitable data model for complex business data is a challenging task. During model development the user makes assumptions and simplifications that might prove as a restriction in later stages of data warehousing. Wether this happens because of ignorance or simply because there is no need to analyze that particular facet of data at that given time: Making changes to the conceptual model afterwards often becomes even more of a challenge. Throughout this paper we will use a minimal bookstore example to illustrate basic concepts and challenges that arise when performing OLAP on XML data.

2.1.1 XQuery Enriched Metadata

As we plan to use XQuery for analysis queries using XQuery in conjunction with XML as a metadata layer comes in handy. Our approach is *XQuery enriched metadata* where Facts, Measures & Dimensional Hierarchies are accessed using XQuery / XPath expressions. These *access paths* into *primary data* may be defined straightforward in terms of XPath locations steps or might as well be more complex algorithmic XQuery expressions.

This approach offers three advantages, first and foremost we use *one* well defined interface to access both metadata and primary data. Second we may use the information coded in our metadata layer to generate the graphical user interface and provide the user a visual exploration tool to analyse his data. Third we may use the XQuery expressions stored in our metadata to automatically expand parts of our queries according to our metadata. If, for example, the user wanted to group the data based on the hour the sale took place we could either explicitly model a hierarchy that contains *hour members* (Houro, ..., Hour23) or we could implicitly model the *hour members* as a XQuery function i.e.`fn:hours-from-dateTime(facts/pathTo/dateTime)`. In addition this would offer us the possibility to perform some very basic schema integration tasks, although it is not very likely that this can be done with sufficient performance for real time analysis.

2.2 OLAP Extensions for XQuery

For XML to meet up with Relational OLAP several query language solutions have been proposed, among them by Hümmer et. al. [3] the formulation of OLAP queries as XML-fragments sent to a database server that in return sends a XML-fragment containing the inquired analysis data. However solutions that build upon XML databases and a standardized query language (e.g. XQuery) are rarely used.

We will however, show that using XML in conjunction with XQuery will be able to meet up with the capabilities provided by relational OLAP implementations. In order to do so we first gather the required tools and operators known from relational OLAP and adapt them for XQuery. As for today in [10] there exists a working draft that provides all operations necessary for OLAP. In the following section we will concisely discuss those concepts and how they resemble to the relational OLAP analysis workflow.

2.2.1 Heterogenous & Imprecise data

In contrast to Relational OLAP where missing, incomplete and de-normalized data is avoided at the schema layer, XML offers a whole new perspective to store and query non conventional data. Heterogenous schemata possibly derived from numerous data sources may, to some extent, be queried without prior normalization. If we look at the simple XML tree storing sales information from two example bookstores (Figure 1), we notice the heterogenous structure of the XML file, where a book has one or more authors.

2.2.2 Value Based Grouping

Value Based Grouping is described in [10, Section 3.8.7]. The XQuery `group by` works similar to the standard SQL operator as it only allows grouping based on (XQuery) atomic values.

A simple query, getting the per author, per date sales of books in traditional OLAP has to take the 1:n relation between book and author into account. XQuery allows us to do this in a much more convenient way. We "relax" the grouping criterion on author by skipping the intermediate `authors`-Node, using the following group by clause:

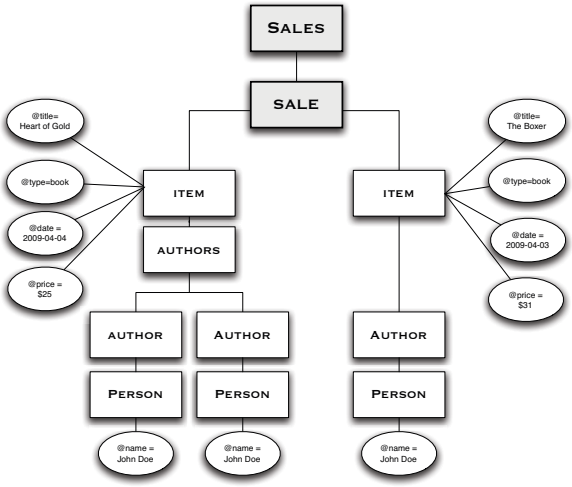


Figure 1: Example XML Document

Example 1

```
group by /sale/item[@type="book"]//author, /sale/item[@type="book"]/@date
```

which as a result yields three tuples for the two
books, each one uniquely defined by its unique (Author, Date)-pairs. This might seem counter intuitive for our analysis requirement as the first book ("Heart of Gold") would return two distinct results for either author. This is why we need to define a concept of uniqueness for any given node with respect to the measure under consideration. To achieve a both flexible and easy to use solution we propose a modification to the relational grouping operator that adds adjustable granularity:
`group by` $/step_0/.../step_n$ [`granularity` $/step_0/.../step_n$] It defines which node is used as the unique group identifier.

The modification `granularity` $/step_0/.../step_n$ resembles to the traditional `group by`.

Reclaiming **Example 1** we would define the granularity level by:

Example 2

```
group by /sale/item[@type="book"]//author, /sale/item[@type="book"]/@date
  granularity /sale/item
```

which would result in only two groups for the nodes satisfying `/sale/item[@type="Book"]`. Please note that the granularity definition does not take the heterogenous structure of the author nodes into account, as we only take their (common) parent nodes into account. This extension is similar to the tree pattern relaxations proposed by [11] and exploits the XML structure in a straight forward way.

Using this concept all subsequent aggregate functions like `sum`, `avg` object to summarizability and return intelligible results to the user.

2.2.3 Partitioning: The Window Clause

With the introduction of the window clause in [10, Section 3.8.4] we gain the ability to perform *time series analyses*. XQuery 1.1 defines two kinds of window clauses that allow us to iterate over a defined sequence of items. We can use both, overlapping (sliding) and exclusive (tumbling) windows to compute Measures.

A typical scenario might be generating an moving three day average of the per author, per date sales from Example 1. In order to so we partition the the resulting items using the window clause:

```
for tumbling window $w in $resultsFromExample1
                                    (: $resultsFromExample1 contains: SUM of sales per
                                      day per author :)
    start at $s when fn:true()      (: set $s to the first item :)
    only end at $e when $e - $s eq 2 (: put the subsequent 2 items into $w :)
                                    (: $w now contains 3 subsequent cells per day :)
return
 <cell info="3 day sales average for $s/author from {$s//date} to {$e//date}">
   { avg($w) }
 </cell>
```

2.3 Group by Cube & Rollup

As the Cube & Rollup operations are special cases of concatenated GROUP BY clauses we can, for the moment, simply mimic this behaviour: For a Rollup over n-Attributes we calculate the (n+1)-combinations of

$$(A_1, \ldots, A_n) \ldots (A_1, \ldots, A_{n-1}), \ldots (A_1, A_2), (A_1), ()$$

to compute each super aggregate for the given dataset and union the respective results. As manually formulating this (n+1)-UNIONS would be very lengthy we add the convenience operator GROUP BY ROLLUP that automatically expands to (n+1)-UNIONS.

The same holds for the GROUP BY CUBE operation that expands to all 2^n possible concatenations of grouping attributes.

2.4 Full-text operations

Besides traditional OLAP we think Full-Text analysis capabilities are crucial to suite warehousing needs and provide a convenient way to analyze non conventional textual data. Full-Text Operations could be used to:

(1) Group the nodes based on their textual content, based on ad-hoc user defined criteria.

(2) Provide custom consolidation paths in conjunction with a thesaurus, based on the specified nodes textual contents.

(3) Criteria for Slice & Dice operation to confine the analyzed nodes.

Consider for example the following XQuery code snippet querying real world patent data:

```
group by  $x//case-file-statement/text[. ftcontains "database"
      and(. ftcontains "biologic" or "chemical")]
```

In case of grouping the following code partitions our nodes into two distinct spaces, each containing the nodes that met the requirements.
A slightly more advanced approach could be grouping the nodes based on their relevance ranking:

```
let $score := score
group by  fn:floor($score * 10)
where $x//case-file-statement/text[. ftcontains "database"
      and(. ftcontains "biological" or "chemical")]
```

Where each group consists of members with a similar relevance for the given keywords. Although these ideas do not conform to the traditional OLAP tasks they might offer new possibilities during the analysis of data.

3 Conclusion

We have shown that XML/XQuery is suitable for OLAP operations and is able to meet up with relational OLAP in terms of analysis needs.

Thanks to XQuery's capabilities as a programming language OLAP operations may – at least in theory – be as complex as the user wants them to be, in both metadata modelling and primary data analysis.

With *XQuery enhanced Metadata* we are able to allow more complex metadata definitions that may be used to easily access complex underlying primary data.

The first steps will be taken with the implementation of the XQuery 1.1 Draft. After these initial steps we will need benchmarks to identify if there are typical access patterns to work on further optimizations. The second step will include building a metadata driven GUI to visually explore OLAP aggregates and provide the user with an appropriate tool for querying this data.

Another big challenge for future work will be the integration of materialized cubes, that by design, are not organized hierarchically and thus require special treatment in serialization.

References

[1] H. Wang, J. Li, Z. He, and H. Gao, "Olap for xml data," *Computer and Information Technology*, Jan 2005.

[2] B. Park, H. Han, and I. Song, "Xml-olap: A multidimensional analysis framework for xml warehouses," *Lecture Notes in Computer Science*, Jan 2005.

[3] W. Hümmer, A. Bauer, and G. Harde, "Xcube: Xml for data warehouses," *... 6th ACM international workshop on Data warehousing and OLAP*, Jan 2003.

[4] M. Jensen, T. Møller, and T. Pedersen, "Specifying olap cubes on xml data," *Journal of Intelligent Information Systems*, Jan 2001.

[5] A. M. Tjoa, A. Rauber, P. Tomsich, and R. Wagner, "Olap of the future," pp. 153–166, 2007.

[6] T. Pedersen, "Warehousing the world: a few remaining challenges," in *Proceedings of the ACM tenth international workshop on Data warehousing and OLAP*, pp. 101–102, ACM New York, NY, USA, 2007.

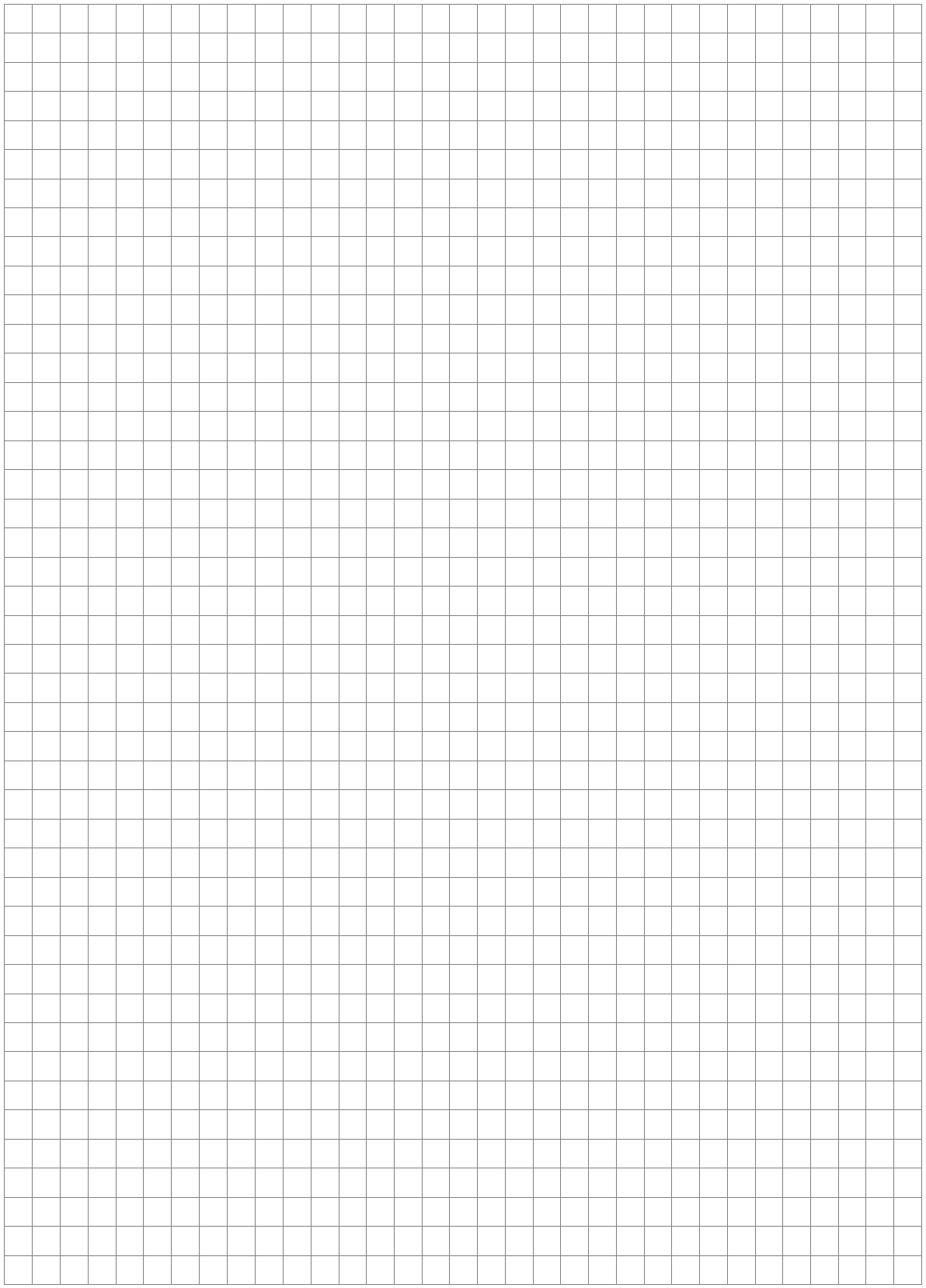
[7] S. Mansmann, *Extending the OLAP Technology to Handle Non-Conventional and Complex Data*. PhD thesis, Universität Konstanz, Universitätsstr. 10, 78457 Konstanz, 2008.

[8] W3C Recommendation, "XQuery 1.0: An XML Query Language," 23 January 2007.

[9] R. Kimball, *The data warehouse toolkit: practical techniques for building dimensional data warehouses*. John Wiley & Sons, Inc. New York, NY, USA, 1996.

[10] W3C Draft, "XQuery 1.1 W3C Working Draft," 3 December 2008.

[11] N. Wiwatwattana, H. Jagadish, and L. Lakshmanan, "X3: A cube operator for xml olap," *23rd International Conference on Data Engineering (ICDE 07)*, Jan 2007.



Modellgetriebene semi-automatische Bereitstellung von Analytischen Informationssystemen

Yvette Teiken

30. April 2009

Zusammenfassung

Analytische Informationssysteme werden von Organisationen immer häufiger zur Informationsversorgung und Entscheidungsunterstützung eingesetzt [Fri07]. Vielfach werden hierfür Analytische Informationssysteme mit zu Grunde liegender Informationslogistik (IL)-Infrastruktur verwendet. IL-Infrastruktur beschreibt den bei Analytischen Informationssystemen über Insellösungen in Fachabteilungen hinaus gehenden Teil. Hierzu gehören ein integrierter Datenbestand sowie eine Analyse-Sicht z.B. in Form von OLAP-Servern. Notwendige Schritte zur Bereitstellung von IL-Infrastruktur sind mit einem hohen Aufwand verbunden und somit kostenintensiv. In diesem Artikel wird ein Ansatz zur einfacheren Bereitstellung von IL-Infrastruktur vorgestellt. Dies soll erreicht werden, indem Teile generativ erzeugt werden. Hierbei wird eine bedarfsgetriebene Analyse vorausgesetzt. Realisiert werden soll ein Top-Down-Vorgehen, das mittels Domänen-spezifischer Modellierung (DSM) erreicht wird. In diesem Vorgehen sollen Aspekte von IL-Infrastruktur mittels einer Familie von Domänenspezifischen Sprachen (DSL) beschrieben werden. Durch dieses Vorgehen können Experten der jeweiligen Domäne Teile der IL-Infrastruktur entwerfen und erzeugen lassen. Generiert werden u.a. Kennzahlen, multidimensionale Sicht, Ladeprozesse, Sicherheit und auch dynamische Berichte.

1 Motivation/Einführung

Organisationen stehen oftmals unter ständig steigendem Kostendruck. Um diesem Kostendruck gerecht zu werden, spielt ein effektives Controlling eine immer wichtigere Rolle. Zunehmend wird dabei auf IT-Unterstützung gesetzt. Diese Art von Systemen werden als Analytische Informationssysteme bezeichnet. Die Analytischen IS werden nach [CG04] in verschiedene Reifegrade unterteilt, die sich in ihrer Integrationshöhe unterscheiden. Für eine Organisation ist der Übergang von fachspezifischen Lösungen hin zu organisationsweiten Lösungen mit hohem Aufwand und somit hohen Kosten verbunden. Hier muss die Integration verschiedener Fachbereiche vorgenommen werden. Dazu gehört der Aufbau eines übergreifenden Metadatenmanagements, die Integration externer Daten und die Definition von Analyseszenarien. Nach [BG05] sind zwei verschiedene Vorgehen zu unterscheiden, das Informationsbedarf-getriebene und das Daten-getriebene Vorgehen ([Muc06]). Bei einem Informationsbedarf-getriebenen Vorgehen wird die Entwicklung des Analytischen Informationssystems auf Basis der inhaltlichen Anforderungen spezifiziert. Diese Art des Vorgehens wird als Top-Down-Vorgehen bezeichnet. Beim Top-Down-Vorgehen stellt die Analyse des Informationsbedarf ein Kernaufgabe dar.

In diesem Beitrag wird ein Top-Down-Vorgehen für die modellgetriebene semi-automatische Bereitstellung von Analytischen Informationsystemen vorgestellt. Hierfür wird im nächsten Kapitel erst die Idee und dann der Lösungsansatz beschrieben.

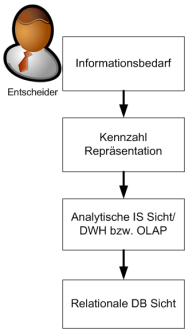


Abbildung 1: Sichten im TopDown Vorgehen

2 Lösungsansatz

Um aus Informationsbedarf eine IL-Infrastruktur eines Analystischen IS abzuleiten zu können, werden verschiedene Zwischenschritte absolviert. Diese Schritte sind schematisch in der Abbildung 1 abgebildet. Ein Informationsbedarf in einem Analytischen IS kann mittels relevanter Kennzahlen beschrieben werden. Dieser Informationsbedarf wird von einem Entscheider formuliert. Dieser richtet sich nach den aktuellen Gegebenheiten der Organisation. Gegebenenfalls werden relevante Grenzen oder Schwellwerte formuliert. Bei einer Kennzahl kann es sich um einfache Kennzahlen handeln, die sich zum Beispiel mittels einer Division beschreiben lassen. Dies ist bei der betrieblichen Kennzahl Umsatzrendite der Fall ($Umsatzrendite = \frac{Gewinn}{Umsatz}$). Ein Informationsbedarf kann jedoch auch durch eine komplexe Kennzahl beschreiben werden. Zu diesen komplexen Kennzahlen gehören epideminologische Kennzahlen, wie sie im Rahmen der Arbeit des EKN[1] verwendet werden. Hier wird mit Erwartungswerten gearbeitet, die Konstrukte wie eine Standardbevölkerung voraussetzen, wie zum Beispiel *Erwartete Fallzahlen* ($e = \sum_{a \in A} R_a * n_a$). Diese Kennzahl stellt eine Art Prognose der zu erwartenden Fallzahlen dar, dieser werden auf Basis einer Bevölkerungsprognose, Fallzahlen und einer externen Bevölkerung ermittelt.

Kennzahlen lassen sich formalisieren. Auf Grundlage der formalisierten Kennzahlen ist eine Formalisierung des Informationsbedarfs möglich. Werden diese formalisierten Kennzahlen mit Dimensionsinformationen angereichert, so lässt sich eine Analytische IS-Sicht in Form einer DWH Sicht erzeugen. Hierbei entsteht implizit ein übergreifendes Metadatenmanagement.

Aus der Analytischen Sicht lassen sich danach die relationalen Strukturen in Form von Datenbank-Sichten erzeugen.

