Martin-Luther-Universität
Halle-Wittenberg
Institut für Informatik
Datenbanken und Informationssysteme

17. Workshop "Grundlagen von Datenbanken"

Wörlitz, Mai 2005

Tagungsband

Stefan Braß
Christian Goldberg
(Hrsg.)

Vorwort

Der vorliegende Bericht enthält die Kurzfassungen der Vorträge, die für den 17. Workshop "Grundlagen von Datenbanken" angemeldet wurden. Der Workshop findet vom 17. bis 20. Mai 2005 in Wörlitz (Sachsen-Anhalt) statt.

Dieser Workshop gehört zu den regelmäßigen Aktivitäten des Arbeitskreises "Grundlagen von Informationssystemen" im Fachausschuss Datenbanken und Informationssysteme (DBIS) der Gesellschaft für Informatik. Primäres Anliegen dieser Workshop-Reihe (und des Arbeitskreises) ist es, die Kommunikation zwischen deutschsprachigen Forschergruppen zu fördern, die sich mit den theoretischen, konzeptionellen und methodischen Grundlagen von Daten-, Objekt-, Dokumenten- und Wissensbanken sowie des Semantic Web beschäftigen. Der Workshop stellt insbesondere auch ein Forum für Nachwuchswissenschaftler/-innen dar, die aktuelle, sich eventuell noch in der Entwicklung befindende Arbeiten in einem größeren Rahmen vorstellen und diskutieren wollen. Dabei ist auch die Teilnahme von schon etablierten Forscher/-innen wesentlich für den Erfolg dieses Workshops. Im Gegensatz zur häufig anonymen Atmosphäre großer Konferenzen bietet er in zwangloser Umgebung und reizvollem Umfeld die Gelegenheit zu einem sehr viel intensiveren Kontakt zu den übrigen Teilnehmenden. Wie die hohe Zahl der Anmeldungen auch wieder in diesem Jahr gezeigt hat, besteht an Workshops dieser Art und Thematik ein anhaltendes Interesse.

Die Workshops dieser Reihe fanden bislang in Niedersachsen (1989, 1990, 1991, 1992, 1994, 1995, 1997), Sachsen-Anhalt (1996, 2001, 2003), Mecklenburg-Vorpommern (1993, 2002), Baden-Württemberg (1998), Thüringen (1999), Schleswig-Holstein (2000) und Nordrhein-Westfalen (2004) statt. In diesem Jahr liegt der Tagungsort mitten im Dessau-Wörlitzer Gartenreich, welches seit 2000 zum Weltkulturerbe der UNESCO gehört. Der Wörlitzer Park, direkt vor der Tür des Tagungshotels gelegen, lädt gerade in dieser Jahreszeit zum Spazierengehen und Verweilen ein.

Halle (Saale), Mai 2005

Christian Goldberg
Stefan Braß

Wir danken unseren Sponsoren:

ORACLE Deutschland GmbH
http://www.oracle.de

Gesellschaft für Informatik e.V.
http://www.gi-ev.de

Martin-Luther-Universität Halle-Wittenberg
http://www.uni-halle.de

Inhaltsverzeichnis

Eingeladener Vortrag

Reguläre Beiträge

Eingeladener Vortrag

Reasoning im und für das Semantic Web

Wolfgang May
Georg-August-Universität Göttingen
may@informatik.uni-goettingen.de

Zusammenfassung

Das "Semantic Web" bietet (und erfordert) ein weites Spektrum an wissenschaftlichen Teilbereichen, von der darunterliegenden technischen Realisierung über Datenhaltung und Wissensrepräsentation, Kommunikation bis hin zu –teilweise unentscheidbaren– Logikformalismen. Damit bietet es sich insbesondere zur bereichsübergreifenden, wenn man seine Anwendungen mit einbezieht, auch zur fachübergreifenden Forschung an. In diesem Beitrag wird ein Überblick gegeben wie in dem EU FP6 NoE "ReWeRSe"[1] (Reasoning on the Web with Rules and Semantics) Forschungsgebiete aus verschiedenen Bereiche der theoretischen und praktischen Informatik kombiniert werden. Hierbei werden insbesondere im Umfeld des Bereiches "Datenbanken" relevante Aspekte betrachtet.

1 Einführung und Überblick

Das *Semantic Web* bietet (und erfordert) ein weites Spektrum an Teilbereichen. Die Basis bilden einerseits Netzwerk- und *Kommunikationsaspekte*, von der darunterliegenden technischen Realisierung im ISO/OSI-Modell mit diversen Protokollen und speziellen Dienstleistungen bis hin zu Kommunikationsmechanismen auf abstrakter Ebene. Dazu kommen -in den einzelnen *Knoten* des Web, bzw. Semantic Web - Aspekte aus dem Bereich Datenbanken und Informationssysteme: Modellierung sowie Konzepte und Sprachen für Metadatendefinition, Anfragen an Daten und Metadaten, sowie Datenmanipulation. Der Schritt vom *Web-Knoten* zum *Web* kombiniert die genannten Bereiche, wobei Aspekte verteilter Datenbanken, Interoperabilität, Datenintegration, Sicherheit usw. hinzukommen. Mit der Hinzunahme *aktiver* bzw. *reaktiver* Funktionalität im Web, durch *Web Services* als auch Kommunikation zwischen Knoten (z.B. zur Propagierung von Updates) kommen Aspekte aus den Bereichen aktiver Datenbanken, insbesondere aktiver Regeln, weitere Sicherheitsaspekte, Vertrauen *(Trust)*, Verfahrensweisen (*Policies*), Spezifikation verteilter, dynamischer Prozesse etc. hinzu. Der weitere Schritt zum *Semantic Web* betrifft viele der obigen Aspekte (wobei manches dabei nicht unbedingt schwieriger, sondern durch die Nutzung von Semantik auch einfacher wird – z.B. Datenintegration):

Zum einen muss die Funktionalität der einzelnen beteiligten Knoten *lokal* semantik-tauglich erweitert werden. Hier spielen Ontologien und geeignete Modelle und Sprachen zur Wissensrepräsentation (sowohl extensional als auch intensional) mit zugrundeliegenden Formalismen/Logiken und Schlussmechanismen, sowie Anfragesprachen (Daten sowie auch Metadaten) eine Rolle. Zum zweiten muss diese Funktionalität innerhalb des *Semantic Web* integriert und "aktiviert" werden. Dies erfordert Modelle, die das Wissen mehrerer Knoten integrieren und damit die Grundlage für Anfrage "an das Semantic Web" bilden. Weiterhin müssen diese Modelle Änderungen sowohl lokal als auch global unter Berücksichtigung der Kommunikationsmechanismen ermöglichen. Um solche Änderungen operational zu realisieren sind entsprechende Sprachen zur Spezifikation und Umsetzung von Verhalten (einschließlich des Änderns von Verhaltensregeln) notwendig. Damit zeigt sich schlussendlich auch, dass "Anfragen" an das Semantic Web nicht nur einfache Anfragen sind, sondern ebenfalls von diesem Verhalten Gebrauch machen.

[1] http://rewerse.net

Um diese Anforderungen zu erfüllen muß nicht das Rad neu erfunden werden, sondern ein erfolgversprechender Weg ist, Erfahrungen und Konzepte unter anderem aus den Bereichen *Logik*, *Formale Methoden*, *Datenbanken*, *Kommunikation*, *Programmiersprachen* und *Softwareengineering* zu analysieren, weiterzuentwickeln, und zu kombinieren.

Da das Semantic Web keinerlei zentrale Struktur (weder topologisch, noch thematisch) besitzt, sondern einem "lebenden Organismus" bestehend aus sich autonom entwickelnden, kommunizierenden Knoten entspricht, ist insbesondere darauf zu achten, dass die entwickelten Konzepte modular sind, und *Konzepte* und die tatsächlichen *Sprachen* unabhängig, und dennoch kompatibel gestaltet werden um z.B. den Spezifika unterschiedlicher Anwendungsgebiete Rechnung zu tragen. Ein wichtiges Konzept, das in REWERSE schwerpunktmäßig verfolgt wird sind hierbei *deklarative*, insbesondere *regelbasierte* Sprachen.

Als Basis bietet XML und die damit zusammenhängenden Technologien ein inzwischen etabliertes Framework. Die zu entwickelnden Sprachen verwenden XML als Basis, d.h. arbeiten auf XML oder RDF (das häufig in XML-Repräsentation dargestellt wird) und besitzen auch selber eine XML-Syntax, die den Austausch von Daten, Wissen und Regeln ermöglicht. Die entwickelten Konzepte werden sowohl in eingeschränkten Demonstrator-Szenarios, als auch im praktischen Einsatz in ausgewählten Anwendungen in den Bereichen personalisierter Portale und Lernumgebungen [HN04] und Bioinformatik evaluiert.

Im folgenden wird zuerst kurz auf das zugrundeliegende Modell des Semantic Web eingegangen. In Abschnitt 3 werden die statischen Aspekte, d.h., Datenmodelle und Anfragesprachen, betrachtet bevor Abschnitt 4 die dynamischen Aspekte behandelt. Abschnitt 5 faßt die wichtigsten Punkte noch einmal zusammen und präsentiert einen Sprachvorschlag.

2 Architekturmodell

Wie oben geschrieben, wird das *Semantic Web* als ein System sich autonom entwickelnder, kommunizierender Knoten aufgefaßt:

- jeder Knoten verfügt über einen *lokalen* Zustand, gegeben durch extensionale Daten (Fakten), Metadaten (Schema, Ontologieinformationen), intensionale Daten (Regeln; *derivation rules*), sowie eine Verhaltensbasis (die in Abschnitt 4 betrachtet wird). Alle diese Komponenten können prinzipiell lokal autonom geändert werden.
- jeder Knoten kann Anfragen an andere Knoten stellen (die diese direkt beantworten, oder weitergeben) sowie Nachrichten von anderen Knoten empfangen (*peer-to-peer-Kommunikation*). Mit sochen Nachrichten können neben Antworten auf Anfragen auch Updates anderen Knoten mitgeteilt, bzw. auch ggf. der Zustand anderer Knoten geändert werden (geeignete Berechtigungen vorausgesetzt).
- es gibt Knoten die im wesentlichen als Datenquellen dienen (z.B. Fahrpläne oder Vorlesungsverzeichnisse), und Knoten, die im wesentlichen Dienstleistungen als "Infrastruktur" erbringen (Broker, pub/sub-Dienste, Continuous Query-Systeme etc.), sowie beliebige Zwischenformen (z.B. Reisebüro mit eigenen Daten sowie Broker-Funktionalität).

Die sich daraus für Anfragesprachen, Updates und *ECA-Regeln* ergebenden Konsequenzen sind in [MAB04] beschrieben.

3 Statisches Modell und Anfragen an Zustände

In dem betrachteten Modell des *Semantic Web* verfügt jeder Knoten über eine *lokale* Sicht, die seinen eigenen Zustand (einschließlich Metadaten, Ableitungs- und Verhaltensregeln sowie

der bisher erhaltenen Nachrichten, soweit gespeichert), sowie alle Daten, die ihm per Anfrage zugänglich sind, umfaßt.

Die lokale Sicht ist je nach eigenem Datenmodell z.B. durch eine Datalog-Datenbasis (mit diversen Interpretationsmöglichkeiten von Negation), durch eine F-Logic-Datenbank [KLW95] (das sich Klassenhierarchie, nichtmonotone Vererbung, einen flexiblen Schemabegriff, sowie Anfragen an Metadaten als Sprache für das Semantic Web anbietet), durch eine relationale oder XML-Datenbank, oder auch direkt als RDF/OWL-Daten [RDF00a, RDF00b, OWL04] gegeben, wobei die Datenmodelle von F-Logic und OWL bereits interne Reasoning-Mechanismen umfassen (bei OWL drei nach Ausdruckskraft und Komplexität/Entscheidbarkeit unterschiedliche Versionen "Lite", "DL", entsprechend *Description Logics*, und "Full").

[BM05] klassifiziert die verschiedenen Arten deduktiver Sprachen bzw. Schlussregeln, die hierfür benötigt werden, zum einen als Sichten (*konstruktive Regeln*), zum anderen zur Definition von Integritätsbedingungen und Wissen (*normative* und *deskriptive Regeln*). Konstruktive Regeln sind neben den bekannten Prolog/Datalog-Regeln auch im weiteren Sinne die SFW- bzw. FLWR-Konstrukte von SQL bzw. XQuery, sowie XSLT-Transformationen, die ja ebenfalls eine Sicht auf einen Datenbestand definieren. Normative Regeln können ebenfalls explizit als *Denials* gegeben sein, oder implizit z.B. in Form von DTDs, wobei hier der Üebergang zu *deskriptiven Regeln*, z.B. Ontologien, die auch wiederum unmittelbar als *konstruktive Regeln* zur Ableitung von Wissen genutzt werden können, fließend ist und die Einordnung im Detail auf die Intention und Verwendung durch den Benutzer ankommt.

Web-Fähigkeit der Knoten. Ein Knoten ist "passiv" Semantic-Web-fähig, wenn er nach aussen hin neben dem Zugriff auf Daten auch den Zugriff auf Metadaten und ihre Semantik in Form einer *Ontologie* ermöglicht. Dies geschieht derzeit üblicherweise per OWL, wobei die verwendete, anwendungsspezifische, Ontologie mit den Benutzern oder "Partnern" vereinbart sein muss.

Integrationsabbildungen. Dies ist in gewisser Weise ähnlich dem in verteilten bzw. föderierten Datenbanken [Con97] verfolgten Ansatz, ein gemeinsames Schema (hier: gemeinsame Ontologie) zu definieren. Eine Ontologie enthält jedoch neben dem reinen Schema weitere Metainformationen auf konzeptueller Ebene, z.B. über Beziehungen, abgeleitete Beziehungen etc. Aber, auch hier dient die gemeinsame Ontologie nur als *externes* Schema zwischen den beteiligten Knoten. Intern kann jeder Knoten unterschiedlich realisiert sein. In der Regel werden grosse Datenquellen selber keine RDF/RDFS/OWL-Daten enthalten sondern oft nicht einmal XML, sondern klassische relationale Datenbanken sein. Dieses lokale, logische Schema wird dann auf das globale (OWL-)Schema abgebildet. Zur Beschreibung von Abbildungen zwischen den lokalen und dem globalen (bzw. dem vereinbarten) Schema finden prinzipiell die in [Len02, Len03] diskutierten LAV/GAV-Konzepte ("global/local as view") bzw. GLAV (für P2P-Kommunikation) Anwendung.

"Globale" Anfragen und Ziehen von Schlüssen. Im Gegensatz zu der Situation bei föderierten Datenbanken, wo alle Teilnehmer explizit bekannt sind, können im Web grundsätzlich Teilnehmer "auftauchen" und "verschwinden". Es existiert –auch zu einer vereinbarten Ontologie– in der Regel keine zentrale Instanz, die alle Teilnehmer aufzählt (vgl. Hotels bei einem Reisebuchungs-Szenario). Neue Teilnehmer treten auf, indem sie "Informationen ins Web stellen" und versuchen, sich möglichst prominent referenzieren zu lassen. Im derzeitigen Web, und auch im *Semantic Web*, wird eine gewisse Organisation durch *Portale*, die auf Basis von für den Benutzer nicht sichtbarem Datenaustausch entlang explizit gespeicherter Beziehungen zwischen aktiven Knoten (entsprechend materialisierten Views bei "push"-Kommunikation, oder Unteranfragen bei "pull"-Kommunikation) die Informationen mehrerer Informationsquellen integrieren. Im derzeitigen Web geschieht dies oft noch durch individuelle Abbildungen der ein-

zelnen externen Schemata der Informationsquellen auf das interne Schema des Portals; für neue "Partner" muss somit jedes Mal eine Abbildung definiert werden. Ziel des *Semantic Web* ist, nur eine oder wenige Ontologien zu dem Themenbereich eines solchen Portals zu haben, die als externe Schemata der Informationsquellen zur Verfügung stehen. Tritt hierbei ein neuer Partner ein, muss er sich nur registrieren lassen. Ein Portal kann damit als Schnittstelle zu einer sich dynamisch umkonfigurierenden (sehr locker) föderierten Web-Fragment gesehen werden.

Falls innerhalb des betrachteten Web-Fragmentes unterschiedliche Ontologien verwendet werden, muss weiterhin eine Integrationsabbildung zwischen diesen verwendet (und zur praktischen Umsetzung auch in einem Vermittlungsservice implementiert) werden.

In dieser Situation können alle Anfragen also immer nur "nach bestem Wissen" von einem *Portal* beantwortet werden. Ein Portal besitzt damit ein über sein eigenes Wissen hinausgehendes *Modell*, das jedoch in mehreren Punkten von dem üblichen Modellbegriff abweicht:

- Konsistenz: die dem Portal verfügbaren Informationen können inkonsistent sein. Hier müssen entsprechende Mechanismen sowohl theoretisch als auch pragmatisch untersucht und angewendet werden.
- Unvollständigkeit: das Nicht-Bekanntsein einer Information bedeutet nicht, dass die Information in der Realwelt nicht zutrifft. Negative Schlüsse sind mit äußerster Vorsicht zu ziehen, und in der Regel als "es ist nicht bekannt, ob $p(x)$" zu interpretieren (was für den Benutzer dennoch häufig $\neg p(x)$ bedeutet, etwa wenn er ein Hotel buchen möchte).

Daraus ergibt sich, dass je nach Anwendung unterschiedliche, und insbesondere unterschiedlich aufwändige Logiken herangezogen werden (vgl. [BM05]): Übliche Anforderungen wie *excluded middle* ($A \vee \neg A$), *non-contradiction* ($\neg(A \wedge \neg A)$, Refutation $((A \rightarrow (B \wedge \neg B)) \rightarrow \neg A)$ sind nicht mehr gültig. Stattdessen benötigt man nichtmonotone Negation und disjunktive Theorien. Die o.g. Regeln müssen entsprechend dieser Semantiken interpretiert werden.

Auf die Kombination von Ontologien und (Schluss)regeln wird in [ADG+05] detailliert eingegangen (u.a., F-Logic, SWRL und DLP).