2.1 Konkreter Lösungsansatz

In dem in Abbildung 2 skizzierten Ansatz lassen sich unterschiedliche Rollen bzw. Teilaufgaben identifizieren. So werden Kennzahlen zum Beispiel von Entscheidern modelliert, während Metadaten von Modellierungsexperten beschrieben werden. Aus diesem Grunde erscheint es sinnvoll, die Bereitstellung von Analytischen IS in verschiedene Teilaspekte zu zerlegen. Diese Teilaspekte können dann von den jeweiligen Experten/Rollen beschrieben werden, so dass am Ende eine integrierte Gesamtsicht entsteht. Die zu modellierenden Teilaspekte unterscheiden sich durch Vorwissen/Vorbedingungen und unterschiedlich hohes Abstraktionsniveau. Deswegen ist es nicht sinnvoll, diese Aspekte alle auf die gleiche Art zu beschreiben. Deshalb soll ein DSM basierter Ansatz für die Erzeugung von Analytischen IS verfolgt werden (siehe auch [KT08]). Hierfür soll eine Familie von grafischen und textuellen DSLs entwickelt werden, ähnlich wie es in [WK06] durchgeführt worden ist. Die Verwendung von DSLs hat den Vorteil, dass Modelle von Fachexperten in ihrer eigenen Begriffswelt erstellt werden können und keine zusätzlichen Sprachen vom Fachexperten für die DWH-Konzeption, wie zum Beispiel UML, erlernt werden müssen. UML

[1] http://www.krebsregister-niedersachsen.de/

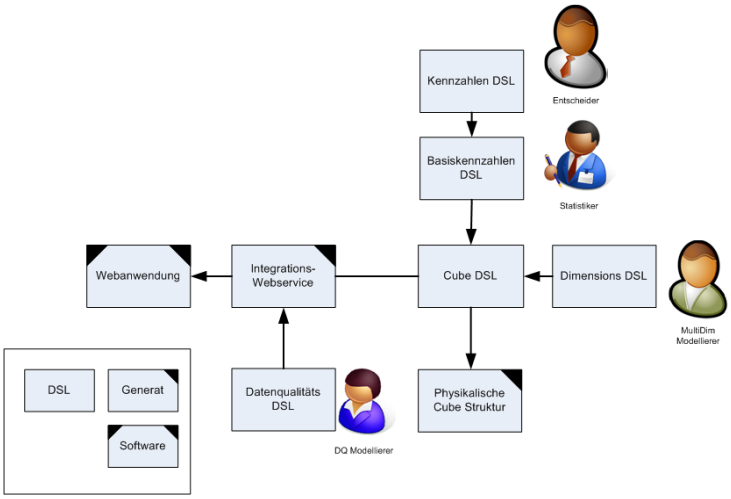


Abbildung 2: Familie von DSLs und deren Generate

ist zwar ebenfalls eine DSL, allerdings für den Einsatzbereich Softwareentwicklung, und somit nur bedingt für DWH-Fachexperten geeignet. Beispiele für Modellierungssprachen im Bereich konzeptueller Modellierung stellen ADAPT [Bul96] und ME/R [SBHD99] dar. Sie dienen als Grundlage der späteren grafischen DSL.

In der Abbildung 2 sind die mögliche Familie von DSLs und deren Generate abgebildet. Die Kennzahlen DSL dient zur Beschreibung von Kennzahlen und deren Verknüpfung miteinander. Diese DSL soll möglichst einfach gehalten sein und sich rein auf die Kennzahl-Verknüpfung beschränken. Die Basiskennzahlen DSL hingegen ist komplexer. Sie beschreibt die multidimensionale Annotation von Kennzahlen. Ziel der Kennzahlmodellierung ist die Überführung in eine multidiemsionale Struktur. Je nach Art der Organisation und gegebenen Umständen kann eine Kennzahl unterschiedliche Dimensionen aufweisen. Zum Beispiel kann die Kennzahl *Gewinn* bei einer Supermarktkette andere Ausprägungen haben als bei einem Krankenhaus. Eine Supermarktkette wird ihren Gewinn nach OPS- oder ICD-Dimensionen annotieren. Es kann auch vorkommen, dass die gleiche Dimension verwendet wird, diese jedoch unterschiedliche Bedeutungen hat. So kann eine geografische Dimension bei einer Supermarktkette das Fillialnetz beschreiben während bei einem Krankenhaus das Einzugsgebiet gemeint ist. In dem hier vorgestellten Ansatz wird diese Art von Semantik nicht berücksichtigt, da beide Bedeutungen das gleiche schematische Konzept beschreiben.

Die Cube DSL beschreibt die Kennzahl aus multidimensionaler Sicht. Es wird die multidimensionale Modellierungssprache ADAPT verwendet. Sie dient dazu, eine einfache Überführung in eine multidimensionale Sicht zu ermöglichen. Weiterhin wird hier die mögliche Verknüpfung von Cubes auf Zellebene beschrieben. Dies wird benötigt, um die Operationen, die mit den Kennzahlen beschrieben sind, auf Cube Ebene zu realisieren. Dies ist vom Prinzip ähnlich, wie das in [GKS09] vorgeschlagene Vorgehen.

Für die Beschreibung von Dimensionen existiert ebenfalls eine DSL. Zwar besteht in ADAPT und anderen Modellierungssprachen die Möglichkeit, Dimensionen zu beschreiben, doch da die Modelle sehr komplex sind, empfiehlt sich die Verwendung einer eigenen DSL. In dieser DSL ist es möglich, komplexe Dimensionen mit parallelen Hierarchien mit vielen hunderten Knoten zu Elementen zu modellieren. Hierfür müssen besondere Formen der Eingabe und Darstellung beachtet werden. Die in der Dimensions DSL beschrieben Dimensionen werden in der Cube DSL referenziert.

Die alleinige Bereitstellung einer DSL oder einer Familie von DSLs stellt allerdings noch keinen hohen Mehrwert für die Realisierung eines Projekts dar. Dies geschieht erst dann, wenn die mit Hilfe einer DSL erstellten Modelle semi-automatisch zu neuen Modellen oder zu Generaten,

wie sie in Abbildung 2 dargestellt sind, transformiert werden. Dabei soll es auch möglich sein, aus einem Modell mehrere Artefakte für unterschiedliche Einsatzzwecke zu erzeugen. Dieser Ansatz wird in der Softwareentwicklung als MDSD bezeichnet (siehe [SVE07]). Dieser soll im Rahmen der Arbeit auf die Bereitstellung von Informationslogistik übertragen werden, um die oben genannten Ziele zu realisieren.

Sichtbare Elemente einer DSL ist die konkrete Syntax. Die eigentliche Bedeutung bzw. Interpretation wird in der abstrakten Syntax der DSL spezifiziert. Diese abstrakte Syntax wird mittels eines Metamodells beschrieben. Werden, wie in diesem Ansatz vorgestellt, mehrere DSLs realisiert, die miteinander interagieren so stellt sich die Frage ob ein gemeinsames oder getrennte Metamodelle verwendet werden sollen (siehe [HCW07]). Da es in dem in dieser Arbeit beschriebenen Ansatz viele überschneidende Elemente gibt, wird ein gemeinsames Metamodell verwendet. Dies bedeutet, dass Modell-zu-Modell-Transformationen Modell-inhärent sind. Mit Hilfe dieser Transformationen lassen sich andere Modelle oder Artefakte erzeugen. In Abbildung 2 sind neben der Familie der DSLs ebenfalls die möglichen Generate abgebildet.

Oftmals wird ein multidimensionales Schema in einer relationalen Datenbank abgelegt. Dies wird mit dem Generat der physikalischen Cube Struktur beschrieben. Für die Repräsentation eines multidimensionalen Schemas existieren verschiedene relationale Abbildungen wie zum Beispiel das Snow- oder Star-Schema (siehe [BG04]). Ein anderes Beispiel für eine solche Abbildung ist das MUSTANG Schema aus [KMR03], welches im OFFIS entwickelt worden ist. Dies ist besonders gut geeignet in Bezug auf epidemologische Fragestellungen. Dadurch, dass in diesem Ansatz ein MDSD-Ansatz verwendet wird, können durch das Anwenden anderer Transformationen auf Basis der Cube DSL andere relationale Modelle erzeugt werden. Es muss nur die Transformation von einem kleinen Teil des Metamodells aus spezifiziert werden.

Ist die Datenstruktur bekannt, die innerhalb der DSLs beschrieben worden ist, so ist auch bekannt, wie die Daten aussehen müssen, die für die Auswertungen benötigt werden. Somit ist es möglich, aus der Cube DSL einen Webservice zu erzeugen, der die spezifizierten Daten entgegen nimmt und diese im Analytischen IS ablegt. Dieser Service stellt dann sicher, dass die integrierten Daten korrekt sind. Allerdings bezieht sich Korrektheit nur auf Teilaspekte der Daten wie Datentyp, ungültige Werte oder Minimum und Maximum. Echte Datenqualität, wie zum Beispiel in [Brü08] beschrieben, kann auf diese Weise nicht erreicht werden. Dies wird mittels der Datenqualitäts DSL beschrieben. Weiterhin ist es denkbar dieses Konzept schrittweise zu erweitern, um somit Aspekte wie dynamische Berichte oder Ladeprozesse zu integrieren.

3 Zusammenfassung

Mit dem hier vorgestellten Ansatz lassen sich auf einfache Weise Analytische IS schneller und in höherer Qualität erzeugen. Weiterhin lässt sich durch diesen Ansatz ein Modell-Repository Aufbauen, welches den Modellierungsaufwand über die Zeit reduziert. Durch den erweiterbaren Ansatz können auch weiter gehende Konzepte wie beispielsweise Reporting integriert werden.

Literatur

[BG04] Bauer, Andreas ; Günzel, Holger: *Data-Warehouse-Systeme. Architektur, Entwicklung, Anwendung.* Dpunkt Verlag, 2004. – ISBN 3898642518

[BG05] Burmester, Lars ; Goeken, Matthias: Benutzerorientierter Entwurf von unternehmensweiten Data-Warehouse- Systemen. In: Ferstl, Otto K. (Hrsg.) ; Sinz, Elmar J. (Hrsg.) ; Eckert, Sven (Hrsg.) ; Isselhorst, Tilman (Hrsg.): *Wirtschaftsinformatik*, Physica-Verlag, 2005. – ISBN 3–7908–1574–8, S. 1421–1440

[Brü08] Brüggemann, Stefan: *Proaktives Management von Konsistenzbedingungen im Analytischen Performance Management*. 2008

[Bul96] Bulos, Dan: OLAP database design: A new dimension. In: *Database Programming&Design* Vol. 9(6) (1996)

[CG04] Chamoni, Peter ; Gluchowski, Peter: Integrationstrends bei Business-Intelligence-Systemen, Empirische Untersuchung auf Basis des Business Intelligence Maturity Model. In: *Wirtschaftsinformatik* 2 (2004), S. 119–128

[Fri07] Friedrich, Dirk: Einfach soll es sein - bei hoher Datenqualität. In: *IS - Informationsplattform für Business Applications* 11 (2007), S. 30–35

[GKS09] Gluchowski, Peter ; Kunze, Christian ; Schneider, Christian: A Modeling Tool for Multidimensional Data using the ADAPT Notation. In: *42nd Hawaii International Conference on System Sciences (HICSS-42)*, 2009

[HCW07] Hessellund, Anders ; Czarnecki, Krzysztof ; Wasowski, Andrzej: Guided Development with Multiple Domain-Specific Languages. In: Engels, Gregor (Hrsg.) ; Opdyke, Bill (Hrsg.) ; Schmidt, Douglas C. (Hrsg.) ; Weil, Frank (Hrsg.): *MoDELS* Bd. 4735, Springer, 2007 (Lecture Notes in Computer Science). – ISBN 978–3–540–75208–0, S. 46–60

[KMR03] Koch, Sascha ; Meister, Jürgen ; Rohde, Martin: MUSTANG – A Framework for Statistical Analyses of Multidimensional Data in Public Health. In: *Proceedings of the 17th International Conference Informatics for Environmental Protection*. Cottbus, September 2003, S. 635–642

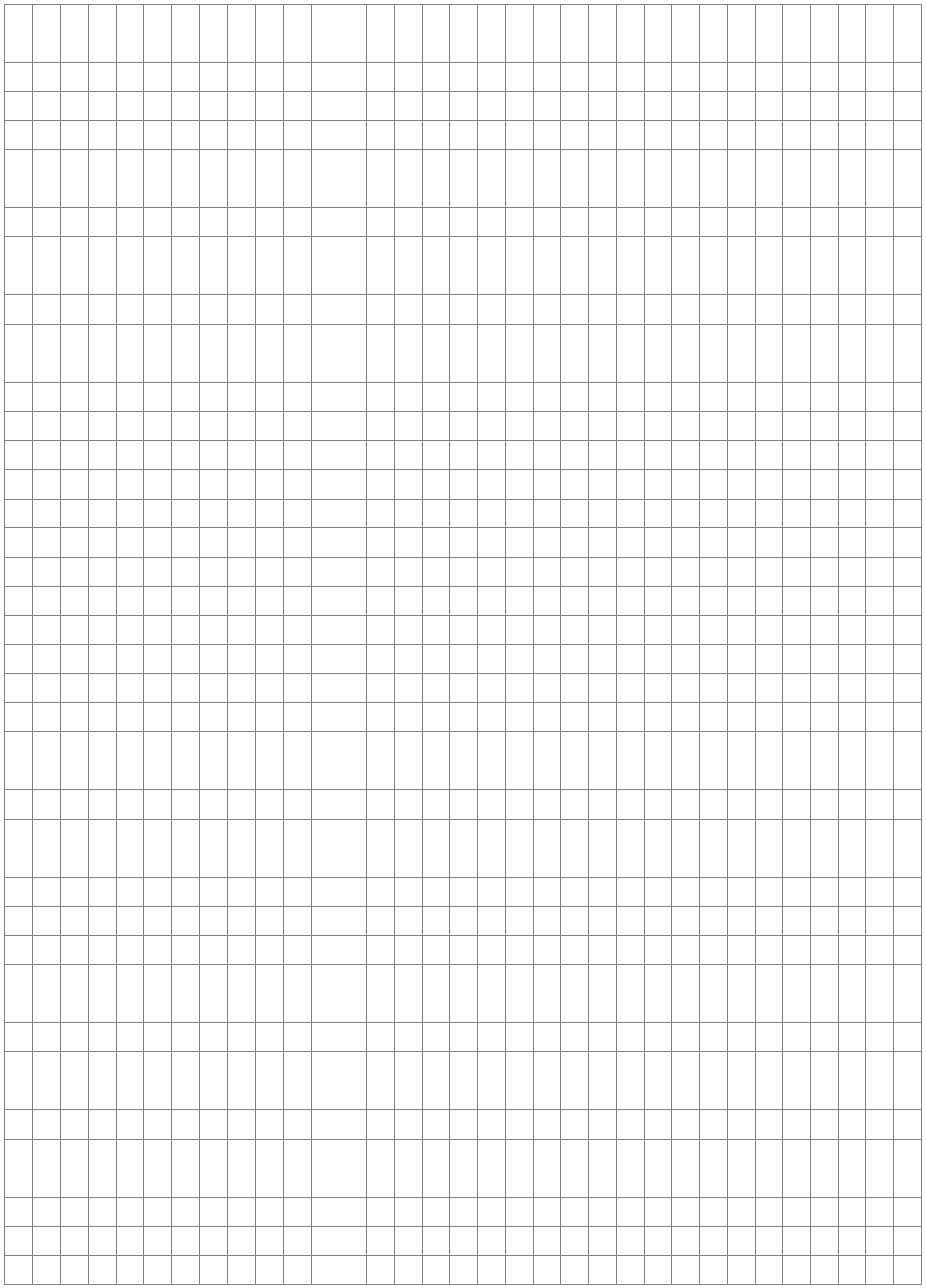
[KT08] Kelly, Steven ; Tolvanen, Juha-Pekka: *Domain-Specific Modeling: Enabling Full Code Generation*. John Wiley & Sons, 2008

[Muc06] *Kapitel* Das Data Warehouse als Datenbasis analytischer Informationssysteme. In: Mucksch, Harry: *Analytische Informationssysteme. Business Intelligence-Technologien und -Anwendungen: Business Intelligence-Technologien und -Anwendungen*. Springer, Berlin, 2006, S. 129–142

[SBHD99] Sapia, Carsten ; Blaschka, Markus ; Höfling, Gabriele ; Dinter, Barbara: Extending the E/R Model for the Multidimensional Paradigm. In: *ER '98: Proceedings of the Workshops on Data Warehousing and Data Mining*. London, UK : Springer-Verlag, 1999. – ISBN 3–540–65690–1, S. 105–116

[SVE07] Stahl, Thomas ; Völter, Markus ; Efftinge, Sven: *Modellgetriebene Softwareentwicklung. Techniken, Engineering, Management*. Dpunkt Verlag, 2007. – ISBN 3898644480

[WK06] Warmer, J. B. ; Kleppe, A. G.: Building a Flexible Software Factory Using Partial Domain Specific Models. In: *Sixth OOPSLA Workshop on Domain-Specific Modeling (DSM'06), Portland, Oregon, USA*. Jyvaskyla : University of Jyvaskyla, October 2006. – ISBN 951–39–2631–1, S. 15–22



Anwendungsspezifische Adaption und Erweiterung von OLAP-Anwendungen auf Basis von MDX

Thomas Hasart, Mario Gleichmann, Ilvio Bruder, Peter Forbrig, Andreas Heuer
IT Science Center Rügen, 18581 Putbus

Zusammenfassung

Die Anforderungen an OLAP-Systeme (Online Analytical Processing) [Pendse, 1998] verschieben sich von z.B. wöchentlichen zu immer kürzeren Aktualisierungsintervallen bis hin zu Auswertungen, die den aktuellen Stand des operativen Systems darstellen. Zugleich vervielfachen sich die auszuwertenden Daten. Dabei werden Funktionalitäten der Frontends wie eine zentrale einfache Schnittstelle, die neben der vom Benutzer programmierbaren Datenanalyse auch Datenmanagementaufgaben wie Import/Export, Backup/Restore, Optimierung und Rechteverwaltung beherrscht, vorausgesetzt.

Vor allem die Abarbeitungsgeschwindigkeit stellt dabei immer noch eine große Herausforderung dar. Am Markt sind zurzeit zwei Trends zu beobachten. Zum einen entwickeln Anbieter eigene In-Memory-OLAP-Datenbanken um ihre Anwendung zu beschleunigen. Zum anderen scheuen viele Anbieter den enormen Entwicklungsaufwand für eigene OLAP-Backends und setzen deshalb Datenbanken der etablierten Anbieter ein (Ansätze u.a. in [Tjoa et al., 2007]). Hier hat sich vor allem MDX als Quasi-Standard durchgesetzt.

In diesem Paper werden wir ausgehend von den Anforderungen aktueller OLAP-Anwendungen, MDX (Multi-Dimensional Expressions [Willand, 2004] [Microsoft, 1999]) und seine Splashing-Möglichkeiten analysieren und notwendige Erweiterungen herausstellen.

1 Einführung

Im Projekt Monicca zum Thema "Model-Driven Account Management in Data Warehouse Umgebungen", geht es darum, OLAP-Anwendungen von den Nutzerschnittstellen bis hin zur funktionalen Ebene adaptierbar zu gestalten. Dabei wird für Key Accounts [Sidow, 2007] ein Werkzeug entwickelt, mit dem diese ihre Kunden aufgrund verschiedener Daten aus Data Warehouses oder anderer Quellen besondere Angebote unterbreiten können. Hierdurch wird den Key Account Managern eine spezielle Schnittstelle auf die Daten der Schlüsselkunden bereitgestellt. Diese ermöglicht es, in Anwendungen umfassendere, angepasstere und geeignetere Analysefunktionalität anzubieten. Das allgemeine Problem der Personalisierung von OLAP-Anwendungen wird mittels Techniken im Bereich der Sichtdefinition und Metadaten-gestützte Verwaltung von OLAP-Data Cubes gelöst. Zur Erzeugung der Nutzersichten wird eine Modellsprache entwickelt, die nötige OLAP-Operationen für die Sichten über Verknüpfungen bis hin zur Definition der Ausgabe und der folgenden Interaktion beschreiben kann. Dieser Modell-gestützte Ansatz ist die Grundlage für das "End-User Development" mit dem Ziel, die Anwendung personalisierbar zu gestalten.

2 Architektur

Die Architekturübersicht in Abbildung 1 zeigt die verschiedenen zur Verfügung stehenden Interfaces. Der BI-Client nutzt eine proprietäre Schnittstelle, die direkt auf den OLAP-Processor zugreift. Externe Anwendungen wie Microsoft Excel oder Crystal Reports können XML/A (XML for Analyses [Simba-Technologies-Inc, 2006]) oder ODBO (OLE DB for OLAP [Microsoft, 1997]) nutzen. Beide Interfaces nutzen den MDX-Processor der die Anfragen an den OLAP-Processor weiterleitet. Als Vorteil von ODBO ist vor allem die breite Unterstützung durch die Anbieter zu nennen. Dafür ist ODBO auf die Windows Plattform beschränkt. Bei XML/A sieht es genau anderes herum aus. Dieser Web-Service Standard ist plattformunabhängig. XML/A wird

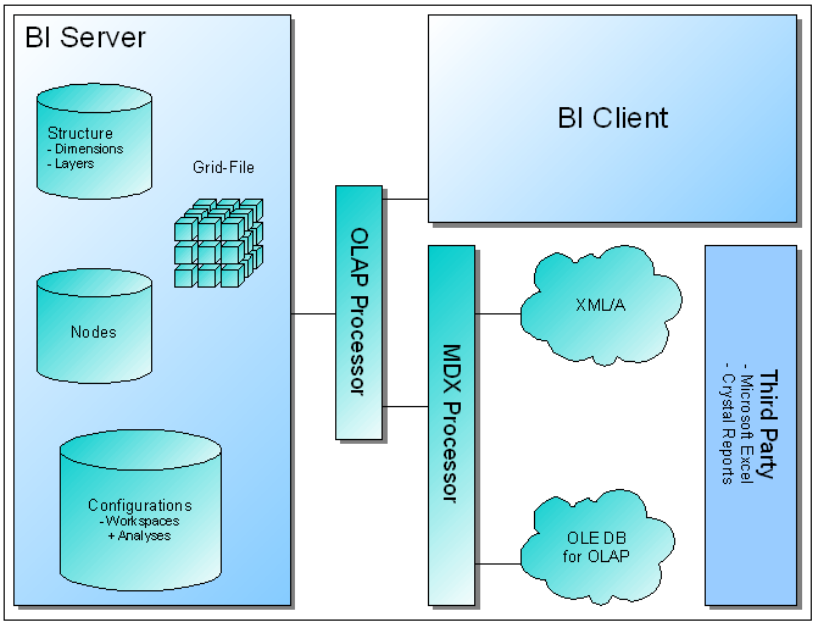


Abbildung 1: Interface Integrationsarchitektur

aber nur zögerlich von den Anbietern unterstützt weil vermutet wird, dass der XML-Overhead und der damit einhergehende Performanceverlust, Microsoft zu einer Änderung der Spezifikation veranlassen könnte. Das auf beiden Schnittstellen verwendete MDX wird nicht von allen Anbietern vollständig implementiert. So fehlen vereinzelt Prozeduren oder Prozeduren führen zu anderen Ergebnissen so dass einzelne Frontend-Anbieter nur Bindungen zu bestimmten Backends zulassen.

3 Splashing/Allocation

OLAP ermöglicht die strukturierte Analyse mehrdimensionaler Daten. Jede Dimension kann mehrere Aggregationsebenen enthalten. Beim Zusammenrechnen der Werte unterer Ebenen zur nächst höheren, existieren teilweise komplexe Abhängigkeiten. Neben der einfachen Summe und dem Durchschnitt ergibt sich nicht selten der Gesamtwert durch Gewichtung der Einzelwerte in Abhängigkeit zu einer anderen Kennzahl. Auch sind Aggregationsmuster Ebenen spezifisch so dass z.B. auf der Ebene der Monate anders aggregiert wird als auf der Ebene der Tage. Neben diesem lesenden Zugriff gewinnt aber immer mehr der schreibende Zugriff, der für die Planung notwendig ist, an Bedeutung. Bei der Planung wird oft vom Nutzer für einen bestimmten Zeitraum ein zu erreichender Gesamtwert vorgegeben. z.B.: 110% des Vorjahreswertes. Dieser Jahreswert muss nun vom System auf die darunter liegenden Ebene (z.B.: Monate oder Quartale) und von dieser weiter bis auf die unterste Ebene verteilt werden. Es müssen die Berechnungen der Aggregation also reversiert werden und deren Teilbeträge auf die Einzelwerte aufgeteilt werden. Diese Aufteilung wird Splashing oder Allocation genannt. Diese automatische Verteilung verursacht zahlreiche Probleme, die im folgenden Abschnitt näher erläutert werden sollen.

4 Probleme beim Splashing

Gelegentlich ist eine Änderung von mehreren Kennzahlen gleichzeitig notwendig, insbesondere wenn Kennzahlen voneinander abhängig sind. Dazu werden die internen Berechnungsergebnisse

der einen Kennzahl zur Änderung der abhängigen Kennzahl benötigt. Ein weiterer Aktualisierungszyklus würde nicht die gleichen Ergebnisse bringen, da die Ausgangswerte ja schon im ersten Durchlauf geändert wurden. Ein Konflikt der sich nur durch gleichzeitiges Setzen aller Kennzahlen lösen ließe.

Wird auf einen Knoten dessen Wert NULL ist verteilt kann dies leicht zum Sprengen der Speichergrenzen führen, da eine Verteilung wohl allen darunter liegenden Knoten einen Wert zu weisen wird, fortgesetzt bis in die unterste Ebene. Eine performante OLAP Lösung macht sich aber gerade zu Nutzen, dass in den seltensten Fällen alle Kombinationsmöglichkeiten Werte enthalten. An einem Beispiel soll dies verdeutlicht werden. Es sei ein mehrdimensionaler Würfel (Cube) mit den Dimensionen Zeit, Artikel, Kunde und Produktionsstandort gegeben. Es wird ein neuer Artikel A2 in die Artikelstruktur unter Oberwarengruppe OW1 und Warengruppe W1 eingefügt. Für diesen Artikel wird für das nächste Jahr 2010 eine Stückzahl von 500 angenommen und dieser Wert in der Kennzahl Menge im Schnittpunkt OW1/W1/A2, 2010 eingefügt. Ein Datenpunkt im Raum ist aber nur valid, wenn ihm ein diskreter Punkt auf der untersten Ebene jeder Dimension zu geordnet wird. Dies bedeutet, dass bei einer Ebenenstruktur Jahr, Monat, Tag in der Dimension Zeit der Wert 500 auf die Tage des Jahres aufgeteilt werden muss. Da es keine Vorgaben gibt werden nun 500/365 = 1,37 Stück auf jeden Tag des Jahres 2010 aufgeteilt. Für jeden dieser neuen Knoten müssen nun auch Knoten auf den anderen Dimensionen definiert sein. Bei vielleicht 500 Kunden erhält jeder Kunde an jedem Tag des Jahres 1,37/500 = 0,003 Stück, von denen an jedem der vielleicht 10 Produktionsstandorte 0,003/10 = 0,0003 Stück produziert werden. Rein mathematisch unproblematisch ergeben sich also 365x500x10 = 1.825.000 neue Punkte. Betrachtet man nun aber gängige Cubes, die zwischen 10 und 20 Dimensionen besitzen von denen z.B. nur die Kundendimension bei Kommunikationsunternehmen schon mehrere Millionen Kunden enthalten kann, kann man sich schnell die Ausmaße solch einer Verteilung vorstellen. Des Weiteren muss man sich verdeutlichen, dass so eine Verteilung auch nicht besonders hilfreich ist, da in der Regel pro Dimension – die Zeit Dimension einmal ausgenommen – nur ein logisch richtiger Knoten existiert. Um dieser Problematik zu begegnen gibt es vielfältige Möglichkeiten. So wäre es denkbar fehlende Verteilungsmuster aus anderen Artikeln zu nehmen – eine sogenannte Verschwägerung – oder Werte aus der Vergangenheit zu nutzen. Auch könnten Vorgaben wie nur auf den ersten, den letzten oder einen bestimmten Knoten definiert werden.

Problematisch ist auch die Berechnung von Kennzahlen auf NULL Werten. Teilweise werden globale Vorgaben wie z.B. Umrechnungsfaktoren für Produktionseinheiten benötigt. Diese können nicht als eigenständige Kennzahlen zur Verfügung gestellt werden da diese dann bei neuen Artikeln nicht existieren würden. Und demzufolge damit auch nicht gerechnet werden kann. Insbesondere bei Rechnungen mit Währungen deren Kurs sich zeitlich ändert sind weitere Schwierigkeiten zu erwarten.

5 Splashing in MDX

Nicht nur das die Serverimplementierung entsprechende Funktionalitäten zur Verfügung stellen muss um diese Verteilungen vornehmen zu können, muss auch der Nutzer die Möglichkeit haben seine Vorgaben dem System mitzuteilen. MDX kennt folgende `UPDATE CUBE` Anweisung um eine Verteilung durchzuführen:

```
UPDATE [CUBE] Cube_Name SET <update clause> [, <update clause> ...n]

  <update clause> ::= Tuple_Expression[.VALUE]=New_Value
   [     NO_ALLOCATION
       | USE_EQUAL_ALLOCATION
       | USE_EQUAL_INCREMENT
       | USE_WEIGHTED_ALLOCATION [BY Weight_Expression]
       | USE_WEIGHTED_INCREMENT [BY Weight_Expression]   ]
```

Dabei werden folgende Splashing-Möglichkeiten unterschieden:

USE_EQUAL_ALLOCATION Jeder Blattzelle, die zur aktualisierten Zelle beiträgt, wird **derselbe** Wert zugewiesen, der auf dem folgenden Ausdruck basiert:

```
<leaf cell value> = <New Value> / Count (leaf cells that are contained in <tuple>)
```

Das entspricht der einfachen Verteilung über die Anzahl der Werte, z.B. würde 1/12 des Jahreswertes auf jeden Monat verteilt werden.

USE_EQUAL_INCREMENT Jede Blattzelle, die zur aktualisierten Zelle beiträgt, wird entsprechend dem folgenden Ausdruck geändert:

```
<leaf cell value> = <leaf cell value> + (<New Value> - <existing value>) /
    Count (leaf cells contained in <tuple>)
```

Dabei wird die Differenz zwischen neuem und alten Gesamtwert durch die Anzahl geteilt und auf die Blattwerte aufgetragen.

USE_WEIGHTED_ALLOCATION Jeder Blattzelle, die zur aktualisierten Zelle beiträgt, wird **derselbe** Wert zugewiesen, der auf dem folgenden Ausdruck basiert:

```
<leaf cell value> = <New Value> * Weight_Expression
```

Der neue Wert wird mittels Weight_Expression, einem Wert zwischen 0 und 1, der im Standardfall den Anteil des alten Blattwertes vom alten Gesamtwert darstellt, gewichtet und auf die Blattzellen verteilt.

USE_WEIGHTED_INCREMENT Jede Blattzelle, die zur aktualisierten Zelle beiträgt, wird entsprechend dem folgenden Ausdruck geändert:

```
<leaf cell value> = <leaf cell value> +
    (<New Value> - <existing value>)  * Weight_Expression
```

Jeder Blattwert wird mit der durch Weight_Expression – standardmäßig ebenfalls der Anteil des alten Blattwertes vom alten Gesamtwert – gewichteten Differenz aus neuem und altem Gesamtwert erweitert.

6 Lösungsansätze

Defizite von MDX lassen sich durch Spracherweiterungen [Vaisman et al., 2002] lösen. Bei den angeführten Splashing Statements kann innerhalb einer Update Klausel nur eine Kennzahl geändert werden. Dies lässt sich durch folgende Erweiterung beheben.

```
<update clause> ::= <tuple clause> [, <tuple clause>...]

<tuple clause> ::= Tuple_Expression[.VALUE]= New_Value
    [     NO_ALLOCATION
        | USE_EQUAL_ALLOCATION
        | USE_EQUAL_INCREMENT
        | USE_WEIGHTED_ALLOCATION [BY Weight_Expression]
        | USE_WEIGHTED_INCREMENT [BY Weight_Expression]   ]
```

Durch die Angabe mehrerer Kennzahlen können nun gleichzeitig auch mehrere geändert werden.

Ein weiteres Manko in den `UPDATE CUBE` Statements ist, dass keine speziellen Vorgaben für die Behandlung von NULL Werten, also ob auf diesen verteilt werden soll oder nicht und wie, angegeben werden können. Eine denkbare Erweiterung wäre:

```
ON_NULL_VALUES <allocation clause> [, <allocation clause>...]

<allocation clause> ::=
    [     USE_ALL
        | USE_LAST
        | USE x
        | USE_PARENT
        | USE_PAST
        | USE_NONE   ]
```

In dieser Form können nun mehrere allocation clauses angegeben werden, die in der angegebenen Reihenfolge Verwendung finden. Z.B: würde

```
ON_NULL_VALUES USE_PARENT, USE_PAST, USE 0
```

bei fehlendem Verteilungsmuster zuerst beim Elternknoten nach einer Verteilung geschaut werden. Wird dort keine Verteilung gefunden wird in der Vergangenheit gesucht. Sind dort auch keine Vorgaben zu finden wird alles auf den ersten Knoten verteilt. Mit `USE_ALL` könnte man im Gegensatz dazu auf jeden möglichen Knoten, mit `USE_LAST` nur auf den letzten, und mit `USE x` auf einen bestimmten, dem x-ten Knoten (0 ist der erste) verteilen. `USE_NONE` würde gar keine Verteilung vornehmen und abbrechen.

7 Zusammenfassung und Ausblick

Bei der Umsetzung von OLAP-Anwendungen auf Basis von MDX existieren Probleme insbesondere im Bereich des Splashing. Mit relativ einfachen praktischen Erweiterungen und Anpassungen von MDX kann ein Plus an Anwendungslogik beschrieben und umgesetzt werden. In weiteren Arbeiten sollen nun die Erweiterungen implementiert und getestet werden.

Literatur

[Microsoft, 1997] Microsoft (1997). OLE DB for OLAP Overview. http://msdn.microsoft.com/de-de/library/ms714903.aspx.

[Microsoft, 1999] Microsoft (1999). Multidimensional Expression (MDX) - Referenz. http://msdn.microsoft.com/de-de/library/ms145506.aspx.

[Pendse, 1998] Pendse, N. (1998). What is OLAP? http://www.olapreport.com/fasmi.htm.

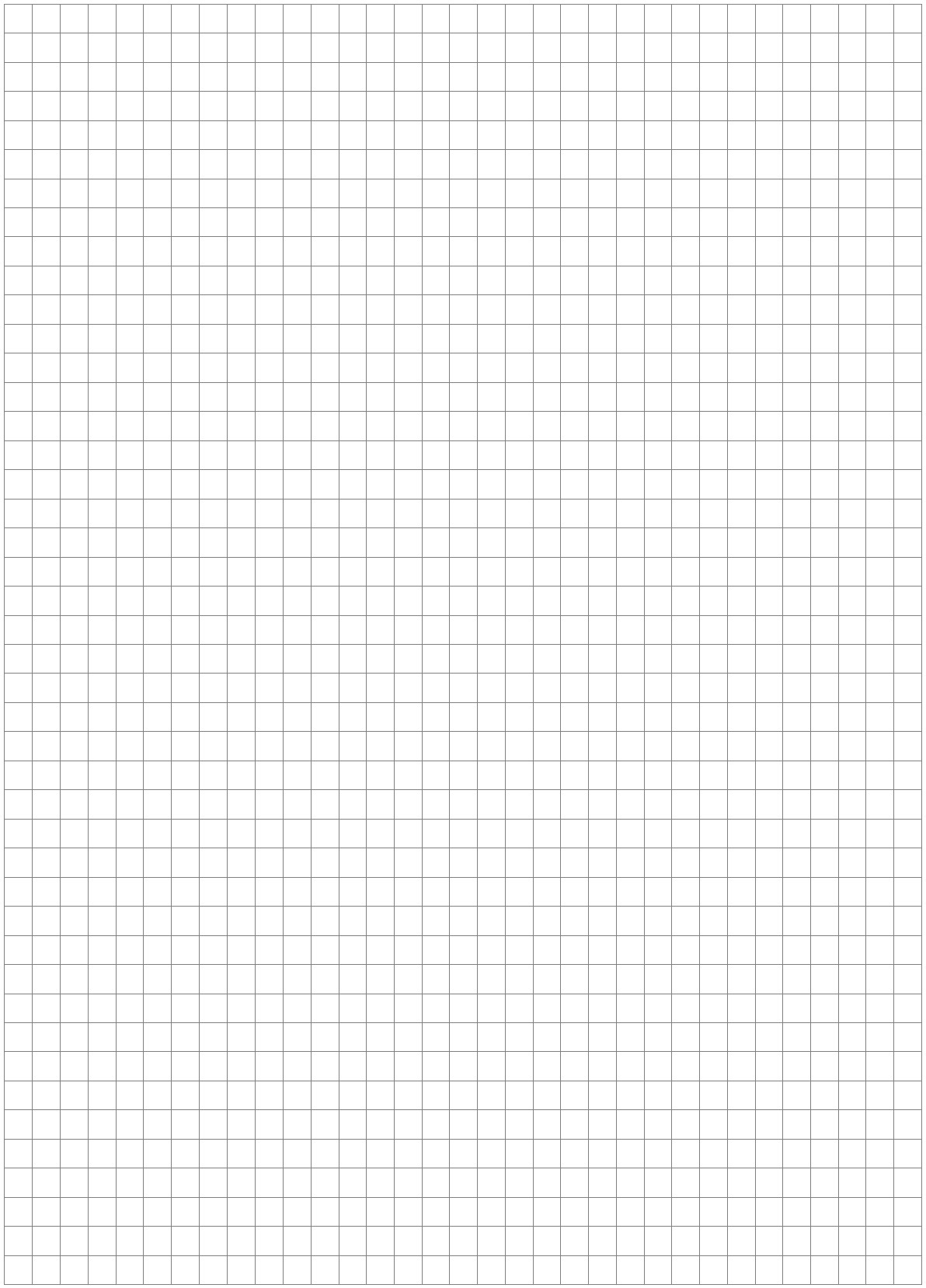
[Sidow, 2007] Sidow, H. D. (2007). *Key Account Management: Geschäftsausweitung durch kundenbezogene Strategien.* mi-Fachverlag, 8. edition.

[Simba-Technologies-Inc, 2006] Simba-Technologies-Inc (2006). XMLA, XML for Analysis, ODBO, OLE DB for OLAP, XMLA provider, bridge, server. http://www.xmlforanalysis.com.

[Tjoa et al., 2007] Tjoa, A. M., Wagner, R., Tomsich, P., and Rauber, A. (2007). *OLAP of the Future.* Österreichische Computer Gesellschaft, Wien.

[Vaisman et al., 2002] Vaisman, A. A., Mendelzon, A. O., Ruaro, W., and Cymerman, S. G. (2002). Supporting Dimension Updates in an OLAP Server.

[Willand, 2004] Willand, S. (2004). Einführung in MDX, Seminararbeit WS04/05.



Automatische Parameterbestimmung durch Gravitation in Subspace Clustering

Jiwu Zhao
Institut für Informatik
Datenbanken und Informationssysteme
Heinrich-Heine-Universität Düsseldorf
D-40225 Düsseldorf, Germany
zhao@cs.uni-duesseldorf.de

Zusammenfassung

Im Vergleich zu den traditionellen Clusteringverfahren ermöglicht Subspace Clustering die Suche nach Clustern in den Unterräumen (Subspaces) der Daten. Man unterscheidet zwei Hauptarten des Subspace-Clustering-Verfahrens: Top-Down- und Bottom-Up-Verfahren. Die Algorithmen des Top-Down-Verfahrens verkleinern die Suchbereiche von hohen zu niedrigen Dimensionen. In dem Bottom-Up-Verfahren suchen die Algorithmen nach Clustern in einer umgekehrten Reihenfolge.

Die Bestimmung der Parameter in den meisten Subspace-Clustering-Verfahren ist nicht einfach, was die Anwendung der Verfahren erschwert. Daher wird in dieser Arbeit ein Verfahren zur automatischen Parameterbestimmung diskutiert.

Keywords: *Subspace Clustering, Parameterreduzierung, Gravitation, Bottom-Up-Verfahren*

1 Grundlagen

Die traditionellen Clustering-Verfahren wie z.B. K-means, DBscan usw. suchen nach Clustern normalerweise im gesamten hochdimensionalen Raum. Subspace Clustering stellt eine Erweiterung des traditionellen Clustering dar, das die Suche nach Clustern in den Unterräumen ermöglicht. Es bietet mehr relevante Informationen, da nur die Kombination von manchen Datenattributen sinnvoll ist. Die *Abbildung 1* zeigt ein Beispiel, in dem drei Cluster auf jeweils zwei, aber nicht drei Dimensionen liegen.

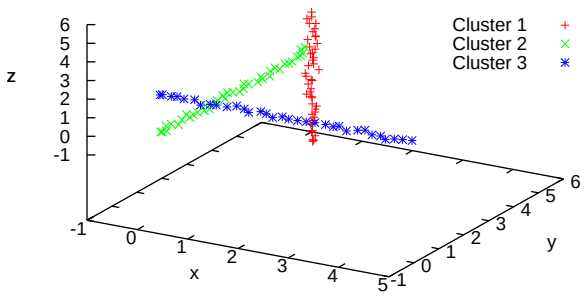


Abbildung 1: Subspace Clustering

Es gibt zwei Hauptarten des Subspace-Clustering-Verfahrens: Top-Down- und Bottom-Up-Verfahren[1]. Zu den Top-Down-Verfahren zählen PROCLUS[2], ORCLUS[3], FINDIT[4], σ-Clusters[5], COSA[6] usw. Die bekannten Beispiele für Bottom-Up-Verfahren sind CLIQUE [7], ENCLUS [8], MAFIA [9], CBF [10], CLTree [11], DOC [12] usw.

Das Problem der meisten Subspace-Clustering-Verfahren ist die Parameterbestimmung. Die *Tabelle 1* veranschaulicht die Parameter der oben genannten Verfahren.

Verfahren	**Kurze Beschreibung**	**Parameter**
PROCLUS	K-Medoid	K
ORCLUS	Abstand zweier Punkte im Unterraum	1. Clusteranzahl 2. Größe des Unterraumes
FINDIT	Medoids mit eigener Abstandfunktion	1. Minimale Punktanzahl im Cluster 2. Minimaler Abstand zwischen den Clustern
σ-Clusters	Coherence (Kohärenz)	1. Clusteranzahl 2. Clustergröße
COSA	Gewichtung der Punkte im Unterraum, K-nächste Nachbarn	1. Grenzwert des Gewichts 2. K
CLIQUE	Gruppierung durch Raster	1. Raster-Größe 2. Grenzwert der Dichte
ENCLUS	Entropie	Größe des Rasterintervalls
MAFIA	Histogramm	1. Grenzwert der Dichte 2. Grenzwert für Merging
CBF	Cell-basiert	1. Grenzwert der Sektion 2. Minimale Dichte
CLTREE	Decision Tree	1. Minimale Anzahl in einem Bereich 2. Dichte für Merging
DOC	Dichtebasiert	1. Maximale Länge 2. Minimale Punktanzahl in einem Cluster

Tabelle 1: Vergleich von Parametern in Subspace-Clustering-Verfahren

Die Tabelle zeigt, dass die meisten Verfahren Parameter benötigen müssen, die sehr schwer zu bestimmen sind. Daher wird in der vorliegenden Arbeit ein Verfahren mit automatischer Parameterbestimmung diskutiert.