Anfragesprachen

Um sowohl der verteilten Natur der Informationen, als auch der zusätzlichen semantischen Ebene gerecht zu werden, müssen die Anfragesprache(n) sowohl beides unterstützen, als auch ggf. voneinander abgrenzen können:

- Anfragen an lokale Daten
- Anfragen an entfernte Daten gezielt per URL+Anfrage (auf der Ontologie-Ebene des *gemeinsamen* Schemas): Anfragen dieser Art werden insbesondere zur Propagation von Updates und sonstigen *Reaktionen* bei der in Abschnitt 4 behandelten Spezifikation von (lokalem und globalem) Verhalten verwendet. Da sie kein Web-weites Reasoning benötigen, sind sie effizienter auszuwerten.
- Anfragen an verteilte Daten auf Semantic-Web-Ebene. Auf dieser Ebene findet auch die Interaktion mit dem Benutzer statt.

Einen Überblick über existierende Anfragesprache für das Web und das Semantic Web wird in [FBS+04] gegeben; Anforderungen an Anfragesprachen sowie Updates sind in [MAB04] beschrieben.

Raum und Zeit. Spezielle Anforderungen an Modellierung, Reasoning und Anfragekonstrukte stellen die Bereiche *räumlicher* und *zeitlicher Daten*, wobei mit letzterem hier zeitliche Wertebereiche (wie etwa bei Terminkalendern), nicht temporale Anfragen über die Entwicklung der Datenbank gemeint sind. Neben typischen zeitbezogenen Anwendungen wie etwa Terminplanung und Auftrags-/Rechnungsabwicklung spielen zeitliche Annotationen auch eine Rolle, wenn die Aktualität von Nachrichten (Zeitpunkt des Auftretens eines Ereignisses gegenüber dem Zeitpunkt der Benachrichtigung) berücksichtigt werden muss.

4 Evolution und Reaktivität

Die beiden Aspekte *Evolution* und *Reaktivität* sind eng miteinander verbunden: Evolution kann –als Verhalten– z.B. durch reaktive Regeln deklarativ beschrieben und implementiert werden. Einen Überblick über beides findet man in [ABB+04, AM05], sowie eine Anforderungsanalyse für das Semantic Web in [MAB04].

Einfaches reaktives Verhalten ist bereits diesseits von Evolution bei der Beantwortung von Anfragen relevant:

- Reasoning kann durch deduktive oder reaktive Regeln beschrieben werden (vgl. die Äquivalenz des intern und in der originalen Spezifikation [KLW95] trigger-basierten Mechanismus für nichtmonotone Vererbung mit Default-Logik; [MK01]).
- Kommunikation zwischen verschiedenen Knoten zur Beantwortung von Anfragen. Reaktive Regeln können hier zur Implementierung der verschiedenen Strategien (*push*, *pull*, sowie *publish-subscribe* und *continuous queries*) verwendet werden [ABB+04, Kap. 1.7]).

Da zur verteilten Beantwortung von Anfragen weitere *Policies* angewendet werden (z.B. wenn eine Datenquelle nicht antwortet, oder um Konsistenz und *Trust* zu berücksichtigen), werden bereits in diesem Fall häufig reaktive Regeln Anwendung finden (für eine Bestandsaufnahme, siehe [BSD+05]).

Verteilte Anfragebearbeitung im Web. Um Anfragen auf Semantic-Web-Ebene effizient zu beantworten muss –wenn man nicht ein allgemeines Broadcasting machen will– zuerst geklärt werden, *welche* Partnerknoten relevante Antworten liefern können. Diese Auswahl basiert auf den Metadaten, die über die einzelnen Quellen bekannt sind [Kos00, Suc02, BDK+03].

Web Services beantworten oft nicht beliebige Anfragen in einer Anfragesprache, sondern nur eine eingeschränkte Menge von Formularanfragen. Ein solcher Web Service wird im Semantic Web üblicherweise durch eine Interfacebeschreibung in (z.B. in WSDL [WSD01] oder OWL-S [OWL03]) spezifiziert. Um solche Web Services zur (Teil)beantwortung von Anfragen zu nutzen, müssen erst geeignete Anfragen ausgesucht, und deren Antworten dann kombiniert werden (*Query Rewriting* [CGLV00, Hal01]).

4.1 Updates und Evolution im ("konventionellen") Web

Soweit wurden ein sich nicht veränderndes, und nur zur Anfragebearbeitung "aktives" Web betrachtet. Im gegenwärtigen Web tritt *Evolution* im wesentlichen in den folgenden einfachen Szenarien auf:

- Lokale Änderungen aufgrund externer Updates, oder lokale Evolution von Knoten als Reaktion auf einfache Ereignisse (etwa wie bei SQL-Triggern) auf Basis lokalen Wissens.
- Einfache Kommunikation (z.B. zur Propagation von Buchungen als Updates) zwischen Knoten durch *Nachrichten* und *reaktive Regeln*.

- Lokales Verhalten von *Web Services* als *Black Box*; die Ergebnisse werden ggf. durch Nachrichten mitgeteilt und auf die beiden vorhergehenden Fälle umgesetzt.
- Seltener: lokale Reaktionen auf lokale Ereignisse, die auch die *Sicht* des Knotens auf seine Umgebung (um nicht einfach von dem nicht existierenden "globalen" Modell zu sprechen) mit einbeziehen. Hierbei hat man erste Ansätze von *koordinierter* Evolution und Verhalten, womit z.B. weitergehende Konsistenz- und Plausibilitätsbedingungen berücksichtigt werden können.

4.2 Updates im Semantic Web

Die Sicht des *Semantic Web* als "lebender Organismus" bestehend aus sich autonom entwickelnden, kommunizierenden Knoten, der insgesamt ein "globales" Verhalten zeigt, führt zu einer Sichtweise als *kooperativer Evolution* der beteiligten Knoten. Neben lokalen Updates an einer Datenbasis müssen dazu verteilte Updates betrachtet werden:

- Updates an lokalen Daten müssen ggf. anderen Knoten, die diese Daten verwenden, zeitnah mitgeteilt werden. Hierzu muss festgestellt werden, wem welche Änderungen mitgeteilt werden müssen, womit prinzipiell dieselben Fragen wie bei *materialisierten Views* auftreten.
- *Intensionale Updates*, die sich auf abgeleitete, evtl. auch entfernte Daten beziehen. Dieser Fall entspricht einem *View Update*, das entsprechend weitergegeben werden muss.

In allen Fällen müssen die Integrations-Abbildungen vom lokalen auf das globale Schema hierzu "rückwärts" durchlaufen werden – d.h., egal ob diese auf LAV oder GAV basieren, endet man am Ende bei dem P2P-typischen GLAV.

Um das Szenario noch zu komplettieren, können auch unvollständig spezifizierte Updates betrachtet werden: Fakten können entweder durch Eingaben bekannt oder auf Ontologie-Level abgeleitet werden, die zu dem "bekannten" Zustand inkonsistent sind, etwa in Gegenwart unvollständiger Information, insbesondere aufgrund der zu erwartenden unvollständigen Nachrichtenübermittlung. Hierbei ist das Update (je nach Zuverlässigkeit) umzusetzen, und –evtl unter Einbeziehung weiterer Nachfragen an andere Knoten– wieder ein konsistenter Zustand herzustellen. Prinzipiell kann man dabei bis hin zu auf *Fuzzy Logics* oder *epistemischen Logiken* [KLM90] basierenden Ansätzen gehen.

Weiterhin kann man auch Updates der *deduktiven Regelbasis* (z.B. Evolving Logic Programs) oder –dem nächsten Abschnitt vorgreifend– Updates der Verhaltensbasis in Betracht ziehen (siehe [ABB+04]).

Update-Sprachen und -Konzepte. So wie Updates in Datenbanken auf den entsprechenden Anfragesprachen basieren, werden Update-Sprachen für das *(Semantic) Web* auf den entsprechenden Web-Anfragesprachen basieren. Verwendet man XML als internes Datenmodell, so ist dies z.B. XQuery+Updates; entsprechend RDQL+Updates für RDF-Daten, und entsprechende Konstrukte für Änderungen der Ontologien (womit man dem reinen Update auch sofort wieder entsprechendes Reasoning beiseitestellen muss, um die Konsistenz zu sichern).

4.3 Evolution und Verhaltensspezifikation

Die "globale", koordinierte Evolution im Semantic Web basiert auf geeignetem *Verhalten* der beteiligten Knoten. Hierbei bleibt die Reaktivität nicht nur auf Reaktionen auf einfache interne Ereignisse, Benutzerinteraktionen oder Nachrichten beschränkt, sondern umfasst auch komplexere Verhaltensweisen, etwa um anwendungsspezifisches Verhalten, z.B. *Business Rules* zu realisieren, oder auch *Policies* im Umgang mit eingehenden Informationen anzuwenden.

4.3.1 ECA-Regeln

Für eine zugleich deklarative und ausführbare Spezifikation von Verhalten bieten sich *reaktive Regeln* nach dem aus dem Bereich *aktiver Datenbanken* bekannten *Event-Condition-Action* (ECA)-Paradigma an [ABB+04, Kap. 4], [Pat99]. Kommunikation und reaktives Verhalten kann damit beides in einem auf der Basis von *Ereignissen (Events)* beschrieben werden: die beteiligten Knoten erkennen Ereignisse, überprüfen daraufhin eine *Bedingung (Condition)* und führen ggf. eine *Aktion (Action)* aus.

Eine ECA-Regel besteht entsprechend aus drei Teilen, die jeweils wiederum in einer Sub-Sprache für Ereignisse, Bedingungen (Anfragen), und Aktionen gegeben sind. Das Spektrum reicht dabei von einfachen reaktiven EA-Regeln bis hin zu komplexen Regeln deren Ausführung komplexe Ereigniserkennung, Anfragen, und Ausführung von Transaktionen beinhaltet.

ECA-Regeln können auf verschiedenen Abstraktionsebenen definiert (und implementiert) sein (siehe [ABB+05, Kap. 2]). Die tatsächliche Umsetzung der elementaren Ereigniserkennung geschieht auf der Datenbank-Ebene: Während für relationale Daten hier nur die bekannten SQL-Trigger zur Verfügung stehen, kann man bei XML-Daten entweder auf DOM-Ebene oder auf XPath/XQuery-Ebene ansetzen. Trigger auf der Ebene des Semantic Web spezifizieren ihre Ereignisse in der Terminologie des RDF bzw. OWL-Datenmodells. Intern müssen sie in den meisten Fällen auf XML- oder SQL-Trigger umgesetzt werden (wobei auch hier wieder berücksichtigt werden muss, dass die RDF/OWL-Schicht ein View ist).

Ereignisse. Ein (atomares) Ereignis ist allgemein jedes feststellbare Ereignis im Web, d.h., lokale Systemereignisse, ankommende Nachrichten (wobei sowohl der Nachrichteneingang selber ein Ereignis ist, als auch die Nachricht möglicherweise über das Eintreten eines anderen Ereignisses informiert), Transaktionsereignisse (Commit etc.), Updates von Daten, oder beliebige anwendungsspezifische Ereignisse. Neben diesen expliziten Ereignissen sollte es in Semantic-Web-Anwendungen auch möglich sein, Ereignisse abstrakt auf Ontologie-Level zu beschreiben; in diesem Fall muss ihre Erkennung auf geeignete Anfragen abgebildet (d.h. eine Zuordnung des Ereignisses zu einem oder mehreren elementaren Ereignissen in einem oder mehreren Knoten; ggf. muß auch eine Überwachung durch Continuous Queries stattfinden) und durchgeführt werden.

Reaktive Regeln beruhen nicht nur auf atomare Ereignissen, sondern verwenden häufig *zusammengesetzte Ereignisse (Composite Events)*, z.B. "wenn E_1 eintritt, und dann E_2 und E_3, aber nicht E_4 innerhalb höchstens 10 Minuten, dann führe A aus". Zusammengesetzte Events werden üblicherweise mit Hilfe von *Event Algebren* [ABB+04, Kap. 2.7], [CKAK94] beschrieben. Die Erkennung von (zusammengesetzten) Ereignissen kann Parameter einzelner Ereignisse an Variablen binden und zurückgeben.

Bedingungen. Bedingungen sind üblicherweise Anfragen. Sie können Parameter enthalten, die durch die Ereignisse definiert wurden, und auch neue Variablen binden.

Aktionen. Aktionen sind häufig einfache Updates oder das Senden on Nachrichten, können aber auch durch kompliziertere Spezifikationen gegeben sein. Sie können mit (aus der Ereigniserkennung und der Auswertung der Bedingung gebundenen) Variablen parameterisiert sein. Der Aktions-Teil kann z.B. als Ausdruck einer Prozessalgebra (CCS/CSP) oder Term über den Aktionen einer Aktionslogik gegeben sein.

Globale ECA-Regeln. Die oben beschriebenen Konzepte gehen implizit von einer *lokalen* Auswertung der ECA-Regeln in einem Knoten aus. Weitergehend können auch *globale* Regeln definiert werden, die nicht mehr ausschliesslich lokal ausgewertet werden können:

- Ereignisse, die explizit verschiedenen Knoten zugeordnet sind,

- *intensionale*, abstrakt definierte Ereignisse für die die genaue Zuordnung zu einem Knoten nicht angegeben ist, und die auch oft nicht auf ein einzelnes Update abbildbar sind, sondern ggf. erst abgeleitet werden müssen. Zur tatsächlichen Erkennung des Ereignisses bleibt hier wohl oft nur der Umweg über *Continuous Queries* übrig.

Globale ECA-Regeln können z.B. als Dienst von Portalen ausgewertet werden.

Auswertung von ECA-Regeln. Der theoretisch interessanteste Teil ist hier die Erkennung von zusammengesetzten Ereignissen. Ein naiver Ansatz wäre, alle Ereignisse zu speichern, und zusammengesetzte Ereignisse als Anfragen an diese *Event Base* umzuschreiben. Dies ist aus Effizienzgründen nicht praktikabel (nach jedem atomaren Ereignis müssten alle Anfragen komplett ausgewertet werden). Stattdessen wird die Ereigniserkennung *inkrementell* durchgeführt: jeder zu erkennende zusammengesetzte Ereignisausdruck wird auf einen Automaten-Schema abgebildet. Wird ein atomares Ereignis festgestellt, wird entsprechend ein Automaten-Schema instanziiert (mit den aktuellen Parametern) und alle bereits "laufenden" Automaten ggf. weitergeschaltet (äquivalente Formalismen existieren auf Basis von Graphen sowie temporallogischen Formeln).

Weiterhin ist es wünschenswert, innerhalb zusammengesetzter Ereignisse auch bereits Werte und Bedingungen aus dem gegenwärtigen Zustand auszuwerten. Bei Verwendung einer inkrementellen Ereigniserkennung sind diese Tests kein Problem.

Neben ECA-Regeln existieren weitere in Frage kommende Formalismen, z.B. Prozessalgebren (CCS/CSP) [Mil83, Hoa85] oder die –ebenfalls regelbasierte– *Transaction Logic* [BK94], die gleichzeitig zur Planung und zur Ausführung von Aktionen verwendet werden kann.

Da die Semantik von Transaction Logic –im Gegensatz zu ECA-Regeln– direkt auf den Begriffen *Struktur* und *Formeln* und einer *Modelltheorie* mit Tarski-Semantik [Kei78] basiert, eignet sich dieses Framework auch direkt um (Korrektheits)Aussagen *über* Regeln und Zustände zu machen. Andere logikbasierte Ansätze verwenden "klassische" modale Temporallogik und Kripke-Strukturen [Eme90] zur Verifikation.

4.3.2 ECA-Sprachentwurf

Um die beschriebenen Anforderungen zu erfüllen, müssen mehrere (immer noch generische) Sprachen und Teilsprachen definiert werden: Zumindest wird eine umgebende Sprache für ECA-Regeln benötigt, die eine Sprache für zusammengesetzte Ereignisse (die wiederum mit atomaren Ereignissen parametrisiert ist), eine Sprache für Bedingungen und eine Sprache für Aktionen einbettet (und eine Variablenübergabe mit diesen ermöglicht). In Anbetracht der Tatsache, dass es jeweils viele verschiedene Vorschläge (und wie oben gesehen auch Abstraktionsebenen) für Event-Algebren, Anfragesprachen und auch für Aktionsspezifikationssprachen gibt, sollte man jeweils nicht nur die Einbettung einer einzelnen, gegebenen Sprache vorsehen, sondern einen modularen Ansatz wählen, der es ermöglicht –unter gewissen Bedingungen– beliebige Sprachen einzubetten.

5 Entwicklung Regelbasierter Sprachen für das Semantic Web

In [BM05] werden Anforderungen an die zu entwickelnden logikbasierten Mechanismen und Sprachen für das Semantic Web definiert, von denen einige hier bereits genannt wurden und deren Quintessenz zum Schluss noch einmal zusammengefasst wird:

- formales, logik-basiertes (Daten)Modell und Modelltheorie mit modular aufgebauter Reasoning-Funktionalität (Negation, Inkonsistenz, Unvollständigkeit),

- auf dem Datenmodell aufbauende Anfragesprache(n) (sowohl Anfragen an den Zustand, als auch im weiteren Sinne die Sprache zur Modellierung von Ereignissen) mit deklarativer Semantik,
- Sprache zur Spezifikationen von Aktionen muss zumindest eine operationale Semantik besitzen; wünschenswert wäre auch hier eine modelltheoretische Semantik,
- alle Sprachen müssen getypt sein,
- Modularität in Syntax und Darstellung, Semantik und Auswertung/Ausführung der Sprachen und Sprachkonzepte: Interoperabilität und Komponierbarkeit (*Composability*).

Kohärenz in Syntax und Darstellung wird u.a. dadurch erreicht, dass alle Sprachen eine XML-Repräsentation (vgl. XSLT, XML Schema) besitzen. Interoperabilität und Komponierbarkeit erfordert wohldefinierte Schnittstellen, zu denen wiederum die durchgehende Typisierung beitraägt. Weiterhin wird dies durch die Definition einer gemeinsamen, erweiterbaren Ontologie unterstützt, womit die Sprachen auch selber wieder zum Teil des Semantic Web werden (und die einzelnen Regeln zu Objekten im Semantic Web).