2 Verfahren

2.1 Definition von Subspace Clustern

Eine Datenbank enthält Objekte und Attribute. Die letzteren können als Dimensionen betrachtet werden, wobei die Objekte die Punkte in diesen Dimensionen sind. Eine Datenbank kann man deswegen als einen Raum $\mathcal{D}$ ansehen, der eine Kombination von einem Attributraum $\mathcal{A}$ und einem Objektraum $\mathcal{O}$ ist:

$$\mathcal{D} = (\mathcal{A}, \mathcal{O})$$

Der Attributraum $\mathcal{A}$ ist definiert als die Menge aller Attribute $\mathcal{A} = \{a_1, a_2, \cdots, a_n\}$, und der Objektraum $\mathcal{O}$ ist die Menge aller Objekte $\mathcal{O} = \{o_1, o_2, \cdots, o_p\}$. Ein Subspace Cluster $\mathcal{S}$ ist ein Unterraum von $\mathcal{D}$:

$$\mathcal{S} = \widetilde{\mathcal{D}} = (\widetilde{\mathcal{A}}, \widetilde{\mathcal{O}})$$

wo $\widetilde{\mathcal{A}} \subset \mathcal{A}$, $\widetilde{\mathcal{O}} \subset \mathcal{O}$ und $\mathcal{S}$ eine in jedem Verfahren unterschiedlich definierte Bedienung $\mathcal{C}$ erfüllt.

Die Anzahl der Objekte in $\mathcal{S}$ wird mit $|\mathcal{S}|$ bezeichnet und $|\mathcal{S}| = |\widetilde{\mathcal{O}}|$. Die Schnittemenge von zwei Subspace Cluster wird wie folgt definiert:

$$\mathcal{S}_1 \cap \mathcal{S}_2 = (\widetilde{\mathcal{A}}_1 \cup \widetilde{\mathcal{A}}_2, \widetilde{\mathcal{O}}_1 \cap \widetilde{\mathcal{O}}_2)$$

2.2 Eindimensionales Gravitationsverfahren (Schritt 1)

Für hochdimensionale Daten ist es schwer, die Anzahl der Cluster vorher einzuschätzen, deswegen ist die Durchführung eines K-medoid-basierten Verfahrens problematisch. Aus diesem Grund basiert der Algorithmus auf Bottom-Up-Verfahren des Subspace Clustering. Der erste Schritt des Gravitationverfahrens wird in eindimensionalen Räumen durchgeführt.

Nehmen wir an, dass jedes Objekt ein wahres Objekt mit Gewicht ist, der Datenbankraum $\mathcal{D}$ geographische Eingenschaften hat und die Objekte in einer Dimension Gravitation (Anziehungskraft) erzeugen können.

Die Gravitationsfunktion ist definiert als $G = \mathcal{G} \cdot \frac{m_1 m_2}{r^2}$. Für die Vereinfachung nehmen wir die Gravitationskonstante heraus, dann erfolgt die Bestimmung der vereinfachten Gravitation zwischen zwei Objekten o_a, o_b folgendermaßen:

$$G_{ab} = \frac{m_a m_b}{r_{ab}^2}$$

Es sei hervorgehoben, dass das Gewicht von einem Objekt $1/N$ ist, wobei N die Anzahl der Objekte ist und der Abstand zwischen o_a, o_b als $r_{ab} = \frac{l_{ab}}{L/(N-1)}$ definiert ist, der eine Proportion des wahren zu dem durchschnittlichen Abstand $L/(N-1)$ bezeichnet, wo L die Länge einer Dimension ist. Dann kann die Funktion des Gravitationsverfahrens für eine Dimension so aussehen:

$$G_{ab} = \frac{1/N^2}{\left(\frac{l_{ab}}{L/(N-1)}\right)^2} = \frac{L^2}{l_{ab}^2 N^2 (N-1)^2}$$

Falls $l_{ab} = 0$ ist, dann werden a und b ein neues Objekt mit Gewicht $m_a + m_b$ bilden.

Die Gravitation eines Objektes und die globale Gravitation in einer Dimension werden durch folgende Maße berechnet:

- Die Gravitation eines Objektes o in der Dimension A: Summe der Gravitation mit allen Objekten, $G_o = \sum_{\forall p \in A} G_{op}$
- Die globale Gravitation einer Dimension A ist der Durchschnittswert der Gravitationen aller Objekte, $\overline{G} = \frac{1}{N} \sum_{\forall o \in A} |G_o|$

Die Gravitation in einem eindimensionalen Raum zeichnet sich durch folgende Eigenschaften aus:

- Ein Objekt in der Mitte hat einen größeren Wert als eines am Rand
- In einem kleinen Bereich liegen die Objekte mit einem großen Gravitationswert näher zu den Nachbarobjekten als die mit einem kleineren Gravitationswert.

Im Vergleich zu $\overline{G}$ kann das Objekt o mit einer sehr kleinen G_o schon als ein Outlier entfernt werden. Es sei betont, dass ein Objekt mit kleiner Gravitation in der Lage ist, wie ein Separator die Clusterobjekte zu trennen. Daher bekommen wir die Clusterinformationen in einem eindimensionalen Unterraum.

In der *Abbildung 2* kann man deutlich sehen, dass die Clusterobjekte im Vergleich zu den Nicht-Clusterobjekten größere Werte der Gravitation haben.

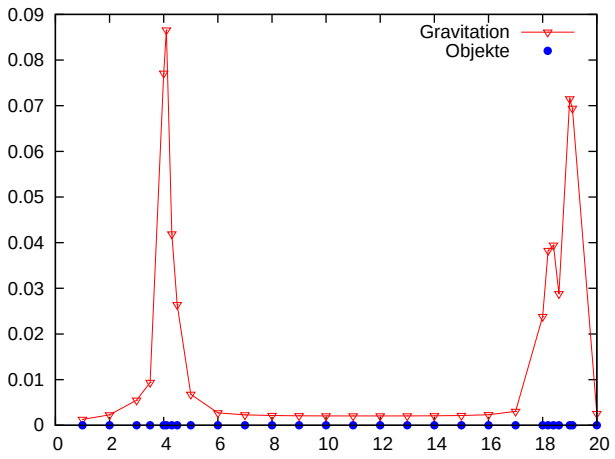


Abbildung 2: Gravitation der Objekte im eindimensionalen Raum

2.3 Suche nach hochdimensionalen Subspace Clustern (Schritt 2)

Nachdem die Cluster in jeder Dimension gefunden worden sind, müssen sie miteinander in hohen Dimensionen kombinieren werden, damit neue Subspace Cluster gebildet werden können. Die Kombination wird durch das Prinzip der Schnittmenge durchgeführt. Zum Beispiel wird für Subspace Cluster $\mathcal{S}_1$ und $\mathcal{S}_2$ die Schnittmenge $\mathcal{S}_1 \cap \mathcal{S}_2$ überprüft. Falls $\mathcal{S}_1 \cap \mathcal{S}_2$ die Bedienung $\mathcal{C}$ erfüllt, ist die neue Menge auch ein Subspace Cluster. Dieser Prozess wird durchgeführt, bis kein neuer Subspace Cluster gefunden wird. Dann ist das Subspace Clustering abgeschlossen.

Das Überprüfen ist einfach: Die Anzahl der Objekte in $\mathcal{S}_1 \cap \mathcal{S}_2$ wird berechnet, und falls $|\mathcal{S}_1 \cap \mathcal{S}_2| > M$ ist, bildet $\mathcal{S}_1 \cap \mathcal{S}_2$ einen neuen Subspace Cluster. M ist die minimale Anzahl von Objekten in einem Cluster. Da $|\mathcal{S}_i \cap \mathcal{S}_j| \leq |\mathcal{S}_i|$ ist, haben die neugebildeten Cluster nicht mehr Objekte als die älteren, deswegen soll M nicht groß sein. Wenn M auf 0 gesetzt wird, dann ist es möglich ganz kleine Cluster zu finden, die später entfernt werden können. M ist trotzdem ein Parameter, den man durch eine grobe Bestimmung im Verfahren verwenden kann.

3 Schlussfolgerung

In dieser Arbeit wurde die Idee für Subspace Clustering mit einer Gravitationsfunktion vorgestellt, um das Problem der schweren Parameterbestimmung zu beseitigen.

Die vereinfachte Gravitationsfunktion wird auf eindimensionale Objekte verwendet. Dank den guten Eigenschaften der Gravitationsfunktion ist es möglich Cluster und Outlier zu trennen. Hochdimensionale Subspace Cluster können durch die Kombination von Objekten in niedrigen Unterräumen gebildet werden.

Eine spätere Arbeit wird sich der Beziehung von Objekten durch die Gravitation in mehreren Dimensionen widmen. Es wird versucht, die vereinfachte Gravitationsfunktion noch zu verfeinern, damit sie an hohe Dimensionen angepasst werden kann und die Trennung der Cluster- von Nicht-Clusterobjekten besser erfolgt.

Literatur

[1] Lance Parsons, Ehtesham Haque, and Huan Liu. Subspace clustering for high dimensional data: A review. *Sigkdd Explorations*, 6:90–105, June 2004.

[2] Charu C. Aggarwal, Joel L. Wolf, Philip S. Yu, Cecilia Procopiuc, and Jong Soo Park. Fast algorithms for projected clustering. In *Proceedings of the 1999 ACM SIGMOD international*

conference on Management of data, pages 61–72, Philadelphia, Pennsylvania, United States, May 31-June 03 1999.

[3] Charu C. Aggarwal and Philip S. Yu. Finding generalized projected clusters in high dimensional spaces. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 70–81, Dallas, Texas, United States, May 15-18 2000.

[4] K.-G. Woo and J.-H. Lee. *FINDIT: a Fast and Intelligent Subspace Clustering Algorithm using Dimension Voting*. PhD thesis, Korea Advanced Institute of Science and Technology, Taejon, Korea, 2002.

[5] J. Yang, W. Wang, H. Wang, and P. Yu. δ-clusters: Capturing subspace correlation in a large data set. In *Proceedings of the 18th International Conference on Data Engineering*, page 517, February 26-March 01 2002.

[6] J. H. Friedman and J. J. Meulman. Clustering objects on subsets of attributes. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 2002.

[7] Rakesh Agrawal, Johannes Gehrke, Dimitrios Gunopulos, and Prabhakar Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *Proceedings of the 1998 ACM SIGMOD international conference on Management of data*, pages 94–105, Seattle, Washington, United States, June 01-04 1998.

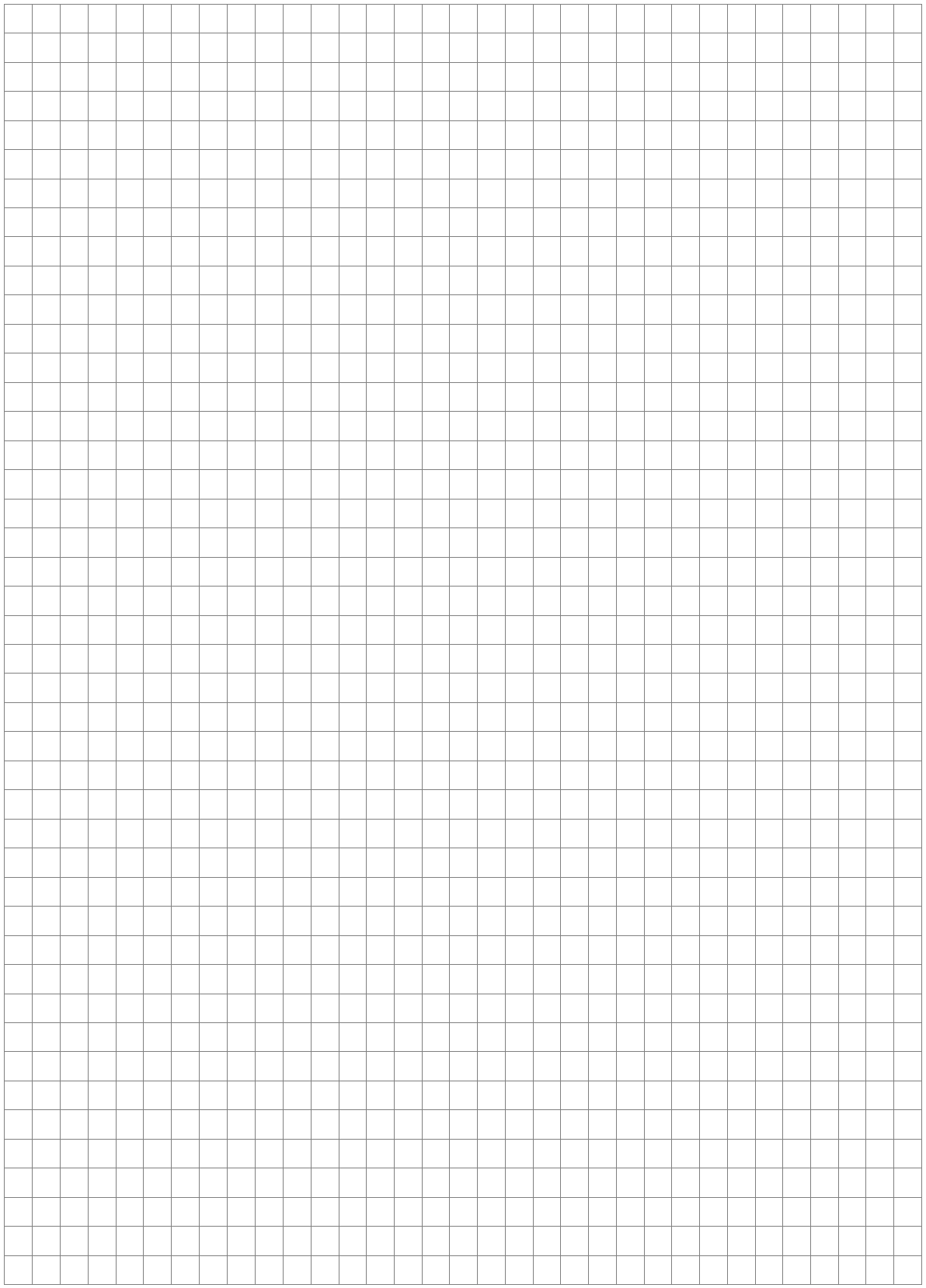
[8] Chun-Hung Cheng, Ada Waichee Fu, and Yi Zhang. Entropy-based subspace clustering for mining numerical data. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 84–93, San Diego, California, United States, August 15-18 1999.

[9] S. Goil, H. Nagesh, and A. Choudhary. Mafia: Efficient and scalable subspace clustering for very large data sets. Technical report cpdc-tr-9906-010, Northwestern University, June 1999.

[10] Du-Seok Jin Jae-Woo Chang. A new cell-based clustering method for large, high-dimensional data in data mining applications. In *Proceedings of the 2002 ACM symposium on Applied computing*, pages 11–14, Madrid, Spain, March 2002.

[11] Bing Liu, Yiyuan Xia, and Philip S. Yu. Clustering through decision tree construction. In *Proceedings of the ninth international conference on Information and knowledge management*, pages 20–29, McLean, Virginia, United States, November 06-11 2000.

[12] Cecilia M. Procopiuc, Michael Jones, Pankaj K. Agarwal, and T. M. Murali. A monte carlo algorithm for fast projective clustering. In *Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, Madison, Wisconsin, June 03-06 2002.



Replikation im mobilen Pflegeassistenzsystem MARIKA

Martin Düffer, Andre Peters

Universität Rostock
Institut für Informatik
{martin.dueffer, andre.peters} @uni-rostock.de

Zusammenfassung Mit dem derzeitigen demografischen Wandel nimmt auch die Bedeutung der häuslichen Gesundheitspflege stark zu. Professionelle Pflegedienste unterliegen dabei einer gesetzlichen Dokumentationspflicht. Die Dokumentation von Pflegeaktivitäten ist eine sehr zeitaufwendige Tätigkeit und nimmt bis zu 40% der täglichen Arbeitszeit in Anspruch. Durch automatische Dokumentation ist es möglich das Pflegepersonal zu unterstützen, so dass letztendlich mehr Zeit für die eigentlichen Pflegetätigkeiten bleibt.
Das in dieser Arbeit vorgestellte Projekt MARIKA beschäftigt sich mit der Entwicklung einer globalen Pflegedokumentations- und Koordinationsplattform. Diese soll das Pflegepersonal durch automatische Aufzeichnung der erbrachten Pflegetätigkeiten unterstützen. Durch Rückschlüsse aus der Analyse und Auswertung dokumentierter Tätigkeiten kann die Planung täglicher Arbeitsabläufe verbessert und die Auslastung vorhandener Ressourcen optimiert werden.
Ein Kernbestandteil des Systems bilden Geräte für den ambulanten mobilen Pflegeeinsatz. Nach Aufzeichnung der erbrachten Pflegetätigkeiten werden die aufgezeichneten Daten zwischen den Komponenten des Systems repliziert. Für den Einsatz in verteilten Datenbanksystemen mit mobilen Teilnehmern werden spezielle Replikationsverfahren vorgestellt.

1 Einleitung

Der „Landesforschungsschwerpunkt Mobile Assistenzsysteme“[1] beschäftigt sich mit der Entwicklung neuer IT-Konzepte und Produkte u.a. in den Bereichen Gesundheit und intelligente Umgebungen unter dem Leitthema der mobilen Assistenz.

Höhere durchschnittliche Lebenserwartungen durch medizinische Fortschritte und rückläufige Reproduktionsraten [2] lassen die Menschheit zunehmend altern. Ein Phänomen, das sich auch im internationalen Vergleich widerspiegelt. Körperliche und geistige Funktionsverlagerungen mit zunehmendem Alter gipfeln nicht selten in späteren Aktivitätseinschränkungen. Gewohnte Tätigkeiten fallen mit steigendem Alter schwerer oder können nicht mehr ausgeübt werden. Ältere Menschen werden dann nicht selten zu Hilfe- oder Pflegebedürftigen. Die Bevölkerungsschicht der älteren Menschen ist die zur Zeit am stärksten wachsende.

Wissenschaft und Industrie stehen vor großen Herausforderungen neue Lösungsansätze für den Umgang mit dem demografischen Wandel zu entwickeln. Häusliche Pflege älterer und kranker Menschen stellt nicht selten eine beträchtliche Belastung für die Angehörigen dar. Bis zu einem Drittel der Aufnahmen in Pflegeheimen sind dem sogenannten „Pflege Burnout“geschuldet.

Ziel neuer Technologien ist es, Angehörige zu entlasten und betroffenen Personen einen hohen Grad an Selbstständigkeit zu erhalten, damit sie so lange als möglich in ihrer gewohnten Umgebung und in ihrem gewohnten sozialen Umfeld verbleiben können. Ist eine Pflege durch Angehörige nicht möglich, besteht in vielen Fällen, alternativ zur stationären Pflege, die Möglichkeit professionelle ambulante Pflege in Anspruch zu nehmen. Ein professioneller Pflegedienstleister unterliegt einer gesetzlichen und sehr genau geregelten Dokumentationspflicht bei der Durchführung von Pflegeaktivitäten.

Der Dokumentationsprozess nimmt dabei nach Angaben des projektbegleitenden Pflegedienstes bis zu 40% der Arbeitszeit in Anspruch. Dokumentationen werden in der Regel handschriftlich erstellt und sind nicht selten ungenau, da ein straffer Pflege- bzw. Dienstplan die Anfertigung einiger Teile der Dokumentation erst zu einem späteren Zeitpunkt zulässt.

Neuheitliche Lösungen sind gefordert, die einen begleitenden, fehlerfreien, korrekten und weitestgehend automatisierten Dokumentationsprozess gewährleisten. Eine durchgehende digitale Verarbeitung kann ebenfalls helfen, Informationsbrüche zu vermeiden und durch geeignete Schnittstellen eine hohe Interoperabilität sicherzustellen. Das so unterstützte Personal bekommt damit mehr Zeit für eigentliche Pflegetätigkeiten und wird bei der täglichen Arbeit zudem durch geeignete Funktionalitäten, wie Erinnerungsfunktionen oder Entscheidungshilfen unterstützt.

Nach einer kurzen Vorstellung des MARIKA Projekts in Abschnitt 2 werden im Abschnitt 3 die Problemfelder skizziert und in Abschnitt 4 erste Überlegungen zur deren Lösung aufgezeigt. Am Schluss gibt Abschnitt 5 eine Zusammenfassung und einen Ausblick auf weitere Forschungsvorhaben.

2 Das MARIKA Projekt

Die gesundheitliche Forschung ist ein enormer Wachstumsmarkt und besitzt sehr viel Potential für Innovationen. Eine große Anzahl Veröffentlichungen bestätigt diesen Trend. Wichtige Schlüsselfunktionen nehmen Koordinations- und Kommunikationsplattformen in der ambulanten Gesundheitspflege ein [3,4]. Diese gestatten den am Pflegeprozess beteiligten Parteien eine verbesserte Organisation und Koordination ihrer Arbeitsabläufe und fördern die korrekte Verbreitung von Informationen untereinander.

Ziel des Projekts MARIKA [1] (**M**obile **A**ssistenz für **R**outen**I**nformation und **K**ranken**A**kte) ist, verschiedenste Technologien für eine weitestgehend automatische Dokumentation des Pflegeprozesses bereitzustellen. Entwicklungen im Projekt erstrecken sich auf folgende Bereiche:

- Sensorbasierte Aktivitätserkennung mittels statistischer Modelle
- Sprachverarbeitung
- Datensicherheit und Datenschutz
- Indoor/Outdoor Lokalisierung (Routeninformation)
- E-Learning Aspekte
- Mobile Datenverwaltung
- Daten- und Funktionsintegration
- Kontent- und Wissensmanagement

Ein Kernbereich des Projekts ist der Informationsaustausch. Dieser beschäftigt sich zum Einen mit der Integration verschiedenster heterogener Datenquellen, zum Anderen mit dem Datenaustausch zwischen verschiedenen Komponenten bzw. Akteuren des Gesamtsystems. Im Abschnitt 4 wird näher auf verwendete Techniken aus dem Bereich der mobilen Replikation eingegangen.

Die von uns fokussierte Pflegeplattform hat zum Ziel, die Kosten im Pflegeprozess zu senken und Dienstleister effizient bei der Koordination und Planung alltäglicher Arbeitsabläufe zu unterstützen. Hohe Interoperabilität ist eine gewichtige Forderung des MARIKA Systems.