Sprachvorschlag (Entwurf). Das untenstehende Beispiel zeigt einen groben Sprachvorschlag für eine Markup-Language "ECA-ML" [ABB+05] für ECA-Regeln, der deutlich an XSLT angelehnt ist. Jede Regel (und auch andere Konstrukte) kann Variablen definieren und mit Werten initialisieren. Eine Regel besteht aus je einem eca:event, eca:condition (optional) und eca:action-Element. Jedes dieser Elemente besitzt ein eca:name-Attribut, sowie ein href-Attribut, das auf eine URI verweist. Ist diese URI ein Directory, so könnte dort z.B. eine Beschreibung der jeweiligen Sprache als XML-Schema oder OWL-Instanz liegen, oder auch im Fall einer Event-Sprache ein Java-Klasse oder ein Web Service, die den Detektionsalgorithmus implementiert (oder eine einfachere Menge von ECA-Regeln, die den Automaten beschreiben) und vom ECA-Interpreter bei Bedarf geladen oder aufgerufen werden kann.

Der Event-Teil der Regel ist als Folge spezifiziert, deren erstes atomares Ereignis das Löschen eines Elements der Antwortmenge von $xpath\text{-}expr_1$ der lokalen Datenbasis ist (ähnlich dem XSLT-Mechanismus). Beim Eintreten eines solchen Events wird die Variable var_1 an das gelöschte Element e gebunden, und die Variable var_2 an $e/xpath\text{-}expr_3$. Wenn danach das im Beispiel nicht näher spezifizierte Teilevent schrittweise detektiert wird, ist am Ende der Ereignis-Teil der Regel erfüllt. Als nächstes wird der eca:condition-Teil ausgewertet (der hier nur einen XPath-Ausdruck testet) – hier könnten ggf. vorher gebundene Variablen verwendet und weitere Variablen gebunden werden. Ist die Beduingung erfüllt, wird der eca:action-Teil ausgeführt (ggf. werden hier die vorher gebundenen Variablen verwendet).

```
<eca:rule>
  <eca:variable name="...">xpath</variable>
  <eca:event name= "eca-events-basic" href="uri" >
    <evt:seq>
      <evt:atomic>
        <delete-of select="xpath-expr1" >
          <variable name="var1" select="." />
          <variable name="var2" select="xpath-expr3" />
        </change-of>
      </evt:atomic>
      <evt:...>
        Spezifikation eines weiteren (zusammengesetzten) Events
      </evt:...>
```

```
    </evt:seq>
  </eca:event>
  <eca:condition language="XPath" href="uri">
    xpath-expr
  </eca:condition>
  <eca:action language="XQuery+Updates" href="uri">
    update xpath-expr4
    set xpath-expr5 := value
  </eca:action>
</eca:rule>
```

Acknowledgement. This research has been co-funded by the European Commission and by the Swiss Federal Office for Education and Science within the 6th Framework Programme project REWERSE number 506779 (cf. `http://rewerse.net`).

Literatur

Die in der folgenden Literaturliste aufgeführten Referenzen bieten einen umfassenden Überblick über den State of the Art in den behandelten Bereichen.

[ABB+04] José Júlio Alferes, James Bailey, Mikael Berndtsson, Franois Bry, Jens Dietrich, Alexander Kozlenkov, Wolfgang May, Paula-Lavinia Pătrânjan, Alexandre Pinto, Michael Schröder, and Gerd Wagner. State-of-the-art on evolution and reactivity. Technical Report I5-D1, REWERSE EU FP6 NoE, 2004. Available at `http://www.rewerse.net`.

[ABB+05] José Júlio Alferes, Mikael Berndtsson, François Bry, Michael Eckert, Wolfgang May, Paula Lavinia Pătrânjan, and Michael Schröder. Use cases in evolution and reactivity. Technical Report I5-D2, REWERSE EU FP6 NoE, 2005. Available at `http://www.rewerse.net`.

[ADG+05] Grigoris Antoniou, Carlos Viegas Damsio, Benjamin Grosof, Ian Horrocks, Michael Kifer, Jan Maluszynski, and Peter F. Patel-Schneider. Combining Rules and Ontologies. A survey. Technical Report I3-D3, REWERSE EU FP6 NoE, 2005. Available at `http://www.rewerse.net`.

[AM05] José Júlio Alferes and Wolfgang May. Course: Evolution and reactivity for the web. In *REWERSE Summer School*, 2005. to appear with Springer LNCS.

[BDK+03] Ingo Brunkhorst, Hadhami Dhraief, Alfons Kemper, Wolfgang Nejdl, and Christian Wiesner. Distributed queries and query optimization in schema-based p2p-systems. In *Databases, Information Systems, and Peer-to-Peer Computing (DBISP2P)*, pages 184–199, 2003.

[BK94] A. J. Bonner and M. Kifer. An overview of transaction logic. *Theoretical Computer Science*, 133(2):205–265, 1994.

[BM05] François Bry and Massimo Marchiori. Ten Theses on Logic Languages for the Semantic Web. In *Proceedings of W3C Workshop on Rule Languages for Interoperability*, 2005.

[BSD+05] Piero A. Bonatti, Nahid Shahmehri, Claudiu Duma, Daniel Olmedilla, Wolfgang Nejdl, Matteo Baldoni, Cristina Baroglio, Alberto Martelli, Viviana Patti, Paolo Coraggio, Grigoris Antoniou, Joachim Peer, and Norbert E. Fuchs. Rule-based policy

specification - state of the art and future work. Technical Report I3-D3, REWERSE EU FP6 NoE, 2005. Available at `http://www.rewerse.net`.

[CGLV00] Diego Calvanese, Guiseppe De Giacomo, Maurizio Lenzerini, and Moshe Y. Vardi. What is query rewriting? In *Knowledge Representation meets Databases (KRDB 2000)*, pages 17–27. CEUR Electronic Workshop Proceedings, `http://ceur-ws.org/Vol-29/`, 2000.

[CKAK94] S. Chakravarthy, V. Krishnaprasad, E. Anwar, and S.-K. Kim. Composite events for active databases: Semantics, contexts and detection. In *Proceedings of the 20th VLDB*, pages 606–617, 1994.

[Con97] Stefan Conrad. *Föderierte Datenbanksysteme: Konzepte der Datenintegration*. Springer, 1997.

[Eme90] E. A. Emerson. Temporal and modal logic. volume B: Formal Models and Semantics, chapter 16, pages 995–1073. Elsevier, 1990.

[FBS+04] Tim Furche, François Bry, Sebastian Schaffert, Renzo Orsini, Ian Horrocks, Michael Krauss, and Oliver Bolzer. Survey over Existing Query and Transformation Languages. Technical Report I4-D1, REWERSE EU FP6 NoE, 2004. Available at `http://www.rewerse.net`.

[Hal01] Alon Y. Halevy. Answering queries using views: A survey. *VLDB Journal*, 10(4):270–294, 2001.

[HN04] Nicola Henze and Wolfgang Nejdl. A Logical Characterization of Adaptive Educational Hypermedia. *New Review of Hypertext and Hypermedia (NRHM)*, 10(1):77–113, 2004. Available at `http://www.kbs.uni-hannover.de/Stamm/publikationen/publikationen.html`.

[Hoa85] C.A.R. Hoare. *Communicating Sequential Processes*. Prentice Hall, 1985.

[Kei78] J. Keisler. Fundamentals of model theory. In *Handbook of Mathematical Logic*, pages 47–103. North Holland, 1978.

[KLM90] S. Kraus, D. Lehmann, and M. Magidor. Nonmonotonic reasoning, preferential models and cumulative logics. *Artificial Intelligence*, 44:167–207, 1990.

[KLW95] Michael Kifer, Georg Lausen, and James Wu. Logical foundations of object-oriented and frame-based languages. *Journal of the ACM*, 42(4):741–843, 1995.

[Kos00] Donald Kossmann. The state of the art in distributed query processing. *ACM Comput. Surv.*, 32(4):422–469, 2000.

[Len02] Maurizio Lenzerini. Data integration: a theoretical perspective. pages 233–246, 2002.

[Len03] Maurizio Lenzerini. Tutorial on information integration. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2003.

[MAB04] Wolfgang May, José Júlio Alferes, and François Bry. Towards generic query, update, and event languages for the Semantic Web. In *Principles and Practice of Semantic Web Reasoning (PPSWR)*, number 3208, pages 19–33. Springer, 2004.

[Mil83] R. Milner. Calculi for synchrony and asynchrony. *Theoretical Computer Science*, pages 267–310, 1983.

[MK01] Wolfgang May and Paul-Th. Kandzia. Nonmonotonic inheritance in object-oriented deductive database languages. *Journal of Logic and Computation*, 11(4), July 2001.

[OWL03] OWL-S: Web Service Ontology. `http://www.daml.org/services/owl-s/`, 2003.

[OWL04] OWL Web Ontology Language. `http://www.w3.org/TR/owl-features/`, 2004.

[Pat99] N. W. Paton, editor. *Active Rules in Database Systems*. Monographs in Computer Science. Springer, 1999. ISBN 0-387-98529-8.

[RDF00a] Resource Description Framework (RDF). `http://www.w3.org/RDF`, 2000.

[RDF00b] Resource Description Framework (RDF) Schema specification. `http://www.w3.org/TR/rdf-schema/`, 2000.

[Suc02] Dan Suciu. Distributed query evaluation on semistructured data. *ACM Transactions on Database Systems*, 27(1):1–62, 2002.

[WSD01] Web Services Description Language (WSDL). `http://www.w3.org/TR/wsdl`, 2001.

Reguläre Beiträge

Visuelle Exploration und semantikbasierte Fusion multivariater Datenbestände

Stefan Audersch, Guntram Flach, Tom Klipps
Zentrum für Graphische Datenverarbeitung e.V., Rostock
Joachim-Jungius-Str. 11, 18059 Rostock
{stefan.audersch, guntram.flach, tom.klipps}@rostock.zgdv.de

Abstract: In verschiedenen Forschungsbereichen spielen Techniken der semantischen Datenintegration eine besondere Rolle. Es existiert ein Bedarf an Lösungen, die über eine Fusion von Daten hinaus eine Nutzung verteilt entwickelter Analysemethoden für Daten ermöglichen.
In dieser Arbeit wird ein Ansatz entwickelt, der basierend auf Techniken der Datenexploration und semantikbasierten Fusion eine Nutzung von Analysemethoden wie DataMining- und Visualisierungstechniken in verteilten Umgebungen erlaubt. Unter Einsatz von Ontologien zur semantischen Beschreibung verteilter Quellen wird es ermöglicht, die Daten und Analysemethoden aus diesen Quellen zu fusionieren.
Es wird eine Architektur für diese Aufgabe vorgestellt. Kern der Architektur ist die Gatewaykomponente, die es dem Analysten erlaubt, Daten und Analysemethoden in einer verteilten Umgebung zu nutzen. Ein Prototyp implementiert die vorgestellten Komponenten.

1 Einleitung

Die automatische Erfassung von Daten durch kommerzielle Geräte und wissenschaftliche Instrumente führen zu immer größeren Mengen von immer komplexeren Daten, deren manuelle Analyse die kognitiven Fähigkeiten des Analysten bei weitem überschreiten. Zur Automatisierung dieser Analysen kommen Techniken aus dem Bereich des Knowledge Discovery in Databases (KDD) zum Einsatz, bei dem Data Mining und Visualisierung zentrale Schritte darstellen. Die visuelle Datenexploration erlaubt es dem Benutzer, einen schnellen Einblick in die Struktur der Daten zu bekommen, Schlussfolgerungen aus den Daten zu ziehen sowie direkt mit den Daten zu interagieren (Overview, Zoom and filter, details-on-demand) [AS04]. Die Qualität der Ergebnisse einer solchen Wissensgewinnung ist stark von dem Expertenwissen abhängig, mit dessen Hilfe die eingesetzten Verfahren gesteuert werden. Neben dem benötigten Wissen ist eventuell die Datengewinnung, Vorverarbeitung bzw. Aufbereitung von Daten nur in einer speziellen Laborumgebung oder unter Einsatz besonderer Mittel, Werkzeuge oder Analysemethoden möglich. Besonders auf den Gebieten der Medizin und Molekularbiologie führt die thematische und räumliche Trennung der weltweiten Forschung dazu, dass eine Vielzahl von Firmen, Gruppen und Konsortien existieren, von denen jede ihre eigene Basis an Forschungsdaten besitzt. Eine Unterstützung der Forschungsarbeit können Werkzeuge und Verfahren bieten, welche die Daten der durchgeführten Experimente mit Informationen aus komplementären Datenquellen anreichern und eine Einordnung und Bewertung der eigenen Daten im Vergleich mit Daten anderer Forschungen ermöglichen. Dabei ergibt sich die Notwendigkeit einer dynamischen Informationsfusion, die eine bedarfsgetriebene, skalierbare Kopplung und Integration von Datenbanken, Datenströmen und Datenanalysemodellen verwirklicht.
Ausgangspunkt dieses Beitrages ist ein Anwendungsszenario, das medizinische Messwerte im Rahmen einer Klinischen Studie[1] betrachtet. In diesem Szenario geht es um die Analyse multivariater Patientendaten (z.B. Nerven-, Leber- und Blutdaten) unter Einsatz von Data Mining und Visualisierungstechniken in verteilten Umgebungen. Obwohl die erhobenen Daten eine semantische Einheit bilden, werden sie aufgrund unterschiedlicher technischer und fachlicher Anforderungen in Teildatenbestände zerlegt und getrennt voneinander analysiert.

[1] In Kooperation mit der Teraklin AG (http://www.teraklin.de)

2 Problemstellung und Anforderungen

Die getrennten Datenbestände werden einzeln ausgewertet, aufbereitet und analysiert. Hierbei kommen unterschiedliche DataMining- und Visualisierungstechniken zum Einsatz, die auf die Beschaffenheit der unterschiedlichen Daten zugeschnitten sind (Abbildung 1). Der Zusammenhang zwischen den einzelnen Daten kann in dieser Phase der Wissensgewinnung nicht erfasst werden. Erst eine zentrale Anwendung, die Zugriff auf die einzelnen Datenbestände (Explorationsquellen) hat, leistet dies.

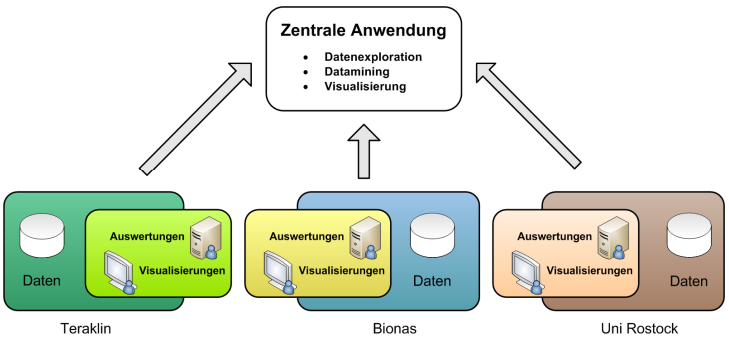


Abbildung 1: Anwendungsszenario

Eine Lösung in diesem Sinne würde es erlauben, auf der gesamten Datenbasis weiterzuarbeiten, wodurch sich beispielsweise die beiden folgenden Aufgabenstellungen lösen ließen:

- Die Visualisierung von Zusammenhängen zwischen Komazustand, Blut- und Leberwerten der Patienten ist auf Grundlage der Teraklin-Datenbasis möglich. Im Rahmen der Studie ist es aber durchaus von Interesse, Messungen an Gewebeproben (Bionas) oder Werte von Aminosäuren (Universität) in einen Zusammenhang mit Blut-, Leber- oder Komawerten zu bringen.
  Eine Kombination von Rohdaten und Ergebnissen einer Datenbasis mit Rohdaten und Ergebnissen anderer Datenbasen ist notwendig, um diese Analysen zu realisieren.
- Zwischen unterschiedlichen Datenreihen der Teraklin-Datenbasis wird ein Zusammenhang vermutet, der sich anhand der vorliegenden Daten nicht eindeutig zeigen lässt. In den Auswertungen der Bionas- oder Universitäts-Daten wird festgestellt, dass ein bestimmtes Phänomen bei einem Teil der Patienten auftritt. Denkbar ist an dieser Stelle die Durchführung der ursprünglichen Analyse der Teraklin-Daten auf dem extern motivierten Teilbereich. Es muss möglich sein, selektive Anfragen an Daten einer Datenbasis unter Ausnutzung von Inhalten anderer Datenbasen zu formulieren. Das Formulieren beliebiger Anfragen an einen virtuellen Gesamtdatenbestand wäre in diesem Fall die optimale Lösung.

Zur Lösung dieser Aufgaben besteht die Notwendigkeit, die Daten sowie die Analyseprozesse aus den verschiedenen Explorationsquellen virtuell zu fusionieren. Voraussetzung für eine intelligente Zusammenführung ist eine maschinenverständliche Semantik der Explorationsquellen. Grundlage hierfür bietet eine gemeinsame Ontologie, die unter anderem eine gemeinsame Terminologie abbildet. Die angedachten Anforderungen sollen nachstehend zusammengefasst und konkretisiert werden:

- **Datenintegration**: Es soll möglich sein, auf der gesamten Datenbasis zu arbeiten, ohne die Daten in einem initialen Schritt in eine einzige Datenbasis zu integrieren.
- **Datenexploration**: In der Datenexploration soll es möglich sein, den gesamten Datenbestand sowie auch die Ergebnisse der lokal durchgeführten Analysen, DataMining-Verfahren und Visualisierungen einzusehen.
- **DataMining und Visualisierung**: Es soll möglich sein, vorhandene DataMining-Verfahren oder Visualisierungen auf neuen (aus der Datenfusion resultierenden) Datenbeständen durchzuführen. Im Sinne einer visuellen Datenexploration soll eine Interaktion mit bestimmten Visualisierungen möglich sein.

- **Semantik**: Die semantischen Beschreibungen sollen es erlauben, verschiedene Datenquellen einfach miteinander zu verbinden und deren Heterogenität aufzulösen. Durch Nutzung der Semantik sollte das Anwenderprogramm dem Benutzer Hilfestellung (z.B. in Form eines Wizards) bei der Exploration geben.

3 Realisierungsaspekte

Die für den Lösungsansatz notwendigen Überlegungen werden im folgenden Abschnitt durch eine Auswahl verschiedener Realisierungsaspekte kurz vorgestellt.