Vorerst wird sich der Datenaustausch primär auf die unmittelbaren Komponenten des Pflegesystems beschränken. In weiteren Entwicklungen sind zusätzliche Verflechtungen mit Systemen externer Einrichtungen, wie Krankenhäusern und Arztpraxen denkbar.

Durch die enge Zusammenarbeit mit einem ortsansässigen ambulanten Pflegedienst kann auf einschlägige Erfahrungen aus alltäglichen Arbeitsabläufen zurückgegriffen werden. Das erleichtert die Entwicklung geeigneter Modelle und Komponenten. Ebenso erleichtert es das Testen dieser unter realen Bedingungen in häuslichen Umgebungen mit tatsächlich Pflegebedürftigen und entsprechend ausgebildetem Pflegepersonal.

3 Problemstellung

Der Dokumentationsprozess im medizinischen Sektor, insbesondere im Umfeld ambulanter Pflege, erfolgt meist handschriftlich und weist oftmals an mehreren Stellen Informationsbrüche auf. Das Gesamtkonzept MARIKA muss die resultierenden Probleme lösen. Der Kernpunkt dieser Arbeit liegt dabei im Bereich der mobilen Datenverwaltung und der damit verbundenen Replikation. Im Folgenden soll ein alltäglicher Tagesablauf skizzieren, wie wir uns ein ambulantes Pflegeszenario in Zukunft vorstellen und aufzeigen mit welchen Problemfeldern wir uns konfrontiert sehen.

In einem zentralen Datenmanagementsystem existieren zu jedem Pflegebedürftigen eine Pflegedokumentation (Krankenakte) und ein vertraglich festgelegter Pflegeplan. Je Pflegekraft wird unter Berücksichtigung patientenbezogener persönlicher Vorlieben, bestimmter zeitlicher Abfolgen, bestimmten festgesetzten Terminen und voraussichtlicher Dauer der beim Patienten zu verrichtenden Pflegetätigkeiten ein täglicher Einsatzplan erstellt. Dieser legt fest, welcher Patient wann und wie versorgt werden muss. Resultierende Routeninformationen werden automatisch erstellt und zusammen mit weiteren erforderlichen Dokumenten auf mobile Geräte repliziert.

Gewöhnlich fährt die Pflegekraft dann die ihr zugeteilten Pflegebedürftigen mit dem Auto ab und verrichtet die im Pflegeplan enthaltenen Tätigkeiten beim Patienten. Die Dokumentation der Tätigkeiten erfolgt automatisch, z.B. durch sensorbasierte Aktivitätserkennung oder Sprachsteuerung. Ist der Pflege- und Dokumentationsprozess beendet, werden die entsprechenden Daten sowohl auf das zentrale Datenmanagementsystem, wie auch auf einen separaten Datenspeicher beim Patienten übertragen. Im zentralen Datenmanagementsystem stehen diese zeitnah für Berechnungen, Analysen und Planungsaktivitäten zur Verfügung. Der separate Datenspeicher erlaubt es, Angehörigen und Einsatz- und Rettungskräften ohne Zugang zum zentralen System Einblick in den derzeitigen Stand der Pflegeaktivitäten zu gewähren. Treten spontane Änderungen im Tagesablauf einer oder mehrerer Pflegekräfte auf, werden die benötigten Daten automatisch auf den mobilen Geräten angepasst bzw. benötigte Daten auf diese übertragen.

Als Geräte für den mobilen Einsatz bei Pflegediensten sind TabletPCs, PDAs und Smartphones denkbar. Im Projekt selbst werden Geräte präferiert, die auf Intels Referenzarchitektur für *Mobile Clinical Assistant* (MCA)[5] basieren. Geräte dieser Architektur bringen eine Vielzahl an integrierten Features, wie Barcode Scanner, RFID Scanner, Fingerprint Sensor, Kamera und verschiedene drahtgebundene wie drahtlose Netzwerk- und Kommunikationseinrichtungen mit. Erste prototypische Anwendungen im Projekt wurden bereits auf dieser Architektur entwickelt. Ein im Einsatz befindliches elektronisches Fahrtenbuch und ein digitales Logbuch liefern ebenfalls erste verwertbare Ergebnisse über benötigte Laufzeiten, Dauer von Behandlungen und zurückgelegten Entfernungen.

Durch einen hohen Grad an Mobilität, den Umgang mit personenbezogenen Daten und der Autonomie beteiligter Komponenten und Mitglieder am Pflegesystem, sehen wir uns ebenso mit Datenschutzanforderungen, wie auch Anforderungen einfacher Handhabung, Portabilität und Adaptivität konfrontiert. Entsprechend sind Daten zu klassifizieren und ein zuverlässiges und einfach zu handhabendes Zugriffsrechtesystem zu implementieren.

Basis für Abrechnungen im Pflegeumfeld sind die sogenannten Servicekataloge, in denen alle Pflegeaktivitäten in spezielle Pflegestufen untergliedert sind. Diese Kataloge beschreiben exakt die Durchführung, das Ausmaß und die abrechenbaren Kosten für alle Pflegeaktivitäten.

Pflegedokumentation und Pflegeplan enthalten eine Vielzahl unterschiedlicher Dokument- und Datentypen, wie Sensordaten, Daten zu repräsentativen Pflegeaktivitäten, Text und Multimediadaten, wie Fotos, Videos und Sprachnachrichten und Daten aus zeitlichen und räumlichen Loggingaktivitäten. Diese Daten und Dokumente sind Grundlage für Abrechnungen und Fragestellungen, wie: „Bei welchem Patienten wurden welche Tätigkeiten wie lange durchgeführt? Traten Besonderheiten auf?“ . Verschiedene Informationen sind zeitkritisch. Dokumente und Daten müssen klassifiziert und durch intelligente Vorauswahl entsprechend repliziert werden.

4 Replikation in MARIKA

Im Abschnitt 3 wurde ein zukünftiger alltäglicher Arbeitsablauf geschildert. Bei dem von uns fokussierten System handelt es sich um ein verteiltes Datenbanksystem mit mobilen Teilnehmern, bestehend aus einem zentralen stationären Datenbanksystem, mobilen Datenbanken auf den mobilen Geräten der Pflegekräfte, sowie eingebetteten Datenbanken auf den Geräten der Patienten, welche hauptsächlich die Bestandteile der persönlichen Pflegedokumentation verwalten.

Nach dem Zielkonflikt der Replikation [6] widersprechen sich die beiden Forderungen des Projekts MARIKA nach maximaler Verfügbarkeit des Gesamtsystems und strikter Konsistenz aller Replikate.

Für weitere Betrachtungen wird im Folgenden auf die Bereiche Kopie-Update-Strategien und Fehlerbehandlung der aus [7] eingeführten „Diskussion von Replikationsverfahren“ eingegangen. Im Gegensatz zu statischen Knoten verteilter Datenbanken, sind mobile Teilnehmer fast ausschließlich vom Netzwerk getrennt. Klassische Verfahren, wie *Read-One-Write-All* sind im mobilen Umfeld ungeeignet, da sie durch Sperrverfahren die Verfügbarkeit aller Replikate für den fehlerfreien Einsatz voraussetzen. Primär-Kopie-Verfahren hingegen führen Änderungen auf der Masterkopie durch und propagieren diese.

Bezüglich des Verhaltens bei Netzpartitionierungen untersucht [7] konservative und progressive Verfahren. Der Fokus progressiver Verfahren liegt dabei auf der Verfügbarkeit des Gesamtsystems. Im unverbundenen Zustand kann auf den mobilen Klienten uneingeschränkt weitergearbeitet werden. Gegebenenfalls auftretende temporäre Inkonsistenzen werden bei diesem Verfahren in Kauf genommen. Bei konservativen Verfahren hingegen liegt der Fokus auf der strikten Erhaltung der Konsistenz. Bei diesem Verfahren kann es ggf. zur temporären Nichtverfügbarkeit des Systems kommen.

Gemäß den Anforderungen des Projekts werden im weiteren die Primär-Kopie-Verfahren und die progressiven Verfahren als Grundlage für Replikationstechniken in mobilen Datenbanken angesehen und eingehender betrachtet.

Die Zwei-Schichten-Replikation [8] oder auch provisorische Transaktion [6] ist ein erweitertes Primär-Kopie-Verfahren [9] und bietet die Möglichkeit, auch beim Arbeiten im lokalen offline-Modus die Integrität des Gesamtsystems zu wahren. Ein Vorteil provisorischer Transaktionen ist, dass beliebig viele lokale Transaktionen ausgeführt werden können, bevor diese auf einem Datenbankserver eingebracht werden. Lange offline-Phasen erhöhen allerdings die Wahrscheinlichkeit von Einbringungskonflikten. Im verbindungslosen Betrieb entstehende Daten sind ebenso wie die durchgeführten Transaktionen selbst zunächst nur provisorisch. Sogenannte Bereichsregeln definieren den Gültigkeitsbereich dieser Transaktionen und sichern damit bestimmte Bedingungen ab.

Bei der nächsten Übertragung zum Datenbankserver wird versucht, unter Beachtung des definierten Gültigkeitsbereiches die Transaktionen auszuführen. Ergebnisse werden an den mobilen Klienten propagiert. Ist es nicht möglich die Transaktionen in Basistransaktionen zu überführen, wird eine entsprechende Fehlermeldung generiert.

Das aus dem Xerox PARC's Bayou Project [10] hervorgegangene Bayou-System verwendet ebenfalls provisorische Transaktionen. [11] beschreibt die Erweiterungen *per-write dependency checks* und *per-write merge procedures* als Verfahren zur verbesserten Konflikterkennung bzw. Konfliktlösung. [12] beschreibt ein einfaches und sehr flexibles *Anti-Entropie-Protokol* für die Update Propagation zwischen schwach konsistenten Replikaten auf Basis von paarweisem Ausgleich, welches zudem das Arbeiten in verschiedensten Netzwerkumgebungen unterstützt.

In der Pflegedokumentation kann zwischen zeitkritischen und zeitunkritischen Daten unterschieden werden. Zeitkritische Daten nehmen einen hohen Stellenwert ein, da sie bestimmte Handlungsverläufe bedingen oder direkt beeinflussen. Die vorgestellten Verfahren sind dahingehend zu untersuchen und zu erweitern, als dass sie die Forderungen nach zeitnaher Replikation und Konsistenz aller Replikate zumindest für den Teil der „wichtigen“ Daten erfüllen.

5 Zusammenfassung und Ausblick

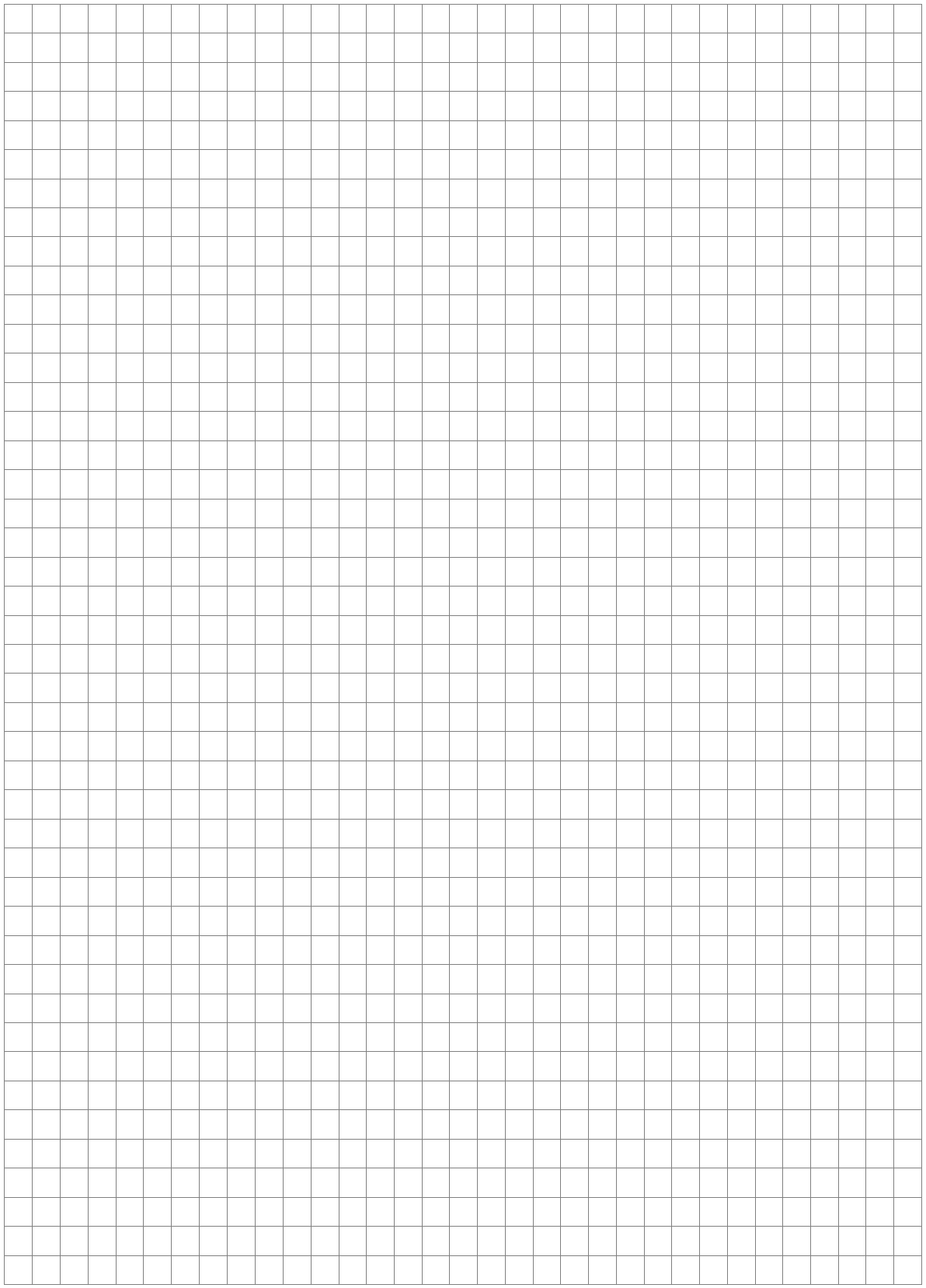
Das Projekt MARIKA befasst sich hauptsächlich mit der Unterstützung ambulanter Pflegekräfte bei der automatischen Dokumentation von Pflegetätigkeiten. Eines der Ziele ist die Bereitstellung einer Pflegedokumentations- und Koordinationsplattform. Das fokussierte System soll zusätzlich zur Dokumentation die Pflegekräfte durch geeignete Funktionalitäten in Planung und Durchführung von Pflegetätigkeiten unterstützen. Dadurch ist eine verbesserte Koordination von Arbeitsabläufen, eine höhere Ressourcenausnutzung und eine Senkung der Kosten im Pflegebereich zu erwarten.

Datenreplikation ist im Projekt MARIKA für die systemweite Bereitstellung von Informationen ein wichtiger Aspekt. Mobile Klienten arbeiten üblicherweise im unverbundenen Zustand. So sind Replikate nicht immer erreichbar. Klassische ROWA Techniken sind damit in verteilten Datenbanksystemen mit mobilen Teilnehmern nicht anwendbar. Erweiterte Primär-Kopie-Verfahren können auch in mobilen Umgebungen zur Propagation von Änderungen genutzt werden. Die Fehlerbehandlung nach Netzpartitionierungen wird durch den Einsatz progressiver Techniken vollzogen und bildet die Grundlage für spätere Synchronisationsverfahren.

Die Forderungen nach maximaler Verfügbarkeit des Gesamtsystems bei gleichzeitiger maximaler Konsistenz der Replikate stehen im Widerspruch zueinander [6]. Weiterführende Untersuchungen müssen zeigen, wie Konflikte durch geeignete Erweiterungen der vorgestellten Techniken minimiert werden können. In diesem Zusammenhang muss aus dem Kontext der Pflegedokumentation heraus eine Klassifizierung der Daten nach Kriterien geforderter Konsistenz bzw. vordergründiger Verfügbarkeit erarbeitet werden.

Literatur

1. Uni Rostock: Landesforschungsschwerpunkt. `http://marika.lfs-ma.de` (2008) [Online].
2. Krüger-Brand, H.E.: VDE-Positionspapier: Potenziale von Ambient Assisted Living. Deutsches Ärzteblatt **46** (2008)
3. Bricon-Souf, N., Anceaux, F., Bennani, N., Dufresne, E., Watbled, L.: A distributed coordination platform for home care: analysis, framework and prototype. International Journal of Medical Informatics **74** (2005) 809–825
4. Karopka, T., Bruder, I., van den Berg, N., Hoffmann, W., Heuer, A.: Ict architecture for a community medicine nurse project. In: HEALTHINF (2). (2008) 26–30
5. Intel: Products and solutions for health and healthcare, mobile clinical assistant (MCA). `http://www.intel.com/healthcare/ps/mca/` (2008) [Online].
6. Mutschler, B., Specht, G.: Mobile Datenbanksysteme : Architektur, Implementierung, Konzepte. Springer (2004)
7. Dadam, P.: Verteilte Datenbanken und Client/Server-Systeme – Grundlagen, Konzepte, Realisierungsstrukturen. Springer-Verlag, Berlin (1996)
8. Gray, J., Helland, P., O'Neil, P.E., Shasha, D.: The dangers of replication and a solution. In: SIGMOD Conference. (1996) 173–182
9. Stonebraker, M.: Concurrency control and consistency of multiple copies of data in distributed ingres. IEEE Trans. Softw. Eng. **5**(3) (1979) 188–194
10. Xerox Palo Alto Research Center: Xerox PARC's Bayou Project. `http://http://www2.parc.com/csl/projects/bayou/` (1993–1997) [Online].
11. Terry, D.B., Theimer, M.M., Petersen, K., Demers, A.J., Spreitzer, M.J., Hauser, C.H.: Managing update conflicts in bayou, a weakly connected replicated storage system. In: SOSP '95: Proceedings of the fifteenth ACM symposium on Operating systems principles, New York, NY, USA, ACM (1995) 172–182
12. Petersen, K., Spreitzer, M.J., Terry, D.B., Theimer, M.M., Demers, A.J.: Flexible update propagation for weakly consistent replication. In: SOSP '97: Proceedings of the sixteenth ACM symposium on Operating systems principles, New York, NY, USA, ACM (1997) 288–301



Datenunvollständigkeit aufgrund der mangelnden Modellierungsmächtigkeit aktuell dominierender Datenmodelle

Fabian Panse

31. März 2009

Zusammenfassung

In den am weitest verbreiteten Datenmodellen, speziell dem relationalen Datenmodell, werden Informationen über die Ausprägungen einzelner Objekteigenschaften in Attributen gespeichert. In vielen Fällen (z.B. bei partiellen Informationen) ist eine Darstellung durch einzelne Elemente des dem Attribut zugehörigen Wertebereichs jedoch nicht möglich und erfordert die Anwendung spezieller Konzepte (z.B. Nullwerte). In aktuell verwendeten Modellen sind diese Konzepte jedoch nur unzureichend auf die notwendigen Erfordernisse ausgelegt. Die ursprünglich vorliegenden Informationen lassen sich daher oft nicht wieder aus den gespeicherten Daten zurück gewinnen. Bisherige Ansätze zur Behebung dieses Problems haben sich aus unterschiedlichen Gründen nicht durchsetzen können. Die hier beschriebene Arbeit enthält daher einen entsprechenden Vorschlag, der sowohl den Informationsverlust während der Datenspeicherung verringern als auch die Schwächen der bisherigen Lösungsansätze hinsichtlich eines fehlenden Durchsetzungsvermögen vermeiden soll. Ersteres wird durch eine Verwendung mehreren Nullwerten ermöglicht, letzteres beruht hauptsächlich auf der Vermeidung gravierender Abweichungen von den aktuell vorherrschenden Modellen. Da dies wiederum eine Beibehaltung wichtiger und fundamentaler Konzepte erfordert, muss die Auswertung der verschiedenen Nullwerte in der drei-wertigen Logik erfolgen. Neben einer Kompatibilität zu den vorherrschenden Modellen, bietet dieses Vorgehen zudem die Vorteile einer geringen Modellkomplexität und ermöglicht eine intuitive Handhabung der auf dem neu entworfenen Modell basierenden Systeme.

1 Einleitung

Die Qualität der in Datenbanken gespeicherten operationalen Daten ist in den letzten Jahren zu einem wichtigen und viel beachteten Faktor geworden. Dies betrifft unter anderem auch das Qualitätskriterium der Datenvollständigkeit. Ausgehend von der Annahme, dass eine dauerhafte und korrekte Speicherung der Daten durch das verwendete Datenbanksystem gewährleistet werden kann, lässt sich unvollständige Datenbasis auf die Minderwertigkeit zweier wesentlicher Vorgänge, der Informationserhebung und der Datenspeicherung, zurückführen. Während die Folgen einer unzureichenden Informationserhebung für die Vollständigkeit einer Datenbasis offensichtlich sind, spielt die Datenspeicherung eine weitere wichtige, heutzutage aber leider weitgehend vernachlässigte Rolle. Die in den heutigen dominierenden Datenbanksystemen vorhandenen Konzepte reichen nicht aus um verschiedenartige Informationen immer durch verschiedenartige Datenwerte oder Datensätze darstellen zu können. Der daraus entstehende Informationsverlust ist in einer minderwertigen Darstellungsfähigkeit des verwendeten Datenbankschemas begründet. Diese lässt sich wiederum auf eine mangelnde Modellierungsmächtigkeit des dem Schemaentwurf zugrundeliegenden logischen Datenmodells zurückzuführen. Bisherige Lösungsansätze konnten zwar den bei der Datenspeicherung auftretenden Informationsverlust verringern, aufgrund der fehlenden Akzeptanz seitens der Datenbank-Gemeinde (Anbieter wie auch Nutzer), konnten sie sich bis dato aber nicht bewähren. Dies zeigt, dass neben der Behebung des Informationsverlustes noch weitere, nicht datenbankspezifische Aspekte berücksichtigt werden müssen. Der hier vorgestellte

Lösungsansatz basiert daher auf dem Entwurf eines Modells, dessen Modellierungsmächtigkeit zum Einen ausreicht, um eine möglichst verlustfreie Datenspeicherung zu ermöglichen, zum Anderen aber auch mögliche Akzeptanzprobleme sowohl von den Systemanbietern als auch den -nutzern umgehen kann.

2 Problemstellung

2.1 Datenvollständigkeit

Die Datenvollständigkeit ist ein Qualitätskriterium, welches den Vollständigkeitsgrad des durch die Daten gespeicherten Wissen im Verhältnis zu dem maximal vorhandenen Wissens der betroffenen Anwendungswelt wiedergibt. Die Datenvollständigkeit kann dabei in zwei Dimensionen unterteilt werden ([8]). Die Abdeckung der Daten gibt an, wieviele der vorhandenen Objekte und Beziehungen der zu modellierenden Anwendungswelt tatsächlich in der Datenbasis durch einen Datensatz gespeichert sind. Die Dichte der Daten spezifiziert, wieviele Eigenschaften der gespeicherten Objekte und Beziehungen durch geeignete Datenwerte in den zugehörigen Attributen informationserhaltend repräsentiert werden. Bezüglich der Daten-Dichte kann eine Unvollständigkeit in genau zwei Fällen auftreten. Entweder liegen bezüglich der zu speichernden Eigenschaft keine vollständigen Informationen vor, und/oder die vorliegenden Informationen lassen sich nicht verlustlos durch Daten speichern.

2.2 Informationsverlust während der Datenspeicherung

Informationen werden während der Datenspeicherung durch schemakonforme Datensätze dargestellt. Handelt es sich dabei um ein Datenbankschema, bei welchem zur Speicherung von Objekteigenschaften Attribute verwendet werden, so stehen zur Darstellung der Ausprägung einer Objekteigenschaft lediglich die Elemente des dem zugehörigen Attribut zugewiesenen Wertebereichs zur Verfügung. In vielen Datenbanksystemen soll zudem noch die Atomarität der Attributwerte gelten (z.B. aufgrund der 1. Normalform). Dies bedeutet, dass sich jede der zu speichernden Informationen durch einen einzelnen Wert aus dem zugehörigen Wertebereich darstellen lassen muss. Ist dies nicht möglich, muss auf andere Darstellungskonzepte zurückgegriffen werden. Das am meisten gebräuchlichste dieser Konzepte ist die Verwendung von Nullwerten. Enthält die Menge der Ausprägungsinformationen jedoch verschiedenartige Informationen, die sich nur durch einen gleichen Nullwert darstellen lassen, bildet die Datenspeicherung Informationen mit durchaus unterschiedlichen Informationsgehalten auf einen gleichen Datenwert ab und ist folglich nicht injektiv. Da eine nicht injektive Abbildung keine inverse Abbildung besitzt, können die ursprünglich vorhandenen, aus der zu modellierenden Welt erhobenen Informationen nicht wieder aus den gespeicherten Datensätzen hergeleitet werden. Ein irreperabler Informationsverlust während der Datenspeicherung ist demnach die Folge.

Welche Mittel zur Datenspeicherung zur Verfügung stehen, hängt also von dem verwendeten Datenbankschema und somit u.a. von der Modellierungsmächtigkeit des beim Schemaentwurf verwendeten Datenmodells ab. Die am weitest verbreiteten und von allen großen relationalen Datenbanksystemen verwendeten Variationen des relationalen Datenmodell besitzten lediglich den einzigen Nullwert *null*. Als ein diese Variationen stellvertretendes Datenmodell wird im Folgenden das als DM_L^{SQL} bezeichnete Modell des aktuellen SQL-Standards betrachtet. Eine Verwendung dieses Modells bedeutet, dass zur Darstellung aller verschiedenartiger, nicht durch einen atomaren Wert darstellbarer Informationen nur ein einziger Nullwert vorhanden ist. Da dieser Nullwert auch für den Fall, dass keinerlei Informationen über die betreffende Objekteigenschaft vorliegen verwendet wird und während der Informationsgewinnung keine falschen Annahmen getroffen werden dürfen, besitzt der Nullwert *null* eine *no-information*-Interpretation. Wird also eine Information auf *null* abgebildet, so suggeriert der aus der Speicherung dieses Objektes

resultierende Datensatz, unabhängig von dem tatsächlich darzustellenden Informationsgehalt, dass zu der betreffenden Objekteigenschaft keine Informationen vorhanden sind. Daraus wiederum folgt, dass jede Information, die durch *null* repräsentiert wird, verloren geht. Dies ist genau dann der Fall, wenn entweder die Ausprägung einer existierenden Objekteigenschaft nicht exakt bekannt ist (z.B. das Alter einer Person ist gar nicht oder nur begrenzt (Alter ≥ 18) bekannt), die Ausprägung zwar bekannt aber nicht konstant ist und innerhalb eines begrenzten Wertebereichs variiert (z.B. das schwankende Gewicht einer Person) oder diese Eigenschaft für das betreffende Objekt zumindest vorübergehend schlicht nicht vorhanden ist (z.B. eine Person hat kein Telefon). Das Informationen über die Nichtexistenz von Objekteigenschaften ebenfalls auf *null* abgebildet werden, zeigt, dass es sich bei den durch *null* repräsentierten Informationen nicht immer um unzureichendes Wissen und somit nicht zwangsweise um das Resultat einer mangelnden Informationserhebung handeln muss. Denn während die Unbekanntheit eines Wertes eine Unkenntnis darstellt, die durch vorhandene Teilinformationen lediglich eingeschränkt werden kann, ist die Nichtexistenz eines Wertes mit keinerlei Unwissen verbunden. Wie eine bekannte Ausprägung einer Objekteigenschaft stellt sie eine eindeutige und klare Information dar (in der Aussage „Die Person Müller hat kein Telefon“ ist keinerlei Unwissen enthalten). Diese exakte Erkenntnis (die Erkenntnis der Nichtexistenz) geht bei der Abbildung auf *null* jedoch wieder verloren und stellt aufgrund der fehlenden Unkenntnis den qualitativ schwerwiegendsten Informationsverlust dar. Als ein den bei der Datenspeicherung auftretenden Informationsverlust veranschaulichendes Beispiel soll im Folgenden eine Relation *Schüler_ belegt_ Fach(Schüler,Fach,Note)* dienen. In dieser Relation soll gespeichert werden, welcher Schüler (Name) welches Fach (Bezeichnung) belegt oder belegt hat. Desweiteren kann ein Schüler eine Prüfung über eben dieses Fach ablegen. Die resultierende Note ist für deutsche Oberstufen typischerweise eine Zahl von 0-15 und wird nach Abschluß der Prüfung in dem Attribut Note vermerkt. Nach der Informationserhebung ist bekannt, dass der Schüler Klaus im Fach Mathematik noch keine Prüfung abgelegt hat und dass der Schüler Hans im selben Fach bereits die Prüfung sowohl abgelegt als auch bestanden (Note 5-15) hat, seine genaue Note jedoch unbekannt ist. Vom Schüler Karl ist nur bekannt, dass er die Mathematikprüfung abgelegt hat, mit welcher Note er diese abgeschlossen hat, ist gänzlich unbekannt. Vom Schüler Gustav ist nur bekannt, dass er das Fach Mathematik belegt hat; über die Tatsache, ob er eine Prüfung abgelegt hat oder nicht, liegen keinerlei Informationen vor. Der aus einer idealen und informationserhaltenden Abbildung resultierende Datenbestand ist in Tabelle 1 dargestellt ('-' für nicht existent und '?' für keinerlei Informationen). Die Information, die über das Prüfungsergebniss des jeweiligen Schülers bekannt ist, lässt sich allerdings in keinem der vier Fälle durch eine der Zahlen von 0-15 darstellen. Demzufolge erfolgt bei der Verwendung eines auf dem Datenmodell DM_L^{SQL} basierenden Schemas jedesmal der Eintrag des Nullwertes *null* (siehe Tabelle 2).

Schüler	Fach	Note
...	...	...
Klaus	Mathematik	-
Hans	Mathematik	5-15
Karl	Mathematik	0-15
Gustav	Mathematik	?
...	...	...

Tabelle 1: Ergebnis einer idealen, verlustlosen Datenspeicherung

Schüler	Fach	Note
...	...	...
Klaus	Mathematik	*null*
Hans	Mathematik	*null*
Karl	Mathematik	*null*
Gustav	Mathematik	*null*
...	...	...

Tabelle 2: Ergebnis einer verlustbehafteten Datenspeicherung

Will nun ein Nutzer Informationen aus den Daten dieser Relation gewinnen, kann er zwischen den einzelnen Nullwert-Einträgen nicht mehr unterscheiden und muss um falsche Schlussfolgerungen zu vermeiden, für jeden Schüler annehmen, dass keine Informationen über dessen Prüfungsstand vorliegen. Demzufolge würden auf eine Anfrage, welche Schüler die Mathematikprüfung bestanden haben (Note ≥ 5), keiner der vier obigen Schüler als sicheres und alle als mögliches Ergebniss

resultieren. Dabei lagen anfangs zumindest für Klaus (Prüfung noch nicht bestanden) und Hans (Prüfung sicher bestanden) ausreichend Informationen vor, um sie bezüglich der gestellten Anfrage eindeutig und klar qualifizieren zu können.

3 Verwandte Arbeiten

Die unzureichende Modellierungsmächtigkeit der aktuell dominierenden relationalen Datenmodelle war bereits in den vergangenen Jahren und Jahrzehnten Bestandteil verschiedenartiger Lösungsansätze. Während eine Vielzahl an früheren Ansätzen das Nullwert-Konzept verwendeten (u.a. [10], [6], [7], [3], [5], [1], [2]), bauten andere Modelle, wie z.B. Wahrscheinlichkeitstheoretische- (u.a. [4]) oder Fuzzy-relationale Datenmodelle (u.a. [9]) auf völlig neuen Konzepten auf. Da sich nach der bisherigen Ansicht mehrere Nullwert-Arten nicht in der drei-wertigen Logik interpretieren lassen, basieren die auf dem Nullwert-Konzept beruhenden Ansätze auf vier- oder mehrwertigen Logiken. Alle oben genannten Ansätze haben gemeinsam, dass sie sich bereits auf fundamentaler Ebene von den aktuell dominierenden Modellen unterscheiden. Obgleich diese Ansätze mehr oder weniger zur Verringerung des Informationsverlusts während der Datenspeicherung gegeignet sind, hat sich bisher keines der aus diesen Ansätzen resultierenden Modelle durchsetzen können. Dies lässt sich auf zwei wesentliche Eigenschaften zurückführen. Zum Einem besitzen viele dieser Ansätze eine sehr hohe Komplexität (z.B. resultierend aus vier- oder mehrwertigen Logiken), welche die Verwendung für einfache Datenbanknutzer sehr erschwert. Zum Anderen führen die gravierenden Unterscheidungungen fundamentaler Modelleigenschaften wiederum zu einem stark veränderten Verhalten der resultierenden Datenbanksysteme. Aktuelle Datenbanksysteme sind jedoch seit Jahren etabliert, aktuell vorhandene Datenbank-Anwendungen sind auf diese ausgerichtet und sowohl Datenbankanbieter als auch -nutzer haben sich an deren Systemverhalten gewöhnt. Die Aussicht auf durch einen Wechsel des zugrundeliegenden Datenmodells entstehende Veränderungen dieses Systemverhaltens finden daher nur wenig Anklang in der breiten Masse der Datenbank-Gemeinde. Die fehlende Akzeptanz der bisherigen Lösungsansätze ist neben der bereits erwähnten Komplexität daher wohl hauptsächlich durch marktwirtschaftliche (Hohe Kosten bei Systemumstellung) und psychologische Faktoren (Umstellung bei der Systemnutzung) verschuldet.

4 Lösungsansatz

Die bei der Datenspeicherung in einen relationalen Datenbanksystem vollzogene Abbildung von Informationen auf operationale Datensätze ist nur dann verlustfrei, wenn sie injektiv und somit umkehrbar ist. Identische Abbildungen verschiedener Informationen und der damit verbundene Informationsverlust sollten daher auf ein Minimum reduziert werden. Um auch den aus den marktwirtschaftlichen und psychologischen Faktoren resultierenden Anforderungen zu genügen, erfolgte der Entwurf eines logischen Datenmodells, das den bestehenden Informationsverlust durch eine erhöhte Modellierungsmächtigkeit beschränken, zugleich aber auch die Schwächen der bisherigen Ansätze in Form einer mangelnden Akzeptanz vermeidet kann. Die erhöhte Modellierungsmächtigkeit ist, wie bereits aus anderen Ansätze bekannt, durch die Hinzunahme neuer Nullwerte erfolgt. Zur Vermeidung einer fehlenden Akzeptanz wurden neben einer geringen Komplexität und einer intuitiven Benutzung besonders weitreichende Abweichungen bezüglich des Datenmodells DM_L^{SQL} vermieden. Die sich daraus ergebende Modell-Kompatibilität zum Datenmodell DM_L^{SQL} schließt insbesondere den Erhalt der dreiwertigen Logik und andere an den bisherigen Nullwert *null* verknüpften Modellkomponenten mit ein. Der Lösungsansatz basiert dabei vornehmlich auf der Tatsache, dass eine sinnvolle Auswertung von relationalen Vergleichsoperatoren bezüglich verschiedener Nullwerte innerhalb der drei-wertigen Logik, entgegen weitläufiger Meinung, möglich ist. Da sich eine allgemeine Modellierung von partiellen Informa-

tionen hinsichtlich der beabsichtigten Modell-Kompatibilität als nicht trivial erweist, beschränkt sich dieses Modell zunächst auf eine verlustlose Darstellung von nichtexistenten und existenten aber unbekannten Objekteigenschaften. Eine allgemeine Modellierung partieller Informationen soll dann durch spätere Erweiterungen erfolgen, ohne dabei die Kompatibilität einzubüßen oder, falls nicht anders möglich, die auftretenden Abweichungen zumindest minimal zu halten.

Um das oben genannte Ziel zu erreichen, wurden drei Nullwerte eingeführt. Neben dem bereits bekannten Nullwert *null*, welcher zur Repräsentation des Falles, dass zu einem Wert keinerlei Informationen vorliegen verwendet wird, handelt es sich dabei um den Nullwert *nE* zur Darstellung nicht existender und den Nullwert *uk* zur Darstellung existierender aber unbekannter Objekteigenschaften. Um die durch die zusätzlichen Nullwerte dargestellten Informationen auch wieder aus den gespeicherten Daten gewinnen zu können, wurden neben dem aktuell vorhandenen und an die neuen Nullwerte angepaßten $isNull$-Prädikat noch die neu definierten Prädikate $isExistent$, $isKnown$ und $isSpecified$ benötigt. Anforderungen an Attributwerte, wie sie sich zum Beispiel aus der Notwendigkeit eindeutiger Primärschlüssel ergeben, erforderten Maßnahmen, die eine solche Einschränkungen der Attributsausprägungen auch in der Gegenwart mehrerer Nullwerte gewährleisten können, bzw. eine Umsetzung dieser ermöglichen. Dafür reichte die bisherige Spezifikation NOT NULL nicht mehr aus und musste um die weiteren Spezifikationen EXISTENT, KNOWN und SPECIFIED ergänzt werden. Weitere wichtige und zubeachtende Aspekte sind sowohl die Auswertungen der Nullwerte in arithmetischen Funktionen und Aggregationsfunktionen als auch deren Verwendung in weiteren Nullwert bezogenen Konzepten wie Outer Unions, Outer Joins oder in der Sortierung von Tupeln. Eingehendere Erläuterungen der hinzugefügten Nullwerte, deren Auswertung innerhalb der drei-wertigen Logik sowie die Gestaltung der anderen bereits kurz angesprochenen und für die Handhabung mehrerer Nullwerte notwendigen Konzepte sind Teil des Vortrags und werden dort genauer behandelt.

Literatur

[1] K. bun Yue. A More General Model For Handling Missing Information In Relational DataBases Using A 3-Valued Logic. *SIGMOD Record*, 20(3):43–49, 1991.

[2] K. S. Candan, J. Grant, and V. S. Subrahmanian. A Unified Treatment of Null Values Using Constraints. *Inf. Sci.*, 98(1-4):99–156, 1997.

[3] E. F. Codd. More Commentary on Missing Information in Relational Databases (Applicable and Inapplicable Information). *SIGMOD Record*, 16(1):42–50, 1987.

[4] D. Dey and S. Sarkar. A Probabilistic Relational Model and Algebra. *ACM Trans. Database Syst.*, 21(3):339–369, 1996.

[5] G. H. Gessert. Four valued logic for relational database systems. *SIGMOD Record*, 19(1):29–35, 1990.

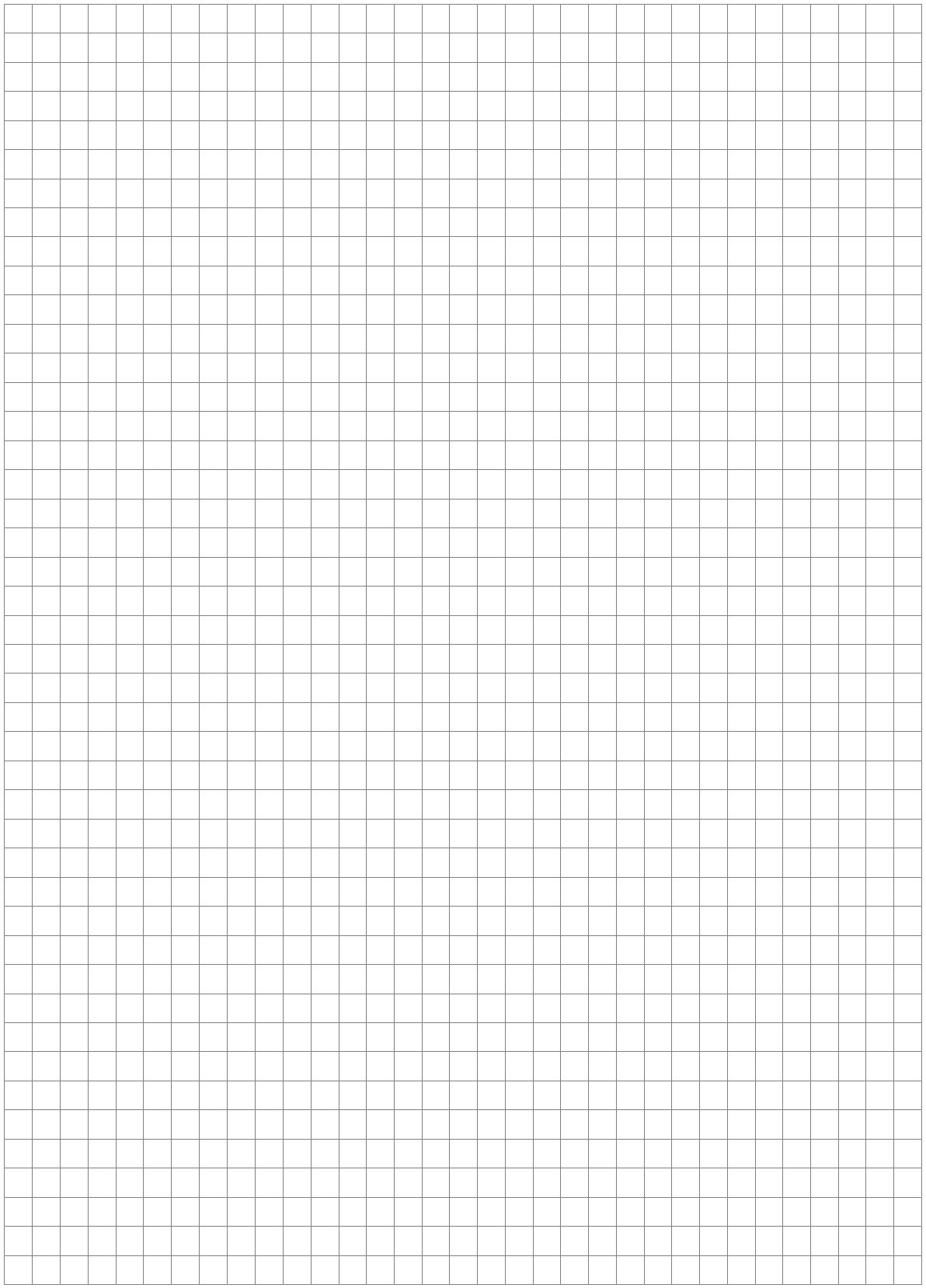
[6] T. Imielinski and W. Lipski. Incomplete Information in Relational Databases. *J. ACM*, 31(4):761–791, 1984.

[7] A. M. Keller and M. Winslett. On the Use of an Extended Relational Model to Handle Changing Incomplete Information. *IEEE Trans. Software Eng.*, 11(7):620–633, 1985.

[8] F. Naumann et al. Completeness of integrated information sources. *Inf. Syst.*, 29(7):583–615, 2004.

[9] G. D. Tré et al. Null values in fuzzy databases. *J. Intell. Inf. Syst.*, 30(2):93–114, 2008.