Prozesse

Verfahren des Data Mining und der Visualisierung von Daten stellen zentrale Schritte im KDD-Prozess dar und bilden die Grundlage für die visuelle Datenexploration. Um diese Verfahren in das System zu integrieren, können diese als Prozesse aufgefasst und als Service von einer Explorationsquelle bereitgestellt werden. Ebenso lässt sich die Bereitstellung von Datentabellen als auch der Zugriff auf Analyseergebnisse als Prozess definieren. Eine Explorationsquelle (Abbildung 2) kann verschiedene Prozesse zur Verfügung stellen. Für die Integration von Prozessen ist es notwendig, die Explorationsquellen und deren Prozesse semantisch zu beschreiben. Die Beschreibungen umfassen dabei Informationen über die von der Explorationsquelle bereitgestellten Prozesse. Für den jeweiligen Prozess sind Informationen über dessen Vorbedingungen, Parameter und Ergebnisse definiert.

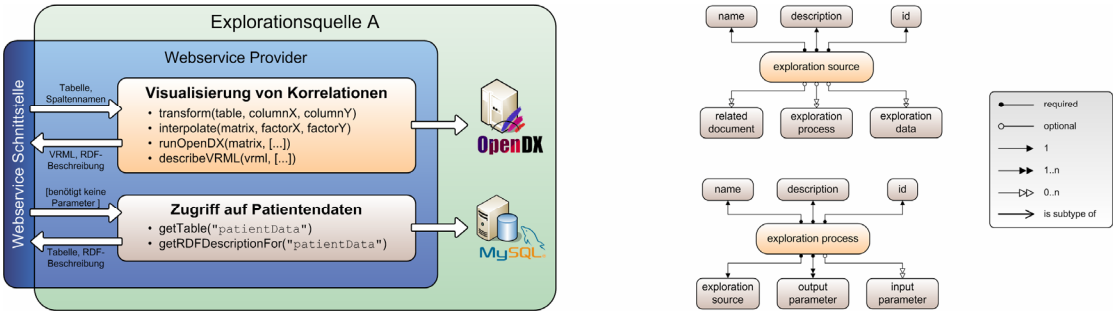


Abbildung 2: Explorationsquelle und semantische Beschreibungen

Im Rahmen der Arbeiten wurden verschiedene Prozesse der Systeme Weka (Data Mining), OpenDX (Visualisierung) und JFreeChart (Visualisierung) entsprechend semantisch beschrieben und als Web Service zur Verfügung gestellt.

Semantische Daten- und Prozessintegration

Auf der Grundlage der semantischen Beschreibung kann die Integration der Daten und Prozesse erfolgen. Bei der Integration von Datentabellen kann hierdurch von Tabellen- und Attributnamen abstrahiert werden [AF04]. Existiert beispielsweise in einer Explorationsquelle ein Prozess P1, der die Korrelation eines Blutwertes zu einem Leberwerte (HE in T1) visualisiert, so kann dieser Prozess nun auch zur Darstellung der Korrelation zu einem anderen Leberwert (MELD in T2) aus einer anderen Explorationsquelle verwendet werden (Abbildung 3).

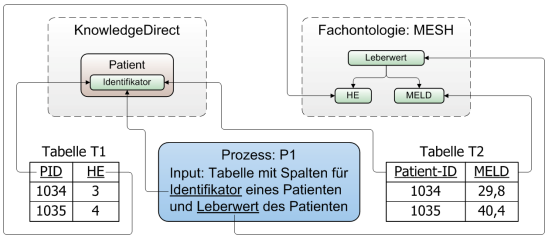


Abbildung 3: Nutzung semantischer Informationen

Auf der Grundlage der semantischen Beschreibungen ist es ebenfalls möglich, Datenbestände zusammenzuführen (Semantic Join) [LR03] und auf deren Basis neue Visualisierungen bzw. Data Mining-Verfahren anzuwenden. So kann beispielsweise mit einem geeigneten Prozess der Zusammenhang zwischen den beiden Leberwerten HE und MELD (Abbildung 3), die sich in unterschiedlichen Explorationsquellen befinden und über den Patientenidentifikator verbinden lassen, dargestellt werden.

Hilfestellungen durch semantische Informationen

Die semantischen Beschreibungen bieten durch Hilfestellungen oder semantische Kontrollen ebenfalls Potential für die Unterstützung des Benutzers bei der Exploration. Visualisierungs- und Data Mining Prozesse lassen sich hinsichtlich Ihrer Eignung für bestimmte Datenstrukturen beschreiben. Auf Basis der im System vorhandenen Metadaten zu den verschiedenen Datentabellen können Vorschläge zur Eignung der bereitgestellten Prozesse gemachen werden [NS04]. So bieten sich beispielsweise Visualisierungen wie Shape Coding oder Parallele Koordinaten erst bei einer großeren Anzahl von Attributen an.
Durch die Nutzung von Fachontologien lassen sich die in den Explorationsquellen bereitgestellten Analyseergebnisse semantisch einordnen und somit besser für weitere Recherchen nutzen. Zudem bieten die Fachontologien dem Benutzer Hilfestellung bei der Auswahl von Attributen. So kann beispielsweise für einen Prozess die Auswahl von Leberwerten (HE, MELD) über die in der Fachontologie enthaltenen Beziehungen leicht erfolgen.

4 Architektur und prototypische Implementierung

Die entwickelten Konzepte wurden in der prototypischen Implementierung *KnowledgeDirect* [Kl04] umgesetzt. Zentraler Kern der Architektur ist das *Knowledge Explore Gateway*, bestehend aus *Control*, *Retrieval* und der *Integration Engine*. Aufgabe der Integration Engine ist die Einbindung verschiedener Explorationsquellen, welche mit der semantischen Integration der in den Quellen bereitgestellten Datenstrukturen, Analyseergebnissen, Analyseprozessen, Data Mining und Visualisierungstechniken einhergeht.

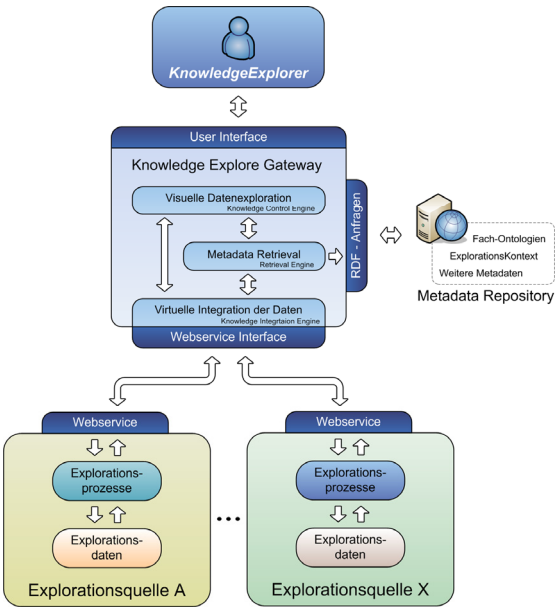


Abbildung 4: KnowledgeDirect-Architektur

Grundlage für die Integration sind semantische Beschreibungen auf der Basis von RDF und OWL, die im Metadatenrepository verwaltet werden. Einen adäquaten Zugriff auf die Metadaten erhält das Knowledge Explore Gateway über die Retrieval Engine (RQL). Die Knowledge Control Engine ermöglicht die einfache Kombination der einzelnen Funktionen und Daten der Explorationsquellen auf Basis der semantischen Beschreibungen und erlaubt weiterhin die Erweiterung um komplexe

Funktionalität auf dem Gebiet des Data Minings und der Visualisierung, wie z.B. 3D-Darstellungen. Der Zugriff auf verschiedene Quellen erfolgt auf der Basis von Web Services und derart transparent, dass sämtliche Aktionen explorationsübergreifend möglich sind. Eine Zuordnung von Daten zu einer bestimmten Explorationsquelle dient nur der Orientierung und birgt keine Einschränkungen in Bezug auf die Nutzung dieser Daten im Zusammenhang mit anderen Explorationsquellen.

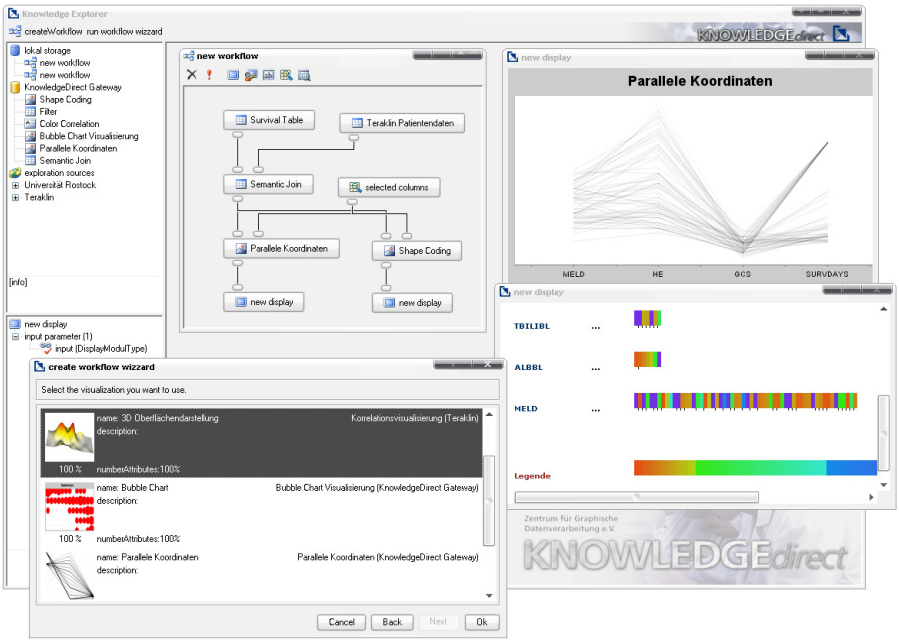


Abbildung 5: KnowledgeDirect Benutzerschnittstelle

Im Rahmen einer prototypischen Implementierung wurden die entwickelten Konzepte und Methoden innerhalb des medizinischen Anwendungsszenarios evaluiert.

5 Zusammenfassung und Ausblick

Das entwickelte KnowledgeDirect-Framework dient der universellen Exploration und semantikbasierten Fusion multimedialer sowie multivariater Datenbestände und ermöglicht es, die getrennt ermittelten Analyseergebnisse unter Nutzung von Ontologiewissen zusammenzuführen. Die von den Explorationsquellen bereitgestellten parametrisierbaren Data Mining- und Visualisierungstechniken, sowie die Analyseprozesse, Rohdaten und aggregierten Daten lassen sich integrieren und erlauben, globale Analysen über den gesamten Datenbestand durchzuführen. Mit semantisch gesteuerter Interaktions- und Navigationstechnik wird es auf einfache Weise ermöglicht, Daten aus verschiedenen Explorationsquellen zu selektieren, zu kombinieren, anzuzeigen und mit ihnen zu interagieren.

Literaturverzeichnis

[AF04] S. Audersch, G. Flach: Universeller Gateway-Ansatz auf der Basis semantisch angereicherter Web Services im Rahmen heterogener eGovernment-Anwendungen, 16. Workshop über Grundlagen von Datenbanken, Monheim, 2004.

[AS04] C. Ahlberg, B. Shneiderman: Visual Information Seeking: Tight coupling of dynamic query filters with starfield displays. Proceedings of ACM CHI94, S.313-317, 1994

[Kl04] T. Klipps: Exploration und semantikbasierte Fusion multivariater Datenbestände in domänenspezifischen Anwendungsumgebungen. Universität Rostock, Diplomarbeit, 2004.

[LR03] U. Leser, P. Rieger: Integration molekularbiologischer Daten. Datenbank-Spektrum 3 (2003) Nr. 6, S. 56-66.

[NS04] T. Nocke, H. Schumann: Meta Data for Visual Data Mining. Proceedings Computer Graphics and Imaging, CGIM 2002, Kaua'i, Hawaii, USA, 2002.

Towards static type checking of Web query language

Sacha Berger, François Bry
Ludwig-Maximilians Universität, Institut für Informatik
Oettingenstr. 67, D-80538 München
sacha.berger@ifi.lmu.de bry@ifi.lmu.de

Zusammenfassung

This article reports on a research project investigating the following two complementary issues: (1) improving how the structure of XML and HTML can be specified, (2) using structure specification (of XML and HTML documents) for static type checking of Web (and Semantic Web) query programs. The first step towards this goal is to provide a schema language like DTD, XML Schema or Relax-NG with better support of graph structured data.

1 Introduction

The schema languages DTD, XML Schema[Con01] and Relax-NG[CM01] are used to specify the type and structure of XML and WWW documents. Although documents and data on the Web often represent graph structures based on references (expressed using eg. ID and IDREF attributes, RDF triples or Hyperlinks), the above mentioned schema languages can only specify tree structures. This article first introduces a novel schema language for specifying XML documents and semi structured data called Regular Rooted Graph Grammars – also referred to as R^2G^2. R^2G^2 gives rise to specify graph shaped XML and semistructured related data types possibly with (directed or undirected) cycles. R^2G^2 is an extension of regular tree grammars (that underly many Web schema languages).

The second issue addressed is using a regular rooted graph grammar as a type language for static type checking of web query or transformation programs. Static type checking consists of (1) inferring data types for all program constructs and (2) detecting type inconsistencies, both at compile time, ie. before query evaluation. The main advantage of static type checking is, as its proponent Robin Milner said, that "well-typed programs do not go wrong"[Mil74]. The Web query language Xcerpt is used as test bed for using Regular Rooted Graph for static type checking.

2 R^2G^2introduced on an example

XML documents are serialisations of tree, or graph, structured data i.e. representations in a linear, or textual, form using parenthesis, the opening and closing tags. Non-tree XML documents play an important role in practice. An example of such an XML document is given below, in which children elements are used to describe (possibly symmetric) friendship relationships. Recall that DTD, XML Schema and Relax-NG cannot express the structure of graph structured XML documents. R^2G^2grammars have been designed to model graph structured documents (possibly containing directed or undirected cycles).

The following figure is (part of) an XML document expressed in Xcerpt syntax. used is actually the Xcerpt data term[SB04] syntax:

```
addressbook{ id1@card{ name{"Snoopy"}, friends[^id1,^id2], phone{"9310"} },
             id2@card{ first{"Charly"}, last{"Brown"}, phone{"9316"} }
           }
```

According to [SB04], such an element is called in the following a *dataterm*. The example represents two addresses (called `card`) of an `addressbook`. The `card` elements are defining

occurrences [1] of elements that may be referenced using the unique identifiers at the left of the `@` signs (as seen in the friends element of the first card). The use of square brackets indicate that the order of the sub elements is relevant (as always in XML), the curly braces indicate the irrelevance of the sub elements order (as common in databases, an extension of XML introduced by Xcerpt).

The following example is a R^2G^2 specifying a structure, or schema, of addressbooks like the previous one. Due to lack of space, R^2G^2will informally be introduced on this example:

```
elmtype AddressBook = addressbook{ @Card* };
elmtype Card        = card{{ Name,
                             friends[ ^Card+ ]?, Contact* }}
elmtype Contact     = phone{ String } | /[Ee]?mail/{ String };
elmtype Name        = name{ String };
```

The grammar consists of 6 rules similar to rules of regular tree languages. The first rule, given on the first line, defines an element type named `AddressBook` with label `addressbook`. In contrast to DTD and similar to Relax-NG and XML-Schema, the element label and the type name may not be the same or related to each other, so that the same element label can be used with different structure in different places in a document.

The right hand side of a rule may be an element type definition term (as seen in the 1st, 2nd and 4th rule) or a disjunction of type definition terms (as seen in the 3rd rule). The content of an element type definition term is a regular expression of element type names or element type definition terms. Elements of type `Card` e.g., contain an element of type `Name`, an optional element labelled `friends` and an arbitrary number of elements of type `Contact` as children. The element type definition term of this rule is enclosed by double braces, indicating that the content definition is incomplete, that therefore elements conforming to this definition may contain further arbitrary elements. This is a convenience feature not used in current XML schema languages, but useful for modelling partly arbitrary content models[2].

Content models enclosed by square brackets (used in the 2nd rule) are ordered content models: only ordered element instance's content can conform to such a content model[3]. Content models enclosed by curly braces are content models with irrelevant order, instances of those types may either be ordered or unordered.

It is possible to model elements with the same type name and content model, but with different labels: this is accomplished by using a string regular expressions (enclosed by "/ßlashes) at the position of the label in the type definition term (see the 3rd rule).

As in the 1st rule's type definition term the `Card` is prefixed with an `@`, child elements of type `Card` must have an an unique identifier to be referrable. The modelling of elements with unique identifier is a feature also available in XML Schema, DTD and Relax-NG: they provide the ability to model ID attributes, which is of comparable expressiveness as Xcerpt's defining occurrence mechanism. In the second rule, a reference type (`^Card`) is part of the definition of the friends element's content model: elements conforming to this type definition term may only contain references to elements of type Card. In contrast to the typed reference mechanism in R^2G^2, current schema languages only provide the ability to model references to arbitrary ID attributes. Hence R^2G^2provides a better way to model graph shaped XML documents.

Further Features of R^2G^2 not mentioned above (for space reason) are (1) *content model rules*, used to provide names for parts of content models, (2) *alias rules*, used to provide alternative names to the same type, (3) *scalar types*, as `String` in the former example, (4) *label types*, to provide names to sets of alternative labels (or label regular expressions) and (5) *local definition and modules*.

3 Validation of Data terms with R^2G^2

Validation, commonly referred to as acceptance test in automata theory, means to test if a "word" is accepted by an automaton, ie. if the word is part of a language associated to the

[1] The term *defining occurrence* is introduced in [SB04]

[2] The expressiveness of $R^2G^2$is not enhanced by this extension, cf. section 3.

[3] A precise definition of the matching is given in section 3.

automaton or its corresponding grammar. Here an XML document is the word to test for acceptance under the grammar (or its corresponding automaton) represented by the schema, with the root indicating the non terminal to use as start symbol.