[10] C. Zaniolo. Database relations with null values . In *PODS*, pages 27–33, 1982.



Vergleich von Strategien zum Clustern von Daten mit fehlenden Werten

Ludmila Himmelspach
Institut für Informatik
Heinrich-Heine-Universität Düsseldorf
D-40225 Düsseldorf, Deutschland
himmelspach@cs.uni-duesseldorf.de

Zusammenfassung

Die klassischen Methoden zur Clusteranalyse wurden entwickelt um auf vollständigen Daten Analysen durchzuführen. Oft fehlen aber in Daten einzelne Werte — systematisch oder unsystematisch —, z.B. infolge der Probleme bei der Datenerfassung, Datenübertragung, Datenbereinigung oder weil Daten aus unterschiedlichen Quellen stammen. Demzufolge können die traditionellen Clusteringmethoden zur Analyse solcher Daten nicht ohne weiteres angewendet werden. Im Rahmen dieses Beitrags werden unterschiedliche Strategien zum Umgang mit fehlenden Werten in Daten für das Clusteringproblem vorgestellt, analysiert und miteinander verglichen. Dabei wird das besondere Augenmerk auf die Untersuchung der Leistungsfähigkeit dieser Verfahren in Abhängigkeit von den Ausfallmechanismen, die den fehlenden Werten zugrundeliegen, und von dem Anteil fehlender Werte in Daten gelegt.

1 Einleitung

Mit dem rasanten Anstieg an Möglichkeiten große Datenmengen elektronisch zu erfassen und zu speichern, haben auch Werkzeuge zur Datenanalyse stark an Bedeutung gewonnen. Diese großen Mengen von Daten können viel potentiell wichtiges Wissen enthalten, das aber zuerst im Rahmen eines *Knowledge Discovery in Databases*-Prozess aus den Daten extrahiert werden muss. Bei der Analyse der Daten geht es oft im ersten Schritt darum auf der riesigen Datenmenge Gruppen von ähnlichen Objekten zu identifizieren, was die Aufgabe der Clusteranalyse ist. Die Methoden der Clusteranalyse finden in vielen Bereichen ihre Anwendung, einschließlich Database Marketing, Web-Analyse, Information Retrieval, Bioinformatik und weiteren.

Den Ausgangspunkt für die Clusteranalyse bilden Daten. Dabei handelt es sich um eine Menge von Merkmalswerten, die in Form einer Datenmatrix für die Auswertung vorliegen. Oft kommen aber in Daten fehlende Werte vor, die z.B. durch Fehler bei der Datenerfassung, Datenübertragung oder Datenbereinigung verursacht werden konnten. Die fehlenden Werte können einer zufälligen Anordnung oder nach bestimmten Mustern in Datenmatrizen vorkommen. Sie können zufälligen oder systematischen Ausfallmechanismen unterliegen.

Die klassischen Methoden zur Clusteranalyse wurden entwickelt, um auf vollständigen Datenmatrizen Analysen durchzuführen. In den Fällen, wo die Vervollständigung der Datensätze durch Datennacherhebung z.B. aus Kosten- oder Zeitgründen unerwünscht oder sogar unmöglich ist, braucht man Analysemethoden, die mit dem Problem fehlender Werte in Daten umgehen können. Zur Behandlung von Daten, die fehlende Werte beinhalten, gibt es im Allgemeinen drei verschiedene Ansätze [LR02, Wag04]. Der erste Ansatz basiert auf der Eliminierung der Datensätze bzw. Merkmale, die fehlende Werte aufweisen. Beim zweiten Ansatz werden fehlende Werte im Rahmen einer Datenvorverarbeitung geschätzt. Der dritte Ansatz besteht darin, die datenanalytischen Verfahren für den Umgang mit fehlenden Werten zu adaptieren. Im Rahmen dieses Beitrags werden diese drei Strategien zum Umgang mit fehlenden Werten in Daten

für das Clusteringproblem am Beispiel des Fuzzy C-Means-Algorithmus anhand eines geeigneten Datensatzes analysiert und miteinander verglichen. Dabei untersuchen wir insbesondere die Leistungsfähigkeit dieser Verfahren in Abhängigkeit von den Ausfallmechanismen, die den fehlenden Werten zugrundeliegen, und von dem Anteil fehlender Werte im Datensatz.

2 Grundlagen

2.1 Fuzzy C-Means-Algorithmus

Der *Fuzzy C-Means-Algorithmus (FCM)* gehört zu den partitionierenden Clusteringalgorithmen, d.h. die zu klassifizierende Datenmenge wird vollständig in eine vorgegebene Anzahl von Cluster zerlegt. Im Unterschied zu den klassischen partitionierenden Clusteringmethoden wird die Zuordnung der Datenpunkte zu den Clustern bei FCM durch die Zugehörigkeitsgrade ausgedrückt [Bez81]. Der Zugehörigkeitsgrad eines Objekts bezüglich eines Clusters drückt aus, wie sicher dieses Objekt dem Cluster zuzuordnen ist. Die Zugehörigkeitsgrade werden basierend auf dem Abstand der Datenpunkte zu den Clustern berechnet und liegen im Intervall zwischen 0 und 1. Dabei zeigt 0 keine Zugehörigkeit des Objektes zu dem betreffenden Cluster. Ein Zugehörigkeitsgrad von 1 zeigt an, dass das Objekt dem Cluster mit Sicherheit zuzuordnen ist.

Wie die meisten partitionierenden Clusteringverfahren findet auch FCM eine optimale Zerlegung der Datenmenge durch die Minimierung der Zielfunktion. Die Zielfunktion für FCM berechnet für alle Cluster die Summen der quadrierten und durch die Zugehörigkeitsgrade gewichteten Abstände der Datenpunkte zu den jeweiligen Clusterzentren und addiert diese Teilsummen. Das heißt, die Cluster sollen so gebildet werden, dass die Abstände der Datenpunkte zu den Clusterzentren minimal sind. Da die Zielfunktion nicht direkt optimiert werden kann, wird sie in jedem Iterationsschritt von FCM bezüglich der Zugehörigkeitsgrade und Clusterzentren minimiert.

2.2 Arten von Ausfallmechanismen

Ein wichtiger Faktor bei der Auswahl eines datenanalytischen Verfahrens, das mit fehlenden Werten in Daten umgehen kann, ist der zugrundeliegende Mechanismus, der zum Ausfall von Daten geführt hat. Neben dem zufälligen Fehlen von Werten in Datenmatrizen kann es sein, dass das Fehlen eines Wertes von der Ausprägung seines Attributs oder von den Ausprägungen anderer Attribute abhängt. Grundsätzlich werden in der Literatur zwei Arten von Ausfallmechanismen unterschieden: unsystematischer (d.h. zufällig fehlend) und systematischer (d.h. nicht zufällig fehlend) Ausfallmechanismus [Ban95]. Der systematische Ausfallmechanismus liegt vor, wenn das Fehlen der Werte von der Ausprägung des Merkmals selbst abhängt, in dem sie fehlen. Die fehlenden Daten werden dann als *„not missing at random“ (NMAR)* bezeichnet [LR02]. Der unsystematische Ausfallmechanismus kann zusätzlich in zwei Klassen eingeteilt werden: *„missing at random“* und *„missing completely at random“*. Die fehlenden Werte in der Datenmatrix werden als *„missing at random“ (MAR)* bezeichnet, wenn das Fehlen der Werte allein von den Ausprägungen der beobachteten Merkmale abhängt. Wenn das Fehlen der Werte in der Datenmatrix unabhängig von Ausprägungen der Attribute ist, unabhängig davon, ob sie beobachtet wurden oder fehlen, dann spricht man von fehlenden Werten *„missing completely at random“ (MCAR).*

3 Strategien zum Umgang mit fehlenden Werten

3.1 Eliminierungsverfahren

Die einfachste Methode mit unvollständigen Daten umzugehen ist die Datensätze oder Merkmale mit fehlenden Werten aus der Datenmatrix zu eliminieren und die Datenanalyse nur auf Grund der vollständig erhobenen Datenobjekte bzw. Merkmale durchzuführen [LR02, Ban95].

Werden Datensätze mit fehlenden Werten von der Analyse ausgeschlossen, wird dieses Verfahren in der Literatur als *„complete-case analysis“* bezeichnet. Werden Merkmale, in denen Datensätze fehlende Werte aufweisen, bei der Datenanalyse nicht betrachtet, so wird dieses Verfahren als *„compelete-variable analysis“* bezeichnet. Die erste Methode wird im Allgemeinen dann angewendet, wenn der Anteil der Datenobjekte mit fehlenden Werten relativ gering ist und bei der Datenanalyse nicht alle Datenobjekte berücksichtigt werden müssen. Ist der Anteil der Datensätze mit fehlenden Werten hoch oder müssen alle Datensätze klassifiziert werden, so eignet sich die zweite Methode besser, wobei auch hier der Anteil der Merkmale mit fehlenden Werten nicht zu hoch sein darf, da sonst diese Vorgehensweise zum Verlust einer für das Clustering aussagekräftigen Dimension führen kann. Trotz der Nachteile wird dieses Verfahren häufig als Default-Ansatz implementiert und als Maßstab für andere Verfahren verwendet.

3.2 Imputationverfahren

Eine andere Möglichkeit mit fehlenden Werten in Datenmatrizen bei der Datenanalyse umzugehen ist die fehlenden Werte im Rahmen einer Datenvorverarbeitung zu schätzen. In der Literatur wird für diese Methode der Begriff *„missing value imputation“* verwendet. Neben der Imputation durch zufällige Auswahl der vorhandenen Werte aus der Datenmatrix, gibt es zahlreiche statistische Verfahren, um fehlende Werte zu schätzen. Die Imputationstechniken reichen von den einfachsten wie z.B. Ergänzung der unvollständigen Datenmatrix durch Minimum, Maximum oder Mittelwert vorhandener Werte, bis zu statistischen Verfahren wie z.B. mittels Regressions-, Varianz- oder Hauptkomponentenanalyse, die versuchen die Zusammenhänge zwischen den Attributen aufzudecken und diese dann zur Bestimmung der Imputationswerte zu nutzen [LR02]. Auch der Expectation-Maximization-Algorithmus (EM) wird oft zur Schätzung fehlender Werte verwendet [DLR77]. Der Hauptvorteil der Imputationsverfahren liegt darin, dass die anschließende Datenanalyse auf der vollständigen Datenmatrix wie im Fall ohne fehlende Werte erfolgen kann. Der Nachteil dieser Vorgehensweise ist neben dem hohen Rechenaufwand jedoch, dass die Ergebnisse der Datenanalyse durch die verwendeten Imputationstechniken stark beeinflußt werden, da während der Datenanalyse zwischen den beobachteten und geschätzten Werten nicht mehr unterschieden wird.

3.3 Adaptierte Clusteringverfahren für Daten mit fehlenden Werten

Der letzte Ansatz für den Umgang mit fehlenden Werten in Daten ist die datenanalytischen Verfahren so zu ändern, dass diese bei der Analyse die Datenobjekte mit fehlenden Werten im vollen Umfang berücksichtigen. Die Strategien den Fuzzy C-Means-Algorithmus an Daten mit fehlenden Werten zu adaptieren, kann man im Allgemeinen in zwei Kategorien unterteilen. Zur ersten Kategorie gehören Verfahren, die beim Clustern unvollständiger Datensätze nur vorhandene Werte einbeziehen. Zur zweiten Kategorie gehören Verfahren, die beim Clustern in jedem Iterationsschritt fehlende Werte in Abhängigkeit von den Clusterzentren oder vorhandenen Attributwerten schätzen und ersetzen. Wir beschränken unsere Betrachtung hier auf drei Verfahren, wobei die ersten beiden zu der ersten und das dritte zu der zweiten Kategorie gehören.

Whole-data strategy (WDS): Bei dieser Methode werden zuerst alle vollständig vorhandenen Datensätze mit FCM klassifiziert. Danach werden die Daten mit fehlenden Werten unter Berechnung der partiellen Distanzen jeweils dem nächstliegenden Clusterzentrum zugeordnet [HB01].

Partial distance strategy (PDS): Dieses Verfahren verwendet partielle Distanzen zwischen den Datenpunkten mit fehlenden Werten [HB01].

Nearest prototype strategy (NPS): Die fehlenden Attributwerte eines Datenpunktes werden durch die entsprechenden Werte des nächstliegenden Clusterzentrums in jedem Itera-

tionsschritt ersetzt [HB01]. Bei der Distanzberechnung wird dabei die partielle Distanzfunktion verwendet [Dix79].

Eine Übersicht über weitere Strategien und sowie deren Vergleich kann in [Him08] gefunden werden.

4 Datenexperimente und Ergebnisse

Die oben beschriebenen Verfahren zum Umgang mit fehlenden Werten wurden am Beispiel von FCM anhand eines künstlichen Datensatzes untersucht, der durch eine Mischung von drei 3-dimensionalen Gaußverteilungen generiert wurde. Der Datensatz besteht aus 100 Datenpunkten, wobei sich diese gleichmäßig auf drei Cluster verteilen (siehe Abbildung 1). Da die abhängigen Dimensionen für das Clustering keine zusätzlichen Informationen liefern, wurden die Daten so generiert, dass es keine Abhängigkeiten zwischen den Werten verschiedener Dimensionen gibt.

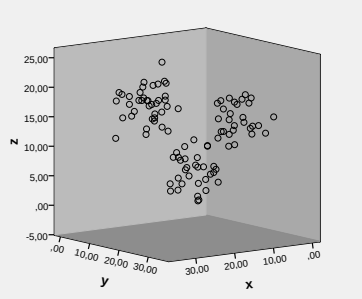


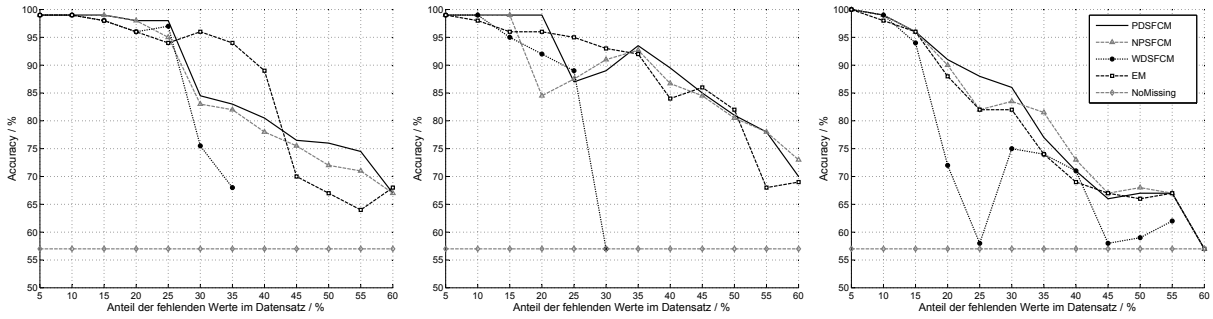
Abbildung 1: Testdatensatz

Um die im Abschnitt 3 vorgestellten Verfahren hinsichtlich ihrer Leistungsfähigkeit anhand der Daten testen zu können, wurden aus dem Datensatz Werte in zwei der drei Dimensionen schrittweise entfernt, wobei sich der Anteil fehlender Werte auf die Gesamtanzahl der Werte im Datensatz bezieht. Um auch prüfen zu können, ob die Leistungsfähigkeit der einzelnen Algorithmen von den unterschiedlichen Ausfallmechanismen abhängt, wurden die Werte gemäß der im Abschnitt 2.2 beschriebenen Ausfallmechanismen MCAR, MAR und NMAR aus dem Datensatz entfernt.

Abbildung 2 zeigt die Ergebnisse der Performance-Analyse der Verfahren PDSFCM, NPSFCM, WDSFCM, EM und NoMissing in Abhängigkeit von dem Anteil fehlender Werte MCAR, MAR und NMAR für die Attribute y und z. Bei den ersten drei Verfahren handelt es sich um die Algorithmen aus dem Abschnitt 3.3. Im Rahmen des EM-Verfahrens wurden fehlende Werte mittels EM-Algorithmus (vgl. Abschnitt 3.2) geschätzt und anschließend mit FCM klassifiziert. Das Verfahren NoMissing entspricht dem „complete-variable“-Ansatz aus Abschnitt 3.1. Als Bewertungsmaß für die Algorithmen wurde der über 10 Durchläufe gemittelte Wert für die Accuracy verwendet, der den Anteil richtig klassifizierter Datenpunkte zur Gesamtanzahl der Objekte im Datensatz prozentual angibt.

Wie man in den Diagrammen erkennen kann, unterscheiden sich die Algorithmen PDSFCM, NPSFCM, WDSFCM und EM hinsichtlich ihrer Leistungsfähigkeit nur unwesentlich voneinander und liegen weit über den Ergebnissen der anderen zwei Verfahren. Die Accuracy-Werte für diese Algorithmen liegen bei einem kleinen Anteil (bis 20%) fehlender Werte im Datensatz für alle Ausfallmechanismen über 90%. Mit steigender Anzahl fehlender Werte im Datensatz unterscheidet sich die Leistungsfähigkeit der Algorithmen in Abhängigkeit von den zugrundeliegenden Ausfallmechanismen. Die besten Ergebnisse sind beim Ausfallmechanismus MAR zu beobachten. Wenn hingegen der Ausfallmechanismus NMAR vorliegt, fallen die Ergebnisse der Algorithmen am schlechtesten aus. Die guten Ergebnisse beim Vorliegen des Ausfallmechanismus MAR sind unter anderem dadurch zu erklären, dass der Anteil der Datensätze mit zwei fehlenden Werten in Vergleich zu anderen Ausfallmechanismen sehr klein ist. Andere Experimente haben jedoch gezeigt, dass die Leistungsfähigkeit der Algorithmen auf Daten mit fehlenden Werten MAR mit hohem Anteil der Datensätze mit zwei fehlenden Werten viel schlechter ist als z.B. beim Vorliegen des Ausfallmechanismus MCAR. Bei einem hohen Anteil fehlender Werte lag die Accuracy für diese Algorithmen sogar unter der für das NoMissing-Verfahren (vgl. [Him08]).

Wie Diagramme in Abbildung 2 zeigen, konnte die Accuracy für den Algorithmus WDSFCM nicht immer berechnet werden. Das liegt daran, dass es ab einem bestimmten Anteil fehlender Werte im Datensatz keine vollständigen Datenobjekte mehr gab, was den Einsatz von WDSFCM



(a) (b) (c)

Abbildung 2: Accuracy für verschiedene Algorithmen in Abhängigkeit von dem Anteil fehlender Werte (a) MCAR, (b) MAR und (c) NMAR für die Attribute y und z.

unmöglich machte. Da dieser Algorithmus Clusterzentren ausschließlich anhand der vollständigen Datensätze berechnet, hängt die Verteilung der Datenpunkte auf Cluster sehr stark von der Verteilung der vollständigen Datensätze ab, was die schlechten Ergebnisse des Algorithmus bei einem hohen Anteil fehlender Werte im Datensatz erklärt.

5 Zusammenfassung und Ausblick

In dieser Arbeit wurden unterschiedliche Strategien zum Umgang mit fehlenden Werten in Daten für das Clusteringproblem anhand eines künstlichen Datensatzes analysiert und miteinander verglichen. Die Testergebnisse haben gezeigt, dass es sinnvoll ist bei der Clusteranalyse alle vorhandenen Werte zu berücksichtigen. So haben die Imputations- und adaptierte Clusteringverfahren bei Experimenten viel bessere Ergebnisse erzielt als das Eliminierungsverfahren. Außerdem haben wir gezeigt, dass die Qualität der Ergebnisse stark von dem Ausfallmechanismus abhängt, der den fehlenden Werten zu Grunde liegt. Deswegen wird das nächste Forschungsziel sein die Verfahren für unterschiedliche Ausfallmechanismen anzupassen, um dadurch bessere Ergebnisse zu erzielen.

Literatur

[Ban95] U. Bankhofer. *Unvollständige Daten- und Distanzmatrizen in der Multivariaten Datenanalyse.* Eul, Bergisch-Gladbach, 1995.

[Bez81] J.C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms.* Kluwer Academic Publishers, 1981.

[Dix79] J.K. Dixon. Pattern Recognition with Partly Missing Data. *IEEE Transactions on System, Man and Cybernetics*, 9:617–621, 1979.