The approach to validation presented here, is an extension of tree automaton-based validation [MLM01], more precisely, a non deterministic bottom up approach of regular tree grammar based validation [MLM01]. Validation is done using a recursive function returning a set of non terminal symbols for validated elements. Each non terminal represents one possible derivation for the element under the given grammar, an empty set indicates an invalid element.

$$\begin{array}{rcl} validate(\mathbf{l}[\mathbf{t_1},\cdots,\mathbf{t_n}],G) & = & \{X|N_1\cdots N_n \in L(r_c) \wedge \texttt{elmtype X} = \mathbf{l}[r_c] \in G\} \\ & where & N_i \in validate(\mathbf{t_i},G) \end{array}$$

This approach needs some extension to cope with the following R^{2}G^{2}features: (1) incomplete or double brace content models, (2) label regular expressions, (3) unordered or curly brace content models and (4) handling of typed references.

$$\begin{array}{rcl} validate(\mathbf{l}\{\mathbf{t_1},\cdots,\mathbf{t_n}\},G) & = & \{X|W \in L(r_c) \\ & & \wedge W \in permutation(N_1\cdots N_n) \\ & & \wedge \mathbf{l} \in L(r_l) \\ & & \wedge \texttt{elmtype X} = r_l\{r_c\} \in G'_{+ALL} \\ & where & N_i \in validate(\mathbf{t_i},G) \end{array}$$

$$\begin{array}{rcl} validate(\mathbf{l}[\mathbf{t_1},\cdots,\mathbf{t_n}],G) & = & \{X|W \in L(r_c) \\ & & \wedge(\ (W \in permutation(N_1\cdots N_n) \\ & & \quad \wedge \mathbf{l} \in L(r_l) \\ & & \quad \wedge \texttt{elmtype X} = r_l\{r_c\} \in G'_{+ALL}) \\ & & \vee(W = N_1\cdots N_n \\ & & \quad \wedge \mathbf{l} \in L(r_l) \\ & & \quad \wedge \texttt{elmtype X} = r_l[r_c] \in G'_{+ALL}))\} \\ & where & N_i \in validate(\mathbf{t_i},G) \end{array}$$

$$validate(\mathbf{id}@\mathbf{l}\alpha\mathbf{t_1},\cdots,\mathbf{t_n}\beta,G) = \{@X|X \in validate(\mathbf{l}\alpha\mathbf{t_1},\cdots,\mathbf{t_n}\beta,G)$$

$$validate(\hat{}\mathbf{id},G) = \{@X|X \in validate(lookup(\mathbf{id}),G)$$

The modified grammar G'_{+ALL} expresses the same as G, but does not use double braces, ie. incomplete content models. Rules with incomplete content model are replaced by rules with complete content model, by (1) changing the double braces into single braces, while maintaining the order type of the braces, (2) adding `ALL*` as first atom of the regular expression and (3) modifying the atoms of the content regular expression as follows:

1. replacing all `a*` atoms by `(a,ALL*)*`
2. replacing all `a+` atoms by `(a,ALL*)+`
3. replacing all `a?` atoms by `(a,ALL)?`
4. replacing all other atoms r_a by r_a`,ALL*`

The rule for the `ALL` type is

```
elmtype ALL = /.+/{ALL*} | /.+/[ALL*] | String
```

For example, the 2nd grammar rule of section 2 can be rewritten to

```
elmtype CARD = card{ ALL*, NAME,
                     ALL*, (friends[^CARD], ALL)?,
                     (Contact, ALL)* }
```

The variables α and β (in the 3rd rule) has been used to capture square and curly braces, as they are treated the same way. The rules with incomplete content models have been rewritten using the ANY type and the ANY type is added to the set of rules. The ANY type captures arbitrary content[4].

[4]The possibility to rewrite any grammar G to a corresponding schema G'_{+ALL} indicates, that double braces do not extend the expressiveness of R^{2}G^2.

Label regular expressions are processed by checking the containment of the data term's label against the language defined by the label regular expression or the type definition term's label interpreted as regular expression.

Unordered and ordered data terms, ie. terms like `name{"Snoopy"}` (unordered) and `friends[^id1, ^id2]` (ordered), are handled in the first, respectively in the second function definition case. Note that in essence the right hand side of the first case is contained in the second case, reflecting that ordered data terms may also match with unordered content model specifications. The containment test for unordered terms is achieved by checking the containment of at least one permutation of the unordered content with respect to the regular expression of a given content model.

Validation of defining occurrences of identifiers and of references is handled by the 3rd, respectively the 4th function definition case. For this purpose a *lookup* function, characteristic to each document validated, is used. An implementation could rely on a hashtable, initialised by an additional pass on the document.

Note that a naive implementation of the algorithm explained above, may not terminate when validating circular structures. Termination can easily be ensured by using lazy evaluation or by labelling validated nodes with their matching types, this means by relying on the set of types used as label in preference to calculating this set again.

4 R^2G^2 for static type checking of web query languages

Static type checking means detecting type inconsistencies at compile time, ie. before query or program evaluation. Essential properties of type checking are decidable and efficiently computability. For these purposes, and in contrast to model checking, type checking works on approximations of the values a program may use or produce, more precisely on superset approximations of the actual set of possible values a specific program construct may compute[Mil74, Car97, Pie02]. All constructs of a language that are to be type checked, must therefore be typeable, this means, such a superset approximation must be assignable to those constructs. The art of finding a good type system is, to have the type approximation as precise as possible, and therefore to find as many errors as possible, while being as general as necessary to still ensure termination and mostly polynomial complexity for the type checking process.

Traditionally all typable constructs of a program had to be explicitly type annotated manually, but modern type systems provide the ability to automatically deduce type annotation based on type inference[Pie02]. This brings the freedom of omitting or providing the type annotation as wished by the programmer. The main advantage of static type checking is, as its proponent Robin Milner said, that "well-typed programs do not go wrong"[Mil74]. General principles of a type system for Web query languages are presented in the following.

The idea of type checking applied to Web query languages means, to detect prior to execution, that query programs (1) do not query the desired data, and that they (2) produce undesired output. Common to many query languages are constructs to (1) query given data or Web sites, and (2) construct results based on the queried information. Language components used for querying usually restrict structure and values of expected results by filtering and pruning elements of the queried data. The schema of elements in the result set can be expressed using R^2G^2. Language components for the construction of answers to a query program usually provide a template-like mechanism to rearrange and integrate the data collected using query constructs. The schema of elements constructed according to the template and the type of the queried data can also be expressed using R^2G^2. According to the specialities of a concrete web query language, further constructs may be of relevance for typing.

Based on the two language component classes that may be typed, two classes of errors exist:

- Query terms may be ill-typed, because the type of the queried data or web site has an empty intersection with the type associated to the query result set (this means, that a query may never return anything when applied to a valid data base)
- Construct terms may be ill-typed, because the type of their result set is not contained in the type of the expected result (this means, the constructed results may have invalid

structure or content with respect to a result type annotation)

Note, that therefore it is necessary to (1) calculate the intersection of two data types, (2) that the inclusion test of a type in another type is still decidable, and that (3) the test of emptiness of a type is decidable. All three properties should be achievable with R^2G^2(not proved by now).

Based on principles previously presented, a type system for the Web query language Xcerpt based on $R^2G^2$is currently being developed as test bed. As there exists already a regular tree grammar based approach of typing Xcerpt [AW03], this will be used and extended to provide type checking for Web queries querying and constructing tree and graph structured data.

5 Conclusion

This article presented R^2G^2, a grammar formalism for specifying schemas of graph structured XML documents and semi structured data. A validation algorithm based on non deterministic tree automaton techniques [MLM01] is presented. Further work needs to be done, to prove some set theoretical properties of R^2G^2's needed for static type checking. Based on R^2G^2 a type system for Xcerpt will be conceived.

Literatur

[AW03] W. Drabent A. Wilk. On Types for XML Query Language Xcerpt. In François Bry, Nicola Henze, and Jan Maluszynski, editors, *PPSWR*, volume 2901 of *Lecture Notes in Computer Science*, pages 128–145. Springer, 2003.

[Car97] Luca Cardelli. *Type Systems*, chapter 103. CRC Press, Boca Raton, FL, 1997.

[CM01] James Clark and Makoto Murata. *RELAX NG Specification.* http://relaxng.org/spec-20011203.html, 2001. ISO/IEC 19757-2:2003.

[Con01] W3 Consortium. XML Schema Part 0: Primer. W3C Recommendation, 2001. http://www.w3.org/TR/xmlschema-0/.

[Mil74] Robin Milner. Fully Abstract Models of Typed lambda-Calculi. *Theoretical Computer Science*, 4:1–22, 1974.

[MLM01] M. Murata, D. Lee, and M. Mani. "Taxonomy of XML Schema Languages using Formal Language Theory". In *Extreme Markup Languages*, Montreal, Canada, 2001.

[Pie02] Benjamin C. Pierce. *Types and Programming Languages.* MIT Press, 2002.

[SB04] Sebastian Schaffert and François Bry. Querying the Web Reconsidered: A Practical Introduction to Xcerpt. In *Proceedings of Extreme Markup Languages 2004, Montreal, Quebec, Canada (2nd–6th August 2004)*, 2004.

Transaktionale Garantien in mobilen Ad-Hoc Netzen

Joos-Hendrik Böse
Freie Universität Berlin, Institut für Informatik, Takustr. 9, 14195 Berlin
boese@mi.fu-berlin.de

Zusammenfassung

Transaktionale Garantien, wie z.B. Atomarität, werden in festen Netzen durch Protokolle realisiert, die auf der Annahme basieren, dass Kommunikationsfehler selten auftreten. In mobilen Umgebungen müssen Kommunikationsfehler aber als Standardfall angenommen werden. Entsprechend müssen neue Ansätze und Architekturen für mobile Umgebungen gefunden werden. Das hier beschriebene Projekt schlägt eine Architektur vor, welche es ermöglichen soll, Transaktionen mit definierten Ausführungsgarantien in mobilen Netzen zu nutzen. Der Fokus liegt dabei auf Atomaritätsgarantien in Handelstransaktionen. Hierzu wird ein entsprechendes Protokoll präsentiert, welches die vorgeschlagene Architektur benutzt, um Atomarität bei Tauschtransaktionen zwischen mobilen Knoten zu gewährleisten.

1 Motivation

Die Forschung zu mobilen Transaktionen beschäftigte sich bisher vor allem mit der Behandlung von Verbindungsunterbrechungen zwischen einem mobilen Client und einem zentralen Datenbankserver. Es werden dabei Transaktionen betrachtet, die Replikate auf mobilen Endgeräten ändern, während diese keine Verbindung zu dem zentralen Datenbankserver haben [3, 4, 9]. Wesentliches Problem dabei ist die Synchronisation der geänderten Kopien mit den Originalen zu einem späteren Zeitpunkt. Die Verbindung zum Server wird in diesen Szenarien meist kontrolliert abgebrochen, was die Anwendung von Protokollen zur Sicherung von Konsistenz ermöglicht.

Bei Transaktionen zwischen mobilen Knoten, die in mobilen Ad-Hoc Netzen (MANETs) organisiert sind, muss dagegen jederzeit mit unkontrollierten Verbindungsabbrüchen gerechnet werden. Stehen keine Informationen über das Bewegungsmodell und den Verbindungskontext der an der Transaktion beteiligten Knoten zur Verfügung, muss angenommen werden, dass die Knoten jederzeit verschwinden können und zu einem beliebigen Zeitpunkt wieder auftauchen. In solch einem System lässt sich nicht sicherstellen, dass transaktionale Garantien, wie z.B. Atomarität, hundertprozentig erfüllt werden können. Gerade Businesstransaktionen haben aber besondere Anforderungen an Transaktion, wie z.B. Geld-Atomarität, Waren-Atomarität und Fairnessgarantien[1] [1, 6]. Bezieht man dagegen Informationen über den Verbindungskontext und das zugrunde liegende Bewegungsmodell mit in die Transaktionsverarbeitung mit ein, so lassen sich Wahrscheinlichkeiten für das Erreichen von bestimmten Garantien abschätzen.

Das hier beschriebene Projekt schlägt eine Middleware Architektur vor, welche die Ausführung von Transaktionen in MANETs mit transaktionalen Garantien ermöglichen soll. Als Anwendungsszenario dient dabei der Austausch von geringwertigen elektronischen Gütern z.B. Bonuspunkte, Musikdateien etc. in einem MANET. Während andere ACID Garantien wie z.B. Isolation ebenfalls wichtig sind, beschränkt sich diese Arbeit zurzeit auf die Betrachtung von Atomaritätsgarantien. Zentrale Komponente dieser Middleware ist der sogenannte Shared Log Space (SLS), der im Abschnitt 2 beschrieben wird. Transaktionsprotokolle, welche die Middleware nutzen, müssen evtl. angepasst werden. Ein Beispiel für solch ein Protokoll wird im Abschnitt

[1] Unter *Fairem Tausch* wird verstanden, dass nach dem Tausch entweder jede Partei das erhalten hat was vor dem Tausch vereinbart war oder keine der beiden einen Vorteil erlangt hat.

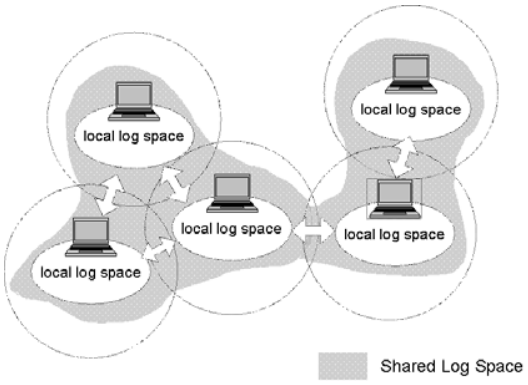


Abbildung 1: Shared Log Space

3 beschrieben und diskutiert. Abschnitt 4 fasst den aktuellen Stand der Arbeit zusammen und gibt einen Ausblick auf noch offene Fragen und zukünftige Aktivitäten.

2 Shared Log Space Middleware

In festen Netzen können Atomaritätsgarantien gegeben werden, die auf der Annahme basieren, dass Prozesse, die an der Transaktion beteiligt sind, Fehler erkennen können und entsprechende Recovery Protokolle ausführen. Diese Protokolle blockieren eventuell, führen aber irgendwann in einen definierten Zustand. Dabei werden vorhandene Logdaten analysiert, um den Stand der Transaktion zu ermitteln. Ist dies auf Basis von lokalen Logdaten nicht möglich, muss mit anderen Teilnehmern der Transaktion kommuniziert werden. Problem in einem MANET ist aber, dass diese Teilnehmer evtl. nicht mehr erreichbar sind.

Idee des Shared Log Space (SLS) ist, die Wahrscheinlichkeit zu erhöhen, mit der ein Knoten im Rahmen eines Recovery Prozesses relevante Logdaten und damit den Status seiner Transaktionen erfahren kann. Damit erhöht sich direkt die Wahrscheinlichkeit, dass die Transaktion zu einem definierten Ende kommt und Korrektheit gewährleistet ist. Um die Verfügbarkeit von Logdaten in einem MANET zu erhöhen, werden diese kontrolliert auf andere Knoten repliziert. Knoten, die aufgrund von Bewegung nicht mehr direkt miteinander kommunizieren können oder nicht zum selben Zeitpunkt mit dem Netz verbunden sind, können trotzdem noch Logdaten des anderen lesen.

Abbildung 1 illustriert die Idee des SLS[2], dieser bildet sich durch die Replikation von lokalen Logdaten auf andere Knoten im MANET. Jeder Knoten der am SLS teilnimmt verfügt über einen lokalen Log Space (LLS) als Middlewarekomponente, durch ihn "sieht" er die im Netz vorhandenen Logdaten. Tritt ein Fehler während einer Transaktion auf, weil z.B. ein Knoten nicht mehr antwortet, so wird die Transaktion abgebrochen und entsprechende Logdaten z.B. eine *Abort* Nachricht in den LLS geschrieben. Im Rahmen einer fehlerfreien Transaktionsverarbeitung werden zu jedem neuen Zustand der Transaktion entsprechende Logdaten in den LLS geschrieben. Hat der Knoten später wieder Kontakt zum MANET und damit zum SLS, synchronisiert er seinen LLS mit den LLS von Knoten in seiner unmittelbaren Umgebung und startet entsprechende Recovery Protokolle.

Die Wahrscheinlichkeit, mit der Logdaten gefunden werden, hängt vom Verteilungsgrad der Daten im Netz ab. Dieser Verteilungsgrad ist um so größer, je mehr Aufwand in Form von Nachrichtenversand bei der Verteilung betrieben wird. Wie viele Knoten erreicht werden und mit welchen durchschnittlichen Wahrscheinlichkeiten diese ausfallen, wird durch das zugrunde

[2] Anmerkend ist zu erwähnen, dass die Idee des SLS wesentlich von Koordinationsansätzen wie Linda [2] und Lime[5] geprägt ist.

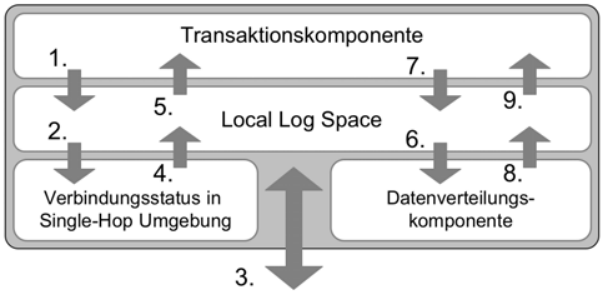


Abbildung 2: Zusammenspiel der einzelnen Middleware Komponenten

liegende Bewegungsmodell sowie durch den aktuellen Verbindungskontext des einzelnen Knotens beeinflusst. Erste Aussagen über den Zusammenhang zwischen Verteilungsstrategien und dem zugrunde liegenden Bewegungsmodell lassen sich aktuellen und laufenden Forschungsarbeiten wie z.B. [7, 8] entnehmen.

Informationen über den aktuellen Verbindungskontextes, wie z.B. die Anzahl der Knoten in direkter Umgebung, deren Bewegungsrichtung und Geschwindigkeit etc. erfordert einen "Cross Layer" Ansatz. Solch ein Ansatz basiert auf der Beobachtung, dass verwendete Netzwerkarchitekturen, wie TCP/IP, aufgrund ihres Schichtenmodells Informationen über den Verbindungskontext für die Anwendungsschicht nicht verfügbar machen. Informationen über den Verbindungskontext sind in mobilen Netzen aber nötig, um Transaktionen besser koordinieren zu können. Kann man erfahren, wie sich z.B. ein Knoten bewegt, so könnte bei der Vergrößerung des Abstandes zwischen zwei Knoten eine Transaktion, die zwischen diesen läuft, gesplittet oder partiell abgeschlossen werden.

Das aktuelle prototypisch implementierte System besteht aus den in Abbildung 2 gezeigten Komponenten. Handelt es sich bei dem Knoten um den Initiator einer Transaktion, so interagieren die Komponenten wie folgt miteinander: 1. die Transaktionskomponente stellt eine Anfrage an den LLS, in der eine bestimmte Mindestwahrscheinlichkeit für eine erfolgreiche Recovery im Fehlerfall gefordert wird. In 2. und 4. ermittelt der LLS relevante Parameter des eigenen Verbindungskontextes (2. und. 4.) um zu bestimmen, welcher Verteilungsgrad von ihm erreicht werden kann. In 3. werden diese Informationen von den Transaktionspartnern eingesammelt, um diese dann zu aggregieren und in 5. der Transaktionskomponente die erreichbare Recoverywahrscheinlichkeit mitzuteilen. Werden die Anforderungen der Transaktionskomponente erfüllt, so kann die Transaktion in 6. gestartet werden. Schreibt die Transaktionskomponente Logdaten, so werden diese vom LLS an die Verteilungskomponente weitergegeben (7.), die eine erfolgreiche Verteilung auf eine bestimmte Mindestanzahl von Knoten an den LLS meldet (8.). Nimmt ein Knoten nicht als Initiator an einer Transaktion teil, so meldet er seinen Verbindungskontext an den Initiator, die Interaktion der Komponenten während der Transaktionsausführung erfolgt analog in den Schritten 6. bis 9. Für diese Architektur wurde ein Protokoll konzipiert, welches im folgenden Abschnitt beschrieben wird.

3 Ein Protokoll für Handelstransaktionen auf Basis des SLS

Fokus dieses Projektes ist wie oben beschrieben, Atomarität insbesondere bei Tauschtransaktionen zu gewährleisten. Der aktuelle Prototyp verwendet dabei ein an [1] angelehntes Protokoll, um Fairness und Atomarität sicherzustellen. Fairness wird gewährleistet, indem nach erfolgreichen Ablauf des Protokolls beide Teilnehmer signierte Verträge über den Inhalt des Tauschgeschäftes besitzen. Erhält einer der Teilnehmer nicht das vereinbarte Tauschobjekt, so kann er zumindest mit seinem Vertrag beweisen, dass er der berechtigte Eigentümer des Objektes ist. Dies erfordert

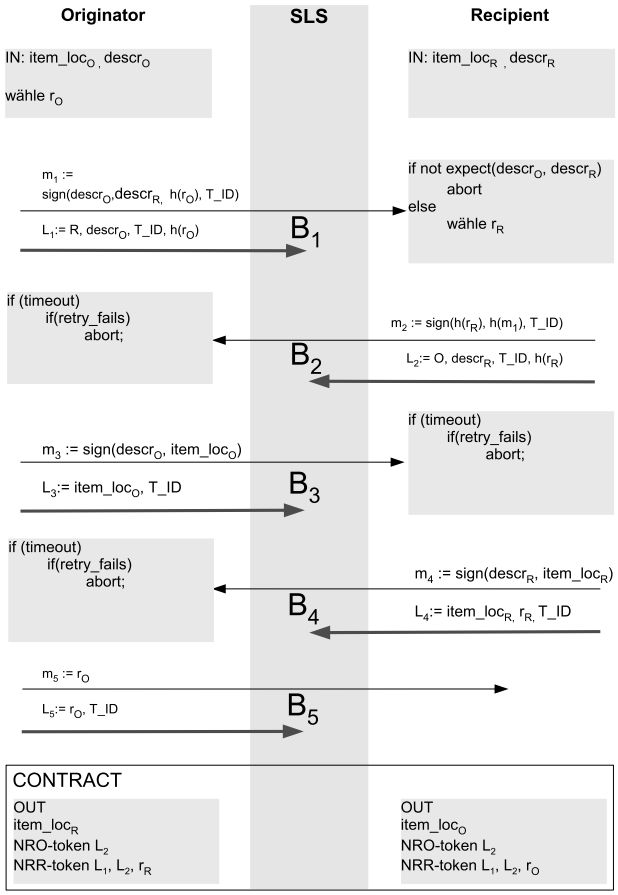


Abbildung 3: Protokoll für Handelstransaktionen

zusätzlich das Vorhandensein einer dritten vertrauenswürdigen Instanz, bei welcher der Vertrag eingeklagt werden kann. Diese ist allerdings nicht Thema dieses Projektes.

Abbildung 3 zeigt das Protokoll mit dem die entsprechenden Verträge geschlossen werden, der Austausch des eigentlichen Gutes ist nicht Teil des Protokolls. Dahinter steckt die Idee, dass ein transaktionaler Vertragsschluss eine kurz laufende Transaktion darstellt, während der Austausch von elektronischen Gütern, z.B. Musik oder Videodateien relativ viel Zeit benötigt. Die Wahrscheinlichkeit, dass ein Fehler während des Vertragsschluss auftritt ist also ungleich kleiner als bei der Übertragung eines großen elektronischen Gutes. Auf Vertragsebene wird sich das System meist in einem sehr konsistenten Zustand befinden, während die Speicherorte der konkreten Güter sich erst mit einer Verzögerung den vertraglichen Besitzbeziehungen anpassen.

Wird das Protokoll erfolgreich durchlaufen, verfügt jeder Teilnehmer über einen NRO- (Non Repudiation of Origin) Token, einen NRR- (Non Repudiation of Receipt) Token und über den Ort[3] an dem das Gut verfügbar ist. Mit Hilfe des NRO und des NRR Token kann jede Partei einer dritten Instanz gegenüber beweisen, dass sich der andere zu dem Tauschgeschäft verpflichtet hatte bzw. dass dieser das ausgetauschte Gut erhalten hat.

Während der Ausführung des Protokolls kann an jeder Stelle die Verbindung zum Tauschpartner unterbrochen werden. Eine Verbindungsunterbrechung wird nach Ablauf eines Timeouts angenommen. Falls der Knoten noch über eine Verbindung zum SLS verfügt, d.h. er noch Nachbarn die am SLS beteiligt sind erreichen kann, so schreibt er eine *Abort* Nachricht in den SLS und bricht die Transaktion lokal ab. Hat der Knoten keine Verbindung mehr zum SLS, dann

[3]Ein Ort kann z.B. eine URL oder ein Service sein, an dem das Item "abgeholt" werden kann.

blockiert er die Ausführung des Protokolls. Ist der SLS wieder erreichbar, synchronisiert sich der LLS mit dem SLS um dann die Recovery der blockierten Transaktionen zu starten. Im Rahmen der Recovery werden die Logdaten im SLS analysiert, findet sich eine *Abort* Nachricht für eine bei Verbindungsabbruch aktive Transaktion, so wird diese lokal ebenfalls beendet. Ansonsten wird das Protokoll an der entsprechenden Stelle weitergeführt.

Das hier beschriebene Protokoll sowie die SLS Middleware wurde auf .NET für PocketPC (.NET Compact Framework) implementiert und in IEEE 802.11 Ad-Hoc Netzwerken getestet. Im Folgenden werden offene Probleme und zukünftige Forschungsarbeiten in diesem Projekt kurz umrissen.

4 Offene Fragen

Eine bisher unbehandelte Frage ist, wie lange Logdaten von einzelnen Knoten vorgehalten werden sollen. Da diese lokale Resourcen belegen, ist einerseits eine möglichst schnelle Freigabe wünschenswert, andererseits beeinflusst eine zu frühe Freigabe die Recoverywahrscheinlichkeiten. Nicht trivial ist die Bewertung der vorgeschlagenen Architektur. In Zukunft soll das System in eine Simulationsumgebung eingebettet werden, die eine kritische Masse an mobilen Knoten und ein MANET simulieren kann. Tests mit dem aktuellen Prototypen und bis zu 20 Knoten haben bisher keine verwertbaren Ergebnisse gebracht, sondern dienen eher als Realitätscheck für die Implementierungen. Weiterhin werden die vorhandenen Verteilungsstrategien weiter optimiert und deren Zusammenhang mit den Bewegungsmodellen untersucht.

Literatur

[1] N. Asokan, Matthias Schunter, and Michael Waidner. Optimistic protocols for fair exchange. In *ACM Conference on Computer and Communications Security*, pages 7–17, 1997.

[2] Nicholas Carriero and David Gelernter. Linda in context. *Commun. ACM*, 32(4):444–458, 1989.

[3] R. A. Dirckze and Le Gruenwald. A pre-serialization transaction management technique for mobile multidatabases. *Mob. Netw. Appl.*, 5(4):311–321, 2000.

[4] Margaret H. Dunham, Abdelsalam Helal, and Santosh Balakrishnan. A mobile transaction model that captures both the data and movement behavior. *Mob. Netw. Appl.*, 2(2):149–162, 1997.

[5] A. Murphy, G. Picco, and G.-C. Roman. Lime: A middleware for physical and logical mobility. pages 524–536.

[6] Henning Pagnia and Felix C. Gärtner. On the impossibility of fair exchange without a trusted third party. Technical Report TUD-BS-1999-02, Darmstadt, Germany, 1999.

[7] M. Scholz S. Bittner, W.-U. Raffel. Heterogeneous mobility models and their impact on data dissemination in mobile ad-hoc networks. Technical Report B 04-16, Berlin Germany, 2004.

[8] M. Scholz S. Bittner, W.-U. Raffel. The area graph-based mobility model and its impact on data dissemination. In *Proceedings if the Third IEEE International Conference on Pervasive Computing and Communications Workshops (Perkomw' 05)*, pages 268–272. IEEE2005, March 2005.

[9] Patricia Serrano-Alvarado, Claudia L. Roncacio, and Michel Adiba. Mobile transaction supports for dbms: An overview.

Context Modeling in OWL for Smart Building Services

François Bry[1], Takashi Hattori[2], Kaoru Hiramatsu[2],
Takeshi Okadome[2], Christoph Wieser[1], and Tatsumi Yamada[2]

[1]University of Munich (http://www.pms.ifi.lmu.de/)
[2]NTT Communication Science Laboratories (http://www.kecl.ntt.co.jp/)

Abstract

This article presents an approach to context-aware services for "smart buildings" based on Web and Semantic Web techniques. The services striven for are first described, then their realization using Web and Semantic Web services is explained. Finally, advantages of the approach are stressed.

1 Introduction

An automated recognition of the people's location in a building (at desk, in a room, underway, etc.) as kind of so-called "contexts" [Dey01] makes it possible to offer advanced services such as

1. context-aware message delivery,
2. context-aware access control,
3. context-aware services for disabled persons, and
4. context-aware management of the technical infrastructure.

Such context-aware services are referred to as "smart building" services.

As a first working hypothesis it is assumed that data on room occupancy, people's locations, and people's activities (e.g. working at one's own desk using a desktop computer, sitting at a conference table, etc.) are collected using cheap sensors such as RFID sensors. Especially RFID sensors can be placed at various locations such as in a gangway, on a chair, etc. The restriction to cheap sensors such as RFID sensors, excluding, e.g., video cameras, is convenient because such sensors are easy to install and to maintain, and processing such sensor data is simpler than, e.g., image processing.

As a second working hypothesis it is assumed that sensor data is expressed in Semantic Web formalisms such as OWL [W3C04]. A Semantic Web approach to the "smart building" perspective is convenient because the Web provides with an widespread and easily deployed data interchange infrastructure, and the emerging Semantic Web provides with an increasing number of versatile data modeling and reasoning methods.

This article describes an approach to such "smart building" services relying on the Web and on Semantic Web formalisms and methods. Central software components of this approach are described: The modeling of knowledge is realized in different ontologies called Micro-Theories written in OWL. The Micro-Theories are used to specify domain specific knowledge, e.g., of the furniture of a building. Such ontologies are integrated into one "Upper Ontology" specifying common sense knowledge such as the ways how to use the furniture. The next component is the derivation of context. As last component, data retrieval is realized through simple queries and ontology reasoning using Semantic Web Query Languages.

Finally, this article explains why standards (1) for the data, sensors might collect in buildings, (2) for an ontology of common life objects, properties and action, and (3) for an interface between the afore mentioned formalisms make it easier to reason on such data and knowledge.

2 Context-aware Services for a "Smart Building"

An automated recognition of people's location (at desk, in a room, underway, etc.) in a building makes it possible to offer advanced services, especially:

Context-aware message delivery. Nowadays one can communicate from every place in an office building, and additionally one can choose between various kinds of communication technologies. Written text or spoken voice can be transmitted synchronous or asynchronous via a mobile phone. The flexible availability of users leads to a bigger extend of communication, catching one's attention frequently. While the sender can choose the time and the technology for communication depending on his context, e.g., while walking or being in the office, the receiver cannot.

Obviously, a smart delivery of messages depending on the receiver's context, would make communication more convenient. A manager attending a meeting wants to get messages only, if they are relevant either for him personally or for the meeting. Hence, instead of arriving during the meeting, a less important email could be delivered after while walking back to his or her office. Since reading email while walking is not convenient, the message could be delivered as spoken text automatically.

Context-aware access control. Office buildings are not open to the public for security reasons. Hence, different strategies such as using different door keys provide access control. This technique causes especially two problems:

1. Providing access to parts of such a building demands a special set of keys necessary, and
2. changing access rights would mean (time-dependant) changing the set of keys.

A feasible solution offers a "smart building" for instance a hotel. Being recognized by the "smart building", the guest living in the hotel has access to his or her room. Since the room is located in an upper floor, the guest may also access the rooms connecting the entrance of the hotel as well as the room such as the stairway or the elevator. Obviously there are other access policies for employees of the hotel. The room service, e.g., has no access to rooms automatically, while guests are inside. In case of an emergency like a fire, the access rights could change for everybody immediately. Everybody could be given access to parts of the hotel, that guide to an exit, such as the stairway, whereas the access to elevators is not granted.

Context-aware services for disabled persons. Orientation in buildings can become difficult for disabled people, if information is needed, that is only available via senses they are impeded. Blind people could be helped while walking through unknown rooms, e.g., walking to a meeting. Since having no knowledge about barriers such as furniture and doors of a room, a blind person could just guess the right way.

An advanced guidance being aware of a person's position in a building could give advise via synthesized speech. Collisions especially with moving barriers could thus be avoided. Arriving at the meeting room, more information could be offered due to the change of context. For example names of the present persons and their relative positions to the blind person could be read out using a voice synthesizer.

Context-aware management of the technical infrastructure. Office buildings are equipped with technical infrastructure such as lamps and jalousie for controlling the brightness and heating or with an air conditioning for managing temperature and humidity. Such sophisticated technical infrastructure is hard to control efficiently with a minimum of power consumption including a high level of convenience.

When a speaker giving a talk in a "smart building" enters the underground car park the lights will be switched on automatically there. As a second consequence of his arrival the air conditioning will start tempering the conference room for the talk in an upper floor. After leaving his or her car the speaker walks through the building to the conference room. On his way the lights will turn on in the current corridors only. While at first the lights in the conference room are bright, later the light intensity is reduced to a lower level during the presentation.

3 A Semantic Web / RFID approach to "Smart Buildings"

Micro-Theories. Obviously, a system offering domain specific services needs to be aware of its domain. In the present case knowledge about the domain of a building needs to be modeled. Several languages such as Topic Maps [Top01] or DAML&OIL [W3C01] and OWL have been developed to specify so called ontologies for knowledge representation. Being the recommendation of the W3C as ontology language we decided to use OWL because there are a lot of tools available for editing ontologies such as Protege [NSD+01], and tools for data retrieval and management such as the Jena Semantic Web Framework [HP05].

Using OWL, an ontology of a "smart building" service can be designed in a bottom-up approach beginning from the basic concepts of an office building such as a chair or a conference room and relations to each other. A chair, e.g., is located next to a table in a room. Further on this room has a direct connection to the conference room via a door, etc. This ontology is restricted to knowledge that is specific to a office building. Common sense knowledge such as a chair is a "touchable thing" is not represented here. Due to this restriction this ontology is referred to as a micro-theory.

Upper Ontology. Instead of modeling knowledge like a "barrier" is a "touchable thing" in micro-theories such common sense knowledge can be derived from a so called upper ontology. On the one hand application development becomes much easier drawing on an approved ontology, and on the other hand various services become scalable much easier, if they are using the same upper ontology. Several ontologies written in OWL have been developed to represent common sense knowledge such as the Wordnet [Fel98] or OpenCYC [Ope]. In this project, we decided in favor of the OpenCYC ontology because it offers widespread knowledge and especially good support for the implementation of "smart building" services.

For making common sense knowledge available to a "smart building" service the micro-theories and the upper ontology need to be connected. Therfore, concepts of a micro-theory can be integrated to the upper ontology using the OWL subclass property. A chair in the micro-theory, e.g., can be defined as a subclass of a OpenCYC seating device and hence a chair inherits all its characteristics like being touchable. In that manner, all other concepts of the micro-theory can be integrated into the upper ontology.

Context Acquisition. "Smart building" services are supposed to act depending on the building's context. To gain context information according to Dey [Dey01] a bridge between the system of the service and reality is needed. In this project, RFID sensors are considered for sensing the attendance of persons. As a working hypothesis, it is assumed that all relevant entities in a building such as persons are known to the formerly introduced ontology. Further it is assumed that each relevant entity is equipped with an unique so called RFID tag. This tag makes

entities visible to RFID sensors of other entities, and furthermore each tag can be identified unambiguously.

Connection to Semantic Web. As a next step the sensor data need to be connected to the ontology of a "smart building" service making it context-aware. For that reason all sensor data is collected using a central so called sensor proxy server. The core tasks of this server are

1. the evaluation of the raw sensor data depending on the sensor type and
2. the integration of the evaluated sensor data into the "smart building's" ontology.

The evaluation of raw sensor data depends on various characteristics such as the sensor type as well as the unit, the accuracy and the validity of data. Therefore, an ontology for sensors called OnSen was developed using OWL within this project. The ontology is inspired by the Physical Meta Language PML [MIT01] and by sensorML [UAH04]. Depending on the sensor specifications in OnSen, raw sensor data is processed and provided for the integration into the "smart building's" ontology.

Besides specifying a sensor's characteristics, OnSen allows to allocate a sensor and a "smart building" entity. Using that information, sensor data can be integrated easily into the "smart building's" ontology using OWL properties. A sensor attached to a conference room, e.g., can sense a RFID tag attached to a person located in that room. As contexts can change frequently, especially if people are walking, the context information needs to be refreshed often.