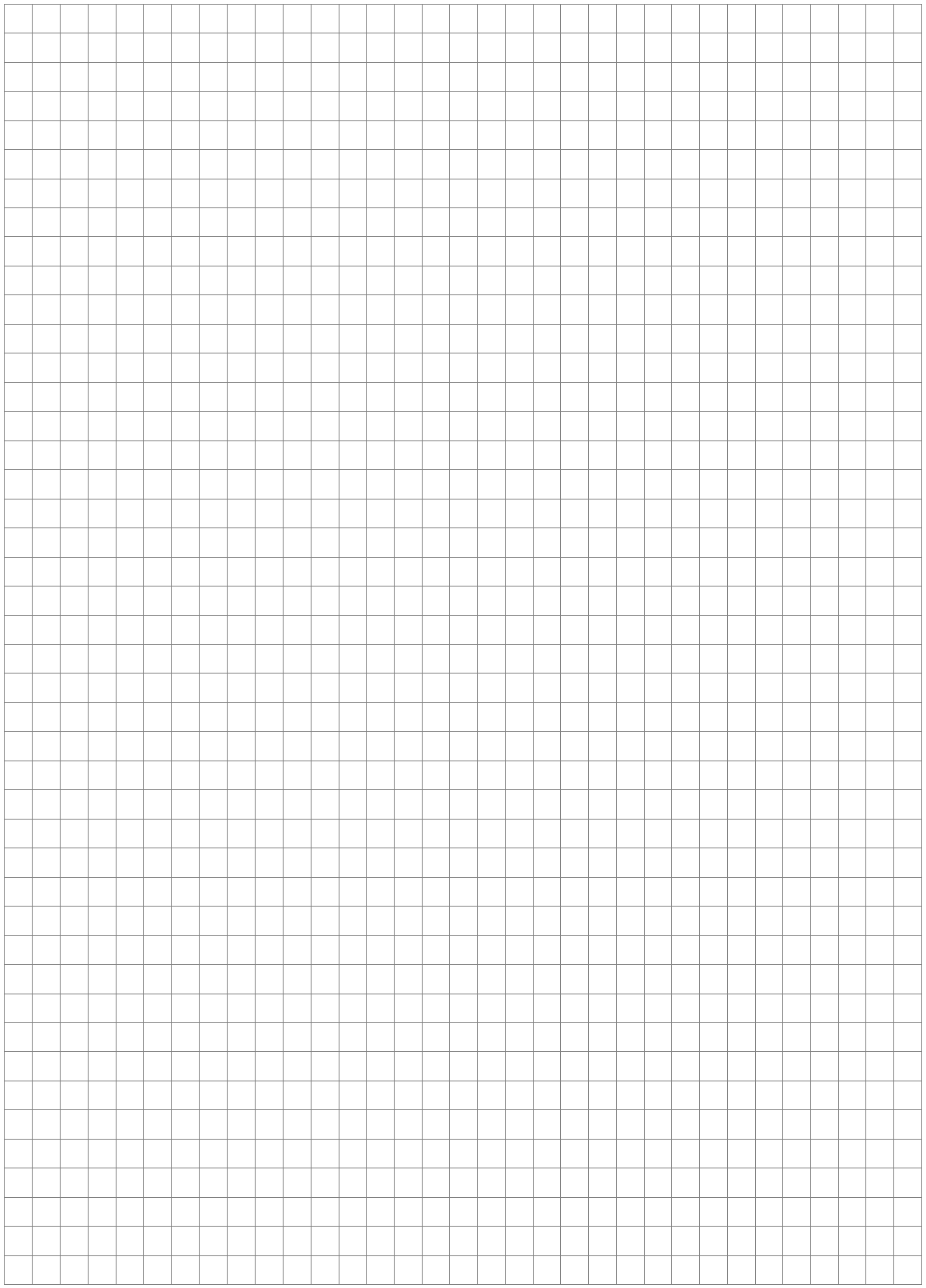
[DLR77] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum Likelyhood from Incomplete Data via EM Algorithm. *Journal of the Royal Statistical Society, Series B*, pages 1–31, 1977.

[HB01] R.J. Hathaway and J.C. Bezdek. Fuzzy c-means Clustering of Incomplete Data. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 31(5):735–744, 2001.

[Him08] L. Himmelspach. Clustering mit fehlenden Werten: Analyse und Vergleich. Masterarbeit, Institut für Informatik, Heinrich-Heine-Universität Düsseldorf, 2008.

[LR02] R.J. Little and D.B. Rubin. *Statistical Analysis with Missing Data.* John Wiley & Sons, 2002.

[Wag04] K. Wagstaff. Clustering with Missing Values: No Imputation Required. In *Classification, Clustering, and Data Mining Applications (Proceedings Meeting of the International Federation of Classification Societies)*, pages 649–658, 2004.



Archiving and Maintaining Curated Databases

Heiko Müller

University of Edinburgh, UK

hmueller@inf.ed.ac.uk

Abstract

Curated databases represent a substantial amount of effort by a dedicated group of people to produce a definitive description of some subject area. The value of curated databases lies in the quality of the data that has been manually collected, corrected, and annotated by human curators. Many curated databases are continuously modified and new releases being published on the Web. Given that curated databases act as publications, archiving them becomes a necessity to enable retrieval of particular database versions. A system trying to archive evolving databases on the Web faces several challenges. First and foremost, the systems needs to be able to efficiently maintain and query multiple snapshots of ever growing databases. Second, the system needs to be flexible enough to account for changes to the database structure and to handle data of varying quality. Third, the system needs to be robust and invulnerable to local failure to allow reliable long-term preservation of archived information. Our archive management system XArch addresses the first challenge by providing the functionality to maintain, populate, and query archives of database snapshots in hierarchical format. This presentation intends to give an overview of our ongoing efforts of improving XArch regarding (i) archiving evolving databases, (ii) supporting distributed archives, and (iii) using our archives and XArch as the basis of a system to create, maintain, and publish curated databases.

1 Introduction

Curated databases are databases that are populated and updated with a great deal of human effort [5]. The content of curated databases is usually collected through (i) consultation, verification, aggregation, and annotation of existing sources, or through (ii) processing, and interpretation of new raw data. A typical example of a curated database is the *Universal Protein Resource* (*UniProt*), the central repository for information of protein sequence and functional annotation [12]. The *UniProt consortium* has a large team of curators that extract biological information from the literature and perform numerous computational analyses to provide all relevant information about a particular protein. Another example of a curated database is the *CIA World Factbook* [1], a comprehensive resource of demographic data. The first unclassified Factbook was published in 1971. Since 1981 the Factbook has been published annually as a book. Recently, the *CIA* decided to exclusively publish the Factbook on the Web. Just as the Factbook, many curated databases act as publications on the Web that are currently replacing the traditional reference works found in libraries. For many database providers it has become common practice to overwrite existing database states when changes occur and publish new releases of the data on the Web. For example, the *CIA* publishes a new version of the Factbook approximately every two weeks. Thus, archiving becomes a necessity to (i) enable retrieval of particular cited database versions, and (ii) to support temporal queries to allow analysis of how certain objects changed over time. Such queries are especially interesting for the *CIA World Factbook*. For example, one might want to find out how the population of European countries changed over the past 30 years and how these changes correlate with energy consumption, or *Gross Domestic Product* change.

Curated databases are predominantly kept in well-organised hierarchical data formats having a key structure that provides a canonical identification for each element by the path in which it occurs and the values of some of its sub-elements. In [6] a nested merge approach to archiving is developed that efficiently stores multiple versions of hierarchical data in a compact archive by "pushing down" time and introducing timestamps as an extra attribute of the data. Archives of multiple database versions are generated by merging the versions into a single hierarchical data structure. Corresponding elements in different versions are identified based on their key values. The archiver stores each element only once in the merged hierarchy to reduce storage overhead. Archived elements are annotated with timestamps representing the sequence of version numbers in which the element appears. This approach has several advantages regarding storage space, retrieval of database versions, and tracking of object history. Based on the algorithms described in [6], we implemented the archive management system XArch [10]. The system allows one to create new archives, to merge new versions of data into existing archives, and execute both snapshot and temporal queries using a declarative query language.

Outline. Our initial approach to archiving makes the critical assumption that the structure of an archived database remains unchanged. This restriction, however, is almost certainly to become an issue when archiving a database over a long period of time. The first part of this presentation discusses our ongoing efforts of archiving and querying evolving databases. Creating archives in a stand-alone manner, however, appears infeasible and undesirable for long-term preservation. The natural solution to enhance reliability is to follow the model of traditional libraries to share the burden of archiving by employing a set of independent, distributed archivers. The second part of this presentation proposes a low-cost solution to distributed archiving. We are currently developing algorithms to synchronize and query distributed archives without requiring any central authority. The presented techniques cover some of the challenges for curated databases discussed in [5]. The last part of this presentation outlines how our archives and XArch may form the basis of a system to create, maintain, and publish curated databases.

2 Archiving Evolving Databases

We start with a brief description of our data model. Archives and database states (called *database snapshots*) are modeled as trees. We consider two types of nodes: (i) *element nodes* having a label and a key value, and (ii) *text nodes* having a value. Only element nodes may occur as internal nodes. Nodes in an archive additionally carry a timestamp that represents the sequence of version numbers in which they appear. Element keys are defined using key constraints. A *key specification* K is a set of *key definitions* $k = (q, s)$, where q is an absolute path of element labels and s is a *key value expression*. Each key definition (q, s), specifies a set of elements reachable by path q and defines how the key value is derived from the elements subtree. We distinguish between three types of key value expressions: (i) *existence*, (ii) *subtree*, and (iii) *values*. All element keys are relative keys, *i.e.,* the key value uniquely identifies an element among its siblings. Elements that are keyed by existence are keyed by their label. Elements that are keyed by subtree are uniquely identified by the value of their whole subtree (see [6] for definitions of subtree values). For elements e that are keyed by values an additional set of relative path expressions $\{p_1, \ldots, p_k\}$ is given. Each p_i specifies an element in the subtree of e whose value is used as part of the key value for e. These values are referred to as *key path values*. In XArch we currently use *XML* as the storage format of archives. Element nodes and timestamps are represented as *XML* elements. Text nodes are *XML* strings. *XML*, however, is not the only possible storage format and we are currently considering other formats like relational databases.

Schema Evolution. When archiving databases over a long period of time, schema changes almost certainly become an issue. Here, we distinguish between changes to (i) key path values, and (ii) to the key specification. If key path values are modified between different database

snapshots the archiver treats the corresponding elements as distinct elements and does not merge them. These changes are particularly common in the *CIA World Factbook* and they limit our ability of tracking an objects history. We are currently developing methods to detect such key path value changes based on complementary timestamps and similarity of element subtrees. Once we are able to detect these changes, we can merge the corresponding elements in a post-processing step and annotate their key values with appropriate timestamps. Thus, we need to extend our data model to allow timestamped key values for element nodes. Problems arise, however, when elements are being moved around between different parents. In order to handle these cases, we would have to extend our data model and consider archives as graphs instead of trees making it even more questionable whether *XML* is the appropriate storage format for our archives.

Changes to the key specification are currently handled as follows: A key specification itself is a tree whose nodes are elements that have a label, a type, and (for value keys) a set of key path expressions. Such a tree is easily transformed into a hierarchical dataset as described above. We are thereby enabled to maintain different key specifications within a separate archive. We recently extended XArch to handle database snapshots with different key specifications within a single archive. Instead of defining a single key specification for all database snapshots, each snapshot may have its own key specification. Before merging a snapshots into an archive we first merge its key specification into the *key specification archive*. When annotating key values for elements in the archive we find the appropriate key definition based on an elements timestamp. This approach models changes to the key specification as insertion and deletion of key definitions. In order to allow a more meaningful and compact description and representation of object histories additional operations are required. For example, when representing the renaming of an element by a delete and insert operation the problem again is to identify elements that are keyed by different key definitions but represent the same real-world object. In [9] a set of *schema modification operators* (*SMOs*) is defined that are capable of describing evolution of relational schemas. A *SMO* takes a schema version as input and produces a new schema version. Apart from creating and deleting tables and columns, *SMOs* allow for example to rename tables and columns, merge tables, and copy or move columns. These operations can easily be adopted to describe key specification modifications. In [7] a functional update language for *XML* (*FLUX*) is presented that also contains structure modifying operators like renaming and nesting of elements. We are looking into *SMOs* and *FLUX* to derive a set *key specification modification operations* capable to describe evolution key specifications. Similar to the approach in [9] we intend to use the information about structure evolution to rewrite queries that are written against the current key specification to retrieve data from snapshots structured under previous key specification.

3 Synchronizing Distributed Archives

Creating archives for long-term preservation in a stand-alone manner appears infeasible and undesirable for several reasons. First of all, having a stand-alone archive creates a single point of failure, making the preservation effort depending on usually unreliable hardware and storage media. Furthermore, the number of online available curated databases is large and only expected to increase in future. The exponential growth of many curated databases is well documented. Thus, maintaining archives for each of these databases requires high-performance systems with high network bandwidth and huge amounts of storage space. The natural solution to enable reliable and low-cost preservation is to share the burden of archiving. A viable approach is to employ a large network of independent, low-cost computers that cooperate to populate and query archives of online databases. Such an approach has for example being taken by *LOCKSS* (*Lots of Copies Keep Stuff Safe*), a system that aims at preserving access to journals and other archival information published on the Web [8]. The main assumption of *LOCKSS*, however, is that archived information like journal articles do not change.

We envision a scenario similar to *LOCKSS* of independent archivers that continuously archive snapshots of curated databases. The problem we are focusing on in our current research is to reconstruct the database history from a given set of individual archives that are maintained without central control. Our goal is to reconstruct the database history by defining a chronological order for all database snapshots in the archives. This problem is complicated for two reasons: First, data sources on the Web rarely provide unique version identifiers with each of their published snapshots. Second, the distributed scenario does not allow us to assume global unique timestamps for database snapshots in different archivers. While internally each archiver may maintain unique timestamps for each of its snapshots, timestamps provided by individual archivers may not necessarily be comparable. Common examples are unsynchronized clocks or incomparable timestamp formats. To define a chronological order in the absence of timestamps, we have to rely on the order of snapshots in individual archives and on the data itself.

The problem of estimating a chronological order for a given set of databases has been referred to as *seriation* [11]. Existing approaches for database seriation focus on small databases of Boolean data values. We are currently developing algorithms to allow seriation of large hierarchical datasets like those in our archives of curated databases. The main idea of our approach is to use a distance measure for pairs of snapshots that reflects their edit distance using insert and delete operations. Given a set of independent archives for the same database, we simply merge these archives into a single archive. The order in which we merge the archives is irrelevant. The edit distance for pairs of snapshots can then be computed efficiently by evaluating the timestamps in the resulting archive. We then compute a total order on all snapshots that (i) adheres to the partial order of snapshots defined by the individual archives, and (ii) minimizes the sum of edit distances of adjacent snapshots in the total order. The problem of seriation for a given set of partially ordered databases is shown to be of high complexity. We develop algorithms and heuristics that solve the problem efficiently and with high accuracy.

4 Maintaining Curated Databases

Maintaining curated databases presents a number of challenges for database research [5]. Archiving and evolution of structure are two of them that are already tackled by XArch. Others topics are propagation of annotations, citation, and provenance.

Citation. Initial proposals on how to make curated databases citable are given in [3]. Archives based on a keyed hierarchical data model satisfy some of the desiderata. First, each element within a hierarchical dataset is uniquely identified by its path and the key values of the elements in the path. Second, the archive records all database versions. Thus, the combination of element path, element keys, and timestamps allows to uniquely identify any element in a particular version of the database and to retrieve the data from the archive. We are currently working with the *IUPHAR receptor database* [2] on methods to publish current and past versions of the database on the Web and allow people to cite and retrieve any particular version of the data. One of the open research questions here is to develop generic and easy-to-use methods for generating appropriate Web pages from an existing archive.

Updates and Provenance. The majority of data in curated databases is derived or copied from other sources. Since the value of curated databases lies in the quality of the data that has been manually collected, corrected, and annotated by human curators, knowing the origin of curated data - its *provenance* - is particular important. In [4] general-purpose techniques for modeling and recording provenance for data that is copied among databases is presented. The techniques are based on a keyed hierarchical data model and a basic update language that allows to insert, and delete nodes, and copy nodes within or between curated databases. User actions while modifying a database are recorded in a convenient form that allows to ask questions such as when some data was first created, by what process did a value arrive in a database, or when

was a node or its subtree last modified. A insert and delete operator have recently been included in XArch, the copy operator remains future work. Each operator modifies the last snapshot in an archive and adds a new database snapshot to the archive. The provenance, *i.e.,* the operator that created a snapshot, is maintained in an appropriate way as described in [4]. Given the presented extensions we believe that XArch forms a viable tool to support the creation, manipulation, and evolution of curated databases.

Acknowledgments

The presented work is based on collaborations with Peter Buneman, James Cheney, and Floris Geerts from the *University of Edinburgh*.

References

[1] https://www.cia.gov/library/publications/the-world-factbook/index.html.

[2] http://www.iuphar-db.org/.

[3] P. Buneman. How to cite curated databases and how to make them citable. In *SSDBM '06: Proceedings of the 18th International Conference on Scientific and Statistical Database Management*, pages 195–203, Washington, DC, USA, 2006. IEEE Computer Society.

[4] P. Buneman, A. Chapman, and J. Cheney. Provenance management in curated databases. In *SIGMOD '06: Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, pages 539–550, New York, NY, USA, 2006. ACM.

[5] P. Buneman, J. Cheney, W.-C. Tan, and S. Vansummeren. Curated databases. In *PODS '08: Proceedings of the twenty-seventh ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 1–12, New York, NY, USA, 2008. ACM.

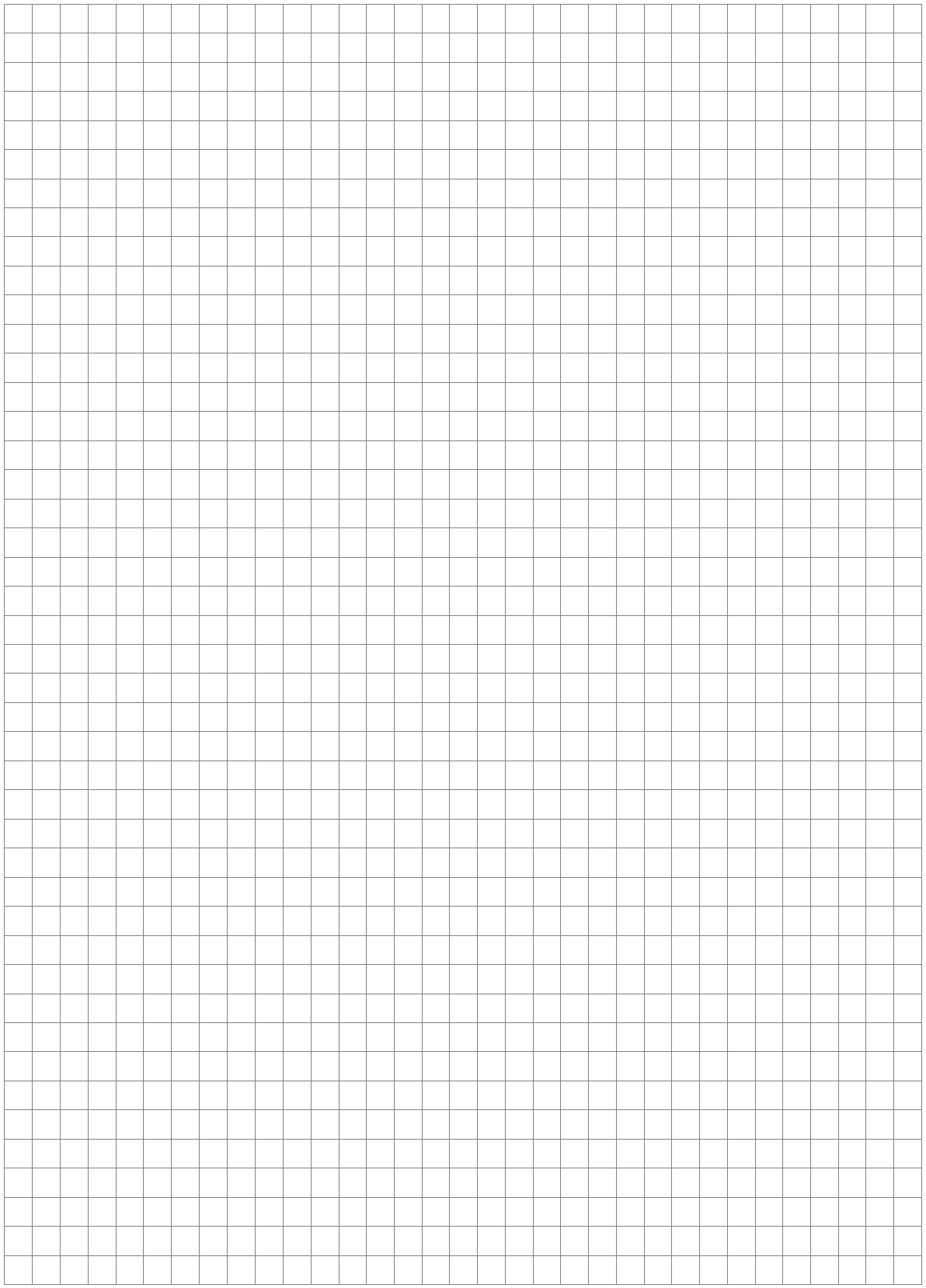
[6] P. Buneman, S. Khanna, K. Tajima, and W.-C. Tan. Archiving scientific data. *ACM Trans. Database Syst.*, 29(1):2–42, 2004.

[7] J. Cheney. Flux: functional updates for xml. In *ICFP '08: Proceeding of the 13th ACM SIGPLAN international conference on Functional programming*, pages 3–14, New York, NY, USA, 2008. ACM.

[8] P. Maniatis, M. Roussopoulos, T. J. Giuli, D. S. H. Rosenthal, and M. Baker. The lockss peer-to-peer digital preservation system. *ACM Trans. Comput. Syst.*, 23(1):2–50, 2005.

[9] H. J. Moon, C. A. Curino, A. Deutsch, C.-Y. Hou, and C. Zaniolo. Managing and querying transaction-time databases under schema evolution. *Proc. VLDB Endow.*, 1(1):882–895, 2008.

[10] H. Müller, P. Buneman, and I. Koltsidas. Xarch: archiving scientific and reference data. In *SIGMOD '08: Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1295–1298, New York, NY, USA, 2008. ACM.

[11] A. Ukkonen, M. Fortelius, and H. Mannila. Finding partial orders from unordered 0-1 data. In *KDD '05: Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 285–293, New York, NY, USA, 2005. ACM.

[12] Uniprot-Consortium. The universal protein resource (uniprot) 2009. *Nucleic acids research*, October 2008.