Data Retrieval using Queries. Having an up-to-data ontology an interface for accessing it is needed. A rather simple approach to query OWL ontologies is querying the underlying representation of OWL data encoded in RDF[W3C99] format combined with a reasoner.

The RDF format allows to specify relations between resources by triples. Each RDF triple consists of a subject, a predicate and an object. Using that triple encoding a relation between a chair and a seating device could be modeled in the following manner:

```
(smartbuilding:chair, owl:subclassOf, opencyc:seatingDevice)
```

Many approaches for querying RDF data discussed in [Mea02] such RDQL, RDFPath, TRIPLE or Versa have been developed. In this project, RDQL is used for specifying queries because its syntax is easy to learn because it is similar to the widespread SQL syntax. A query, e.g., selecting all subclasses of a seating device could be specified as follows:

```
SELECT ?subject
WHERE (?subject, owl:subclassOf, opencyc:seatingDevice)
```

Since `subclassOf` is a transitive property, the query could yield more subclasses like `canvas chair` being a subclass of `chair`. Due to limited memory capacity and/or limited computational power, the set of answers querying the RDF representation of an OWL ontology can be restricted by controlling the capabilities of the ontology reasoning. Instead of full OWL reasoning deriving all possible RDF triples, in many cases deriving the transitive closure of the subclass relation only is sufficient.

Complex Context. More complex context information that goes beyond the context level of one entity such as "There is a meeting in the conference room, if more than five persons are present." is needed for "smart building" services. Such context can be generated using a rule language as supported for instance by the Generic Reasoner of the Jena Semantic Web Framework, e.g., by the following rule:

```
(smartbldng:conference smartbldng:takesPlaceIn smartbldng:conferenceRoom ) <-
   (smartbldng:conferenceRoom smartbldg:personsInside ?persons),
   greaterThan(?persons, "5"))
```

4 Advantages of the Approach and Conclusion

The approach to the "smart building" perspective briefly introduced in this article has the following advantages:

1. It is easily deployable because of the ubiquity of the web especially in an office building.
2. It is rather generic because it relies on common software namely Semantic Web and Web Software.
3. As a consequence of its genericity the approach makes maintenance easy.
4. The approach will benefit of the ongoing improvements and developments of the Semantic Web especially because context modeling and context reasoning are concerned.

The approach has been fully implemented in the NTT Communication Science Labs in Japan. The communication with sensors has been simulated in a prototypic implementation.

5 Acknowledgments

This research has been funded by the European Commission and by the Swiss Federal Office for Education and Science within the 6th Framework Programme project REWERSE number 506779 (cf. http://rewerse.net).

References

[Dey01] Anind K. Dey. Understanding and using context. In *Personal and Ubiquitous Computing, Special issue on Situated Interaction and Ubiquitous Computing*, 2001.

[Fel98] Christiane Fellbaum, editor. *WordNet – An Electronic Lexical DB*. MIT Press, 1998.

[HP05] HP. *Jena Semantic Web Framework*, 2005. http://jena.sourceforge.net.

[Mea02] A. Magkanaraki and G. Karvounarakis et al. Ontology storage and querying. Technical report, Foundtion for Research and Technology Hellas, Institut of Computer Science, Information Systems Laboratory, April 2002.

[MIT01] MIT. *PML Core Specification 1.0*, 2001.

[NSD+01] N. F. Noy, M. Sintek, S. Decker, M. Crubezy, R. W. Fergerson, and A. Musen. Creating semantic web contents with protege-2000. *IEEE Intelligent Systems*, 16(2), 2001.

[Ope] *OpenCYC*. http://www.opencyc.org/doc/.

[Top01] TopicMaps.Org. *XML Topic Maps (XTM) 1.0*, 2001.

[UAH04] UAH Earth System Sc. Lab. *SensorML core*, 2004. http://vast.uah.edu/SensorML/.

[W3C99] W3C. *Resource Description Framework (RDF)*, 1999.

[W3C01] W3C. *DAML+OIL*, 2001.

[W3C04] W3C. *Web Ontology Language (OWL)*, 2004.

XML Perspectives on RDF Querying: Towards integrated Access to Data and Metadata on the Web

Tim Furche, François Bry, Oliver Bolzer
Institute for Informatics, University of Munich
http://www.pms.ifi.lmu.de

Abstract

The integral processing of data and metadata is starting to get recognized as a central challenge for the next decade (e.g. in Pat Selinger's ICDE 2005 Keynote) not only as part of realizing the Semantic Web vision, but also on a smaller scale as part of the next generation of desktop data management (cf. Apple's Spotlight and Microsoft's WinFS). In this article, we focus on metadata represented in the W3C's RDF formalism. We illustrate first steps towards integrating access to RDF metadata and access to standard Web data in XML format. For this, two XML views over RDF data are expressed in the query language Xcerpt and discussed. These views illustrate two different approaches for integrating RDF metadata processing and current data processing techniques.

1 Introduction

The "Semantic Web" is an endeavor widely publicized in [1], envisioning the current Web, which consists of (X)HTML and documents in other XML formats to be extended by metadata specifying the meaning of these documents in forms usable by both human beings and computers.

The integral processing of data and metadata is starting to get recognized as a central challenge for the next decade (e.g. in Pat Selinger's ICDE 2005 Keynote) not only as part of realizing the Semantic Web vision, but also on a smaller scale as part of the next generation of desktop data management (cf. Apple's Spotlight and Microsoft's WinFS that share the aim to extend current file storage and desktop search with extensive metadata facilities).

In the (Semantic) Web context, a number of formalisms have been proposed for representing metadata, in particular RDF, Topic Maps, and OWL. We concentrate on RDF as the most widely used. This article illustrates first steps towards integrating access to standard Web data in XML format and RDF metadata: First, as argued above, integrated access to standard Web data in XML and metadata in RDF is essential. A framework to access RDF data through XML views is proposed. Second, we argue that the currently predominant treatment of RDF data as flat triples is, although easy to comprehend, not the only and often not the best way of considering RDF data. Rather, a view of the RDF data directly as a graph is not only natural and closer to the RDF data model and allows for easy expression of graph patterns using much the same constructs as for navigating in XML data. This is particularly evident in face of incomplete information about the precise graph structure.

The proposed framework is realized by rules in the XML query language Xcerpt that allow (a) the easy conversion between the two views on RDF and (b) the "serialization transparent" querying of RDF, i.e., the querying of RDF in many of the over a dozen serialization formats for RDF proposed in recent years.

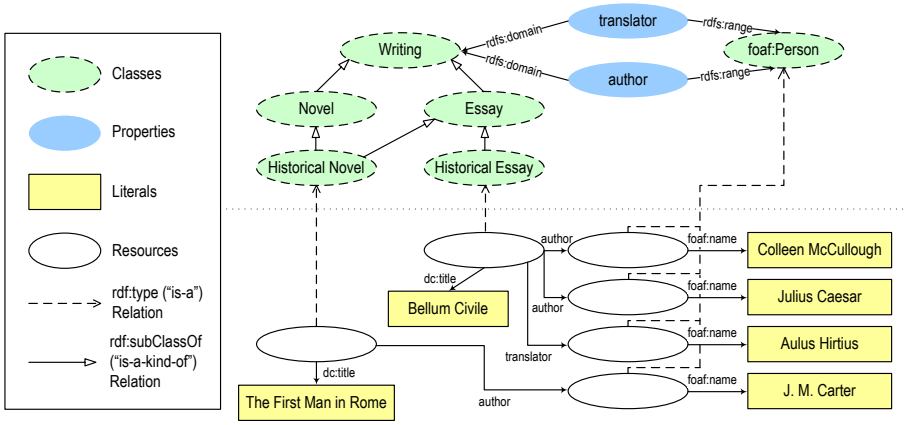


Figure 1: Sample Data: representation as a (simplified) RDF graph.

2 Preliminaries

2.1 RDF and RDF Schema: Metadata Representation in the Semantic Web

RDF [6] data is sets of "triples" or "statements" of the form (*Subject*, *Property*, *Object*). RDF data is commonly seen as a directed graph, whose nodes correspond to a statement's subject and object and whose arcs correspond to a statement's property (thus relating a subject with an object). Nodes (i.e. subjects and objects) are labeled by either (1) URIs describing (Web) resources, or (2) literals (i.e. scalar data such as strings or numbers), or (3) are unlabeled, being so-called anonymous or "blank nodes". Blank nodes are commonly used to group or "aggregate" properties. Edges (i.e. Properties) are always labeled by URIs indicating the type of relation between its subject and object.

RDFS allows one to define so-called "RDF Schemas" or "ontologies", similar to object-oriented data models. Based on an RDFS, "inference rules" can be specified, for instance the transitivity of the class hierarchy, or the type of an untyped resource that has a property associated with a known domain.

RDF can be *serialized* in various formats, the most frequent being XML. Early approaches to RDF serialisation have raised considerable criticism due to their complexity. As a consequence, a surprisingly large number of RDF serialisation have been proposed, cf. [3].

Figure 1 shows the running example for this article, a (simplified) representation of an RDF graph as used, e.g., in a book recommender system.

2.2 Xcerpt, a versatile Web Query Language

Xcerpt [9] is a query language designed after principles given in [4] for querying both data on the "standard Web" (e.g., XML and HTML data) and data on the Semantic Web (e.g., RDF, Topic Maps, etc. data).

Xcerpt is "data versatile", i.e. a same Xcerpt query can access and generate, as answers, data in different Web formats. Xcerpt is "strongly answer-closed", i.e. it not only gives rise to construct answers in the same data formats as the data queries, but also to include in a query program data generated by this same query program. Xcerpt's queries are pattern-based and give rise to incompletely specify the data to retrieve by (1) not explicitly specifying all children of an element, (2) specifying descendant elements at indefinite depths (restrictions in the form of regular path expressions being possible), and (3) specifying optional query parts. Xcerpt's evaluation of incomplete queries is based on a novel form algorithm called "simulation unification". Xcerpt's processing of XML documents is graph-oriented, i.e., aware of the reference

mechanisms (e.g., ID/IDREF attributes and links) of XML. Xcerpt is rule-based: An Xcerpt rule expresses how data queried can be re-assembled into new data items.

3 Two Perspectives on RDF

This section presents two different perspectives on RDF: (1) a flat, almost relational view and (2) a graph view reminiscent of semi-structured data. These perspectives are compared briefly with some existing approaches for RDF querying.

To illustrate these two perspectives, the selection query "Select all *Essays* together with their *authors* (i.e. author URIs and corresponding names)" is used against the data of Figure 1.

3.1 RDF Triples: A Flat, Relational View

The following Xcerpt program expresses the above query on a triple view of the RDF data:

```
1  DECLARE ns-prefix rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   DECLARE ns-prefix books = "http://example.org/books#"
3  GOAL
     result [
5      all essay [
         id [ var Essay ],
7        all author [
           id [ var Author ],
9          all name [ var AuthorName ]
     ]  ]  ]
11 FROM
     and{ RDFS-TRIPLE [
13       var Essay, rdf:type{{}}, books:Essay{{}}            ],
       RDF-TRIPLE [
15       var Essay, books:author{{}}, var Author             ],
       RDF-TRIPLE [
17       var Author, books:authorName{{}}, var AuthorName    ]  }
   END
```

The query pattern (between `FROM` and `END`) is a conjunction of queries against the RDF triples represented in the predicate `RDF-TRIPLE` using the prefixes declared in line 1 and 2. Notice that the first conjunct actually uses `RDFS-TRIPLE`. This view of the RDF data contains all basic triples plus the ones entailed by the RDFS semantics (cf. [2] for a detailed description). Using `RDFS-TRIPLE` instead of `RDF-TRIPLE` ensures that also resources actually classified in a sub-class of `books:Essay` are returned.

In the construct pattern (between `GOAL` and `FROM`), one of the strengths of combining XML and RDF querying in Xcerpt is shown: Following the W3C's requirements for an RDF data access language, yet in contrast to most other RDF query languages, it is possible to construct arbitrary XML: E.g., here, a list of all essays with their authors grouped inside is constructed.

Except for the construction of arbitrary XML, a similar (triple) view of RDF is taken in most of the current RDF query languages, most notably in RDQL and the W3C's SPARQL [7], and also in [8], an approach for querying RDF with XQuery: A query is composed of conjunctions (and in some languages including our proposal disjunctions) of "triple patterns", i.e., triples with variables indicating queried data. Using multiple occurrences of same variables more complex conditions can be expressed, e.g., for traversing paths in the RDF data or even for restricting a resource using several of its properties. While familiar from SQL, this style leads for RDF data to hard-to-read and lengthy queries that also pose problems for evaluation (cf., e.g., [5]).

The previous observations lead us to an alternative view of RDF that is both closer to its actual data model and can make better use of the advanced features of an XML query language such as the traversal of arbitrary length paths in tree or graph data.

3.2 RDF Graph: A Semi-structured View

For this view of RDF, Xcerpt's treatment of XML as graph data is an advantage over XML query languages such as XPath or XQuery, which consider XML as strictly tree shaped, providing no direct support for (ID/IDREF or similar) links in the data model. Although there have been proposals for slicing an (acyclic) RDF graph into trees for processing them with XSLT or XQuery (e.g., [11]), these approaches invariantly suffer (a) from choosing an appropriate slicing and (b) from the (in general) exponential blow-up of the tree view of an acyclic RDF graph.

In Xcerpt, a graph view of RDF is rather natural as the following Xcerpt program expressing the same query as above, but on the graph instead of the triple view, demonstrates:

```
  ... % prefixes and construction identical to above query
FROM
  RDFS-GRAPH {{
    var Essay {{
      rdf:type {{ books:Essay {{ }} }},
      books:author {{
        var Author {{ books:name {{ var AuthorName }} }}
      }}
    }} }}
END
```

The RDF graph view is represented in the `RDF-GRAPH` predicate. Here, the `RDFS-GRAPH` view is used that extends `RDF-GRAPH` as `RDFS-TRIPLE` extends `RDF-TRIPLE`. Triples are represented similar to striped RDF/XML: each resource is a direct child element in `RDF-GRAPH` with a sub-element for each statement with that resource as object. The sub-element is labeled with the URI of the predicate and contains the object of the statement. As Xcerpt's data model is a rooted *graph* this can be represented without duplication of resources.

In contrast to the previous query against the RDF triple view, no conjunction is used but rather a nested pattern that naturally reflects the structure of the RDF graph. The more complex a query gets, the more evident the advantage of the graph view becomes: instead of having to use multiple occurrences of same variables for relating parts of the query, that relation is represented in the structure of the query itself (represented in the textual version of the query shown above by nesting and indentation).

Path traversals of arbitrary length can be expressed using traversal operators such as descendant. E.g., to find all subclasses of a given class one can use Xcerpt's qualified descendant `desc(rdfs:subClassOf<rdfs:Class)*` that is similar to regular path expressions or conditional XPath. Similarly, other constructs for querying XML data with incomplete information about the structure of the queried data can be used for RDF as well.

Considering the efficient evaluation of queries against such a graph view of RDF data, there are results on the efficient evaluation of queries against graph-shaped semi-structured data, cf. [10]. Ongoing work by the authors targets efficient evaluation methods for implementing Xcerpt queries against graph-shaped data. We believe it likely that at least for some interesting subsets of Xcerpt queries efficient evaluation methods against graph-shaped data can be found.

3.2.1 A "Retrospective" View: From Triples To Graphs

A final observation on the graph view is that it can not only be implemented directly on the different RDF serializations (just as the triple view) but also on top of the triple view, as shown in the following rule:

```
CONSTRUCT
  RDF-GRAPH {
    all var Subject @ var Subject:var SubjectType {
      all optional var Predicate {  ^var Object   },
      all optional var Predicate {  var Literal   }
    } }
FROM
  or{
```

```
    RDF-TRIPLE[
10    var Subject,  var Predicate:uri{},
      optional var Literal as literal{{}},
12    optional var Object {{}} where { var Object != 'literal'}
    ],
14  RDF-TRIPLE[
      /.*/:/.*/{{}},  /.*/:/.*/{{}},  var Subject{{}}
16  ]  }
END
```

Notice the use of the `optional` keyword in lines 11 and 12. This indicates that the contained part of the pattern does not have to occur in the data, but if it does occur the contained variables are bound appropriately. In lines 3 and 4 the actual graph structure is constructed: by using the operators `@` and `^` a (possibly cyclic) link can be constructed.

3.2.2 Serialisation Transparency

Aside of providing the above discussed two views on RDF, Xcerpt's rules are also convenient for making the language "serialisation transparent". For each RDF serialisation, a set of rules expresses a translation from or into that serialisation. They can be found in [2], similar functions for parsing RDF/XML in XQuery are described in [8].

4 Conclusion and Outlook

In this article, a brief overview of a framework for RDF querying in the XML query language Xcerpt is presented highlighting in particular the need for reconsideration of the triple view as the only perspective on RDF available in the established RDF query languages. We believe that a richer view of RDF more akin to XML data with graph-shape not only makes the integration of data and metadata easier but also leads in many cases to more succinct queries without sacrificing efficiency.

Acknowledgments. This research has been funded by the European Commission and by the Swiss Federal Office for Education and Science within the 6th Framework Programme project REWERSE number 506779 (cf. `http://rewerse.net`).

References

[1] T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, 2001.

[2] O. Bolzer. Towards Data-Integration on the Semantic Web: Querying RDF with Xcerpt. Diplomarbeit/Master thesis, University of Munich, 2005.

[3] F. Bry, T. Furche, L. Badea, C. Koch, S. Schaffert, and S. Berger. Identification of Design Principles for a (Semantic) Web Query Language. Deliverable I4-D1, REWERSE, 2004.

[4] F. Bry, T. Furche, L. Badea, C. Koch, S. Schaffert, and S. Berger. Querying the Web Reconsidered: Design Principles for Versatile Web Query Languages. *J. of Semantic Web and Inf. Sys.*, 1(2), 2005.

[5] E. Hung, Y. Deng, and V. S. Subrahmanian. RDF Aggregate Queries and Views. In *ICDE*, 2005.

[6] G. Klyne, J. Carroll, and B. McBride. *Resource Description Framework (RDF).* W3C, 2004.

[7] E. Prud'hommeaux and A. Seaborne. *SPARQL Query Language for RDF.* W3C, 2005.

[8] J. Robie. The Syntactic Web: Syntax and Semantics on the Web. In *XML*, 2001.

[9] S. Schaffert and F. Bry. Querying the Web Reconsidered: A Practical Introduction to Xcerpt. In *Extreme Markup Languages*, 2004.

[10] R. Schenkel, A. Theobald, and G. Weikum. HOPI: An Efficient Connection Index for Complex XML Document Collections. In *EDBT*, 2004.

[11] N. Walsh. RDF Twig: Accessing RDF Graphs in XSLT. In *Extreme Markup Languages*, 2003.

PALADIN: A Pattern-based Approach to Large-scale Dynamic Information Integration

Jürgen Göres
Heterogeneous Information Systems Group
University of Kaiserslautern
goeres@informatik.uni-kl.de

Abstract: The nascent data grid technology aims to make access to globally distributed structured and semi-structured databases possible. However, only integrated, transparent access to such data sources, abstracting not only from distribution and implementation details, but also from the diverse forms of heterogeneity, allows maximum leverage for the envisioned dynamic forms of inter-organisational cooperation. Traditional integration technologies rely on a human-driven development process and are therefore not applicable to the ad-hoc style of integration required in these scenarios. PALADIN aims to automate integration planning by using patterns that capture expert knowledge in a machine-processable way. A pattern describes a generic problem constellation encountered in information integration and provides a guideline to its solution, described as transformations on a graph-oriented representation of the data source schemas and their data.

1 Introduction

The IT infrastructure found in most organizations today often grew organically over decades. It is therefore characterised by several forms of heterogeneity, which can rougly be classified into three categories: *Technical heterogeneity* subsumes different hardware, operating systems, programming languages and APIs. *Logical heterogeneity* includes different data models (data model heterogeneity), the diverging use of identical datamodels (schematic heterogeneity) and different structuring (structural heterogeneity). *Semantic heterogeneity* originates from the often subtle discrepancies in the use and understanding of terms and concepts in schemas of different origin.

Existing information integration (II) technology is used successfully to consolidate such systems by providing a unified view that abstracts from these forms of heterogeneity. However, setting up an integration system is to date to a large part still a manual task that can be split into distinct phases: Initially, requirements for the desired integrated system, like the target data model and schema, are determined (*analysis phase*). Based on these results, suitable data sources that can contribute have to be found (*discovery phase*). In the *planning phase* the forms of heterogeneity are identified and resolved by defining a mapping between the source and the target schemas. This mapping or *integration plan* is then implemented by deploying it to a suitable runtime environment like one of the existing II products (*deployment phase*). The resulting system is then available for end users (*runtime phase*). This classical approach to II relies on human expertise in two areas: Integration experts have profound knowledge about how to remedy logical heterogeneity by defining mappings between the source and target schemas. To successfully define such mappings, an understanding of the domain of discourse is needed. Application domain experts are required to resolve the non-trivial semantic gaps between the different schemas, i.e., they deliver insight into the domain concepts and provide information about correspondencies between the different schemas. This elaborate process, while time-consuming and costly, is still adequate for scenarios where the requirements and the data sources deemed for integration are stable. However, it is difficult to apply to dynamic environments like the *virtual organisations* (VO) proposed by [3], a new form of short- to medium-term inter-organisational cooperations, that use grid technology to pool computing resources and data on a global scale.

PALADIN (PAttern-based LArge-scale Dynamic INformation integration) aims to reduce the dependency on human experts. Building upon existing and evolving middleware technology like web and grid services [5] and standard interfaces like those specified by the *Data Access and Integration Working Group* (DAIS) [1] that solve most aspects of the aforementioned technical heterogeneity, we

focus on resolving logical and semantic heterogeneity. Machine-processable *integration patterns* are used to capture II expertise. They enable the collection of both generic problem constellations encountered when integrating heterogeneous data sources as well as guidelines to their solution. If the problem described by a pattern is later discovered in the schemas of a set of data sources chosen for integration, the generic solution provided by the pattern is adapted to the concrete situation. By combining a set of patterns from a pattern library that cover data model, schematic and structural heterogeneity, a proper sequence of schema (and accompanying data) transformations can be discovered that describes a mapping from the chosen data sources to the desired target schema.

Like the integration experts they support or substitute, patterns cannot rely on schema information (i.e. the explicit metadata) alone, but also have to consider the semantics of the data source. Therefore, a medium to convey the semantics in a machine-processable way is needed. In practice, schema matching techniques are applied to identify those elements in the different schemas that correspond in some way. Many approaches to automatic schema matching have been explored (see [6] for an overview). These methods are usually based on a lexical and structural analysis of the schemas. While the results are often promising, they still lack the quality to solely rely on them for automated integration. PALADIN uses *domain schemas* to augment basic schema matching with domain knowledge. Being essentially multi-language ontologies, they describe the terms of discourse of a given domain, the relationships between these terms and their synonyms and optionally translations into different languages. However, even with domain schemas as support, the current state-of-the-art cannot yield schema matches with sufficient quality. Assuming that the user of a data grid is familiar with the application domain, we explicitely include him in the schema matching process by providing an interface that allows to check, correct and amend the correspondencies identified by automated matching.

The remainder of this paper is structured as follows: Section 2 briefly describes the PALADIN metamodel that is capable of handling arbitrary data and metadata. Section 3 describes the machine-processable integration patterns in more detail. Domain schemas are discussed briefly in section 4. Section 5 concludes with a summary and an outlook on future work.

2 Metamodel

To seamlessly handle data and metadata from data sources with different data models, the PALADIN infrastructure has to provide a uniform internal representation for the diverse data-model- or even implementation-specific schema description languages like DTDs, XML Schema or SQL DDL in its many proprietary variants. An existing approach is OMG's *Common Warehouse Metamodel* [2]: It uses a stack of four meta-layers, numbered from M0 to M3, where every element on one layer is an instance of an element on the layer above, and itself forms a class for instances on the layer below: On the M3 layer, the *Meta Object Facility* (MOF) serves as meta-metamodel to describe arbitrary metamodels on the M2 layer. Here, CWM and its extensions already provide metamodels for many standard data models like SQL and XML. One the M1 or model layer, actual schemas are described by instantiating the elements of these metamodels. The data layer M0 finally represents the data stored in these schemas.

While at first glance the CWM appears to provide the ideal infrastructure for PALADIN, certain deficits restrict its use for our purposes: Besides a high degree of complexity, which makes implementation and use cumbersome, the current CWM specification is somewhat outdated. For example, the predefined metamodel for XML is limited to the expressiveness of DTDs, thus making an extension to include XML Schema necessary. This can easily be accomplished using the MOF, but still does not address the fundamental flaw of CWM: the lack of a proper M0 or data layer. CWM provides an instance *meta*model, which places the actual data on the M1 layer, i.e., on the same conceptual layer as their metadata. This not only violates the strict concept of distinct meta layers, but also results in an overboarding complexity of the instance metamodel, making it unsuitable to describe more than sample data. However PALADIN requires the M0 layer to describe an essential part of integration pat-

terns. Therefore, we provide the PALADIN Metamodel (PMM), which borrows CWM's general concept of meta layers and many of its established terms, but remedies its deficits, like providing a conceptionally sound data layer. Using the meta-metamodel, we define those metamodels that are the most relevant, namely XML, object-oriented and object-relational. Further metamodels can easily be added if the need arises. In addition to the metamodels that are used to describe actual source or target schemas, a Match metamodel allows the description of correspondencies within and between arbitrary schemas and across different metamodels which are identified during schema matching. Unlike the generic correspondencies supported by existing schema matchers, which usually only allow to describe the fact that there is some kind of relationship between the elements with a given confidence, we provide an extensible set of correspondencies of a finer granularity that carry more precise semantics, like part-component-relationships, sub- or supersets and many more.

3 Integration Patterns

Integration patterns are PALADIN's fundamental concept to describe human II expertise in a machine-understandable way. Each pattern describes a certain elementary or complex integration problem, i.e., a constellation of schema elements and the correspondencies between them, and supplies a guideline to solve this problem by applying a suitable transformation on both the schemas involved and on the data held in these schemas. Instead of using imperative algorithms to capture this kind of II experience, patterns use a declarative notation which allows the easy extension of the library of available patterns without programming effort. A language is required that offers sufficient expressiveness to describe all kinds of structural, schematic and data model transformations. Many existing languages come to mind: SQL views can transform relational schemas and data, recent extensions of the standard like SQL/XML even allow bridging from SQL to XML. However, SQL is unable to bridge the gap between data and metadata (i.e. solve schematic heterogeneity). Approaches like SchemaSQL [4] try to remedy this situation, but have not gained widespread support. XQuery offers sufficient expressiveness to turn data to metadata and vice versa, but is limited to the XML realm alone. Therefore, neither language is suitable to express the full range of patterns. Defining our own textual language to cope with this problem would not only add to the existing plethora of proprietary data modelling languages that usually more or less imitate and build on SQL, but would also make the extension of PALADIN with additional metamodels difficult, as it would likely require new language constructs. We therefore suggest an approach that understands any schema (and its data) described in the PALADIN metamodel as a typed, attributed multigraph and every operation on the schema and data as a graph transformation. Graph transformations are a well explored concept (see [7]). To represent integration patterns, we chose an approach that is based on the semi-graphical language defined for the PROGRES graph replacement system [8]. While the graphical elements provide an easily readable, yet semantically precise description of the most relevant aspects of a pattern, the textual notation adds expressiveness that would be hard to capture graphically. A graph transformation is described using production rules. A production rule has a left-hand side (LHS) that describes the situation it can be applied to as an abstract subgraph, and a right-hand side (RHS) that describes the result of its application.

Figure 1 shows a simplified sample pattern that describes the necessary steps when source and target schema feature a different degree of normalisation. It also introduces the most important elements of the graphical part of the notation. Every pattern consists of two production rules or *facets*, which are situated on the M1 and on the M0 layer, respectively. The M1 facet describes the effects on the schema: The left-hand side describes a situation, where two tables are connected via a foreign key between a subset of their columns.

To connect left- and right-hand side, every element that should be preserved by the pattern is bound to a variable. An element that is not repeated on the RHS indicates a deletion, an element that is new on the RHS indicates the creation of an element. In the example, the two tables bound to *t1* and

t2 are replaced by a new single table *t3* which receives all columns that were part of either *t1* or *t2*, except those that were part of the foreign key, which are equivalent and are therefore added to *t3* only once. All inbound context edges to a node in a pattern are preserved if the node itself is preserved, or can be rerouted to another node using the diamond symbol. The M0 facet describes the conversion of the data stored in the schema elements of the M1 facet. It essentially describes an equi-join between the two tables and uses a simple expression of the textual notation for the join condition.

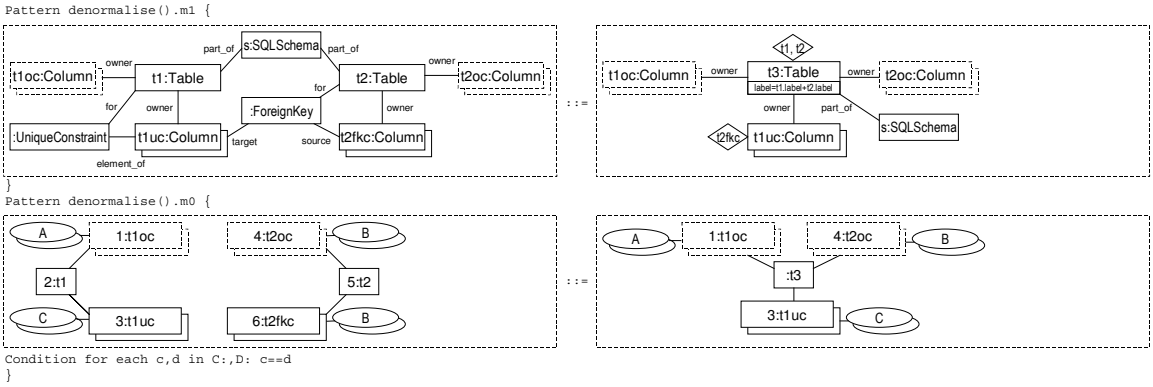


Fig. 1. An integration pattern expressed as a graph transformation

Given a sufficiently large library of patterns, the task of integration planning can now be understood as proving a hypothesis (the target schema) from a set of axoims (the source schemas) using a set of rules (i.e. the patterns). Any successful deduction, i.e., a sequence of rule or pattern applications that can be found, describes a logical query graph that transforms the source data into the desired target schema, with each pattern's M0 facet representing an operator in this graph. To use this integration plan, it has to be translated into a physical plan using the operators provided by the selected runtime environment (RE). While a RE could use the M0 facet directly to execute the query by working on the M0 graph representation of the data, many patterns will not need the additional expressiveness and only use operations that are already part of an algebra for the respective data model. In this cases, it is often more efficient to use a native implementation of an operator if it is provided by the runtime environment, using the graph-oriented operator description only as a fallback, e.g., to configure a generic graph-transformation operator. The same holds true for the specification of the patterns: while the use of graph transformations allows the expression of arbitrary operations on schemas and their data in a declarative way, existing query languages are often sufficient and we therefore support the translation of patterns in a data model specific language into our internal format.

4 Domain Schemas

While the pattern shown in figure 1 relies solely on schema information, many patterns require information about the semantics of a data source. This is especially relevant when a given schema constellation allows the application of several schematically equivalent patterns. Semantic information discovered or defined during schema matching is stored using the Match metamodel and can be used in the definition of the preconditions for patterns. To improve the degree of automation achievable by schema matching techniques, thereby reducing the amount of user interaction to rectify and complete these matches, we introduce the concept of domain schemas as our "Swiss army knife to semantic integration". A domain schema can in principle be modeled using an arbitrary data model and represents a base of reference for a given domain. We provide a simplified variant of the CWM object model, that allows each class to have multiple labels in different languages, which enables their use as a dictionary and acronym list for lexical schema matching techniques. The backbone of each domain schema is the directed acyclic graph (DAG) created by the (multiple-)inheritance relationships between its classes, providing a natural broader/narrower-term relationship which can be used both as a thesaurus and as a means to address subdomains with path expressions. Additional relationships and

special attributes can be used to add further information, like hints to potential references (e.g. foreign keys that were not explicitely defined in a schema) and strong or weak candidate keys. While it is possible to provide monolithic schema matchers that make use of domain schemas, they integrate smoothly with the concept of integation patterns: Figure 2 shows how graph transformations can be used to transfer information from a domain schema to a concrete schema. The *is_a* relationship between two classes *c1* and *c2* in the domain schema *ds* is annotated using an *is_a* match between two model elements *me1* and *me2* that carry the same label as *c1* and *c2*, respectively. Note that since a domain schema class can have multiple labels, the equals-operator used in the condition is interpreted as an existential quantifier.

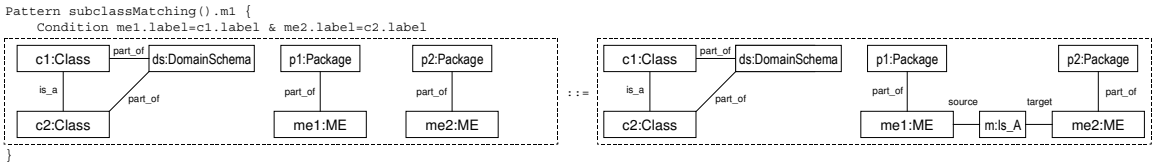


Fig. 2. Using patterns and domain schemas for schema matching

Since this kind of match is independent of the concrete data model of the matched schema elements, the pattern refers to the abstract *Package* and *ME* (ModelElement) classes of the core metamodel of PMM, which are superclasses for every schema or schema element of a concrete metamodel.

5 Conclusion and Future Work

We have introduced two fundamental concepts, integration patterns and domain schemas, to provide an automated approach to information integration with the two essential types of machine-processable knowledge and experience. We currently evaluate existing technologies like the PROGRES graph transformation engine introduced in [8] for their suitability and also analyse different implementation alternatives for creating a new system from scratch. While focussing on the planning phase of the II process, we can transfer both the concept of patterns and domain schemas to other phases. Domain schemas can be used in the discovery process to narrow down the potentially very large set of candidate data sources to the most promising ones. Patterns can also guide deployment by describing the mapping from logical integration plans to the physical plans, i.e., the deployment info for different runtime environments.

References

[1] Mario Antonioletti, Malcolm Atkinson, Susan Malaika, Simon Laws, Norman W. Paton, Dave Pearson, Greg Riccardi: *The Grid Data Service Specification*. September 19, 2003.

[2] Object Management Group: *Common Warehouse Metamodel (CWM) Specification Version 1.1*. March 2003.

[3] Ian Foster, Carl Kesselman, Steven Tuecke: *The Anatomy of the Grid - Enabling Scalable Virtual Organizations*. In Proceedings of the First IEEE International Symposium on Cluster Computing and the Grid, May 15-18, 2001.

[4] Laks V. S. Lakshmanan, Fereidoon Sadri, Iyer N. Subramanian: *SchemaSQL – A Language for Interoperability in Relational Multi-Database Systems.* In Proceedings of the 22nd VLDB Conference, pp. 239-250, 1996.

[5] S. Tuecke, K. Czajkowski, I. Foster, J. Frey, S. Graham, C. Kesselman, T. Maguire, T. Sandholm, D. Snelling, P. Vanderbilt: *Open Grid Services Infrastructure*. June 27, 2003.

[6] Erhard Rahm, Philip A. Bernstein: *A survey of approaches to automatic schema matching*. In The VLDB Journal 10, pp. 334-350, 2001.

[7] Andy Schürr: *Programmed Graph Replacement Systems*. 1997.

[8] A. Schürr, A. W. Winter, A. Zündorf: *The PROGRES Approach: Language and Environment*. In Handbook of Graph Grammars and Computing by Graph Transformation 1, pp. 487-550, 1997.

Top-N Anfrageverarbeitung in PDMS: Anforderungen und Lösungswege

Katja Hose
katja.hose@tu-ilmenau.de
Fakultät für Informatik und Automatisierung, TU Ilmenau
Postfach 100565, D-98684 Ilmenau

Zusammenfassung

Da P2P-Netzwerke gewöhnlich aus einer großen Anzahl von Peers bestehen und somit große Datenmengen verfügbar sind, ist es oft unnötig wenn nicht sogar unmöglich, dem Nutzer ein vollständiges Ergebnis seiner Anfrage durch Einbeziehung aller teilnehmenden Peers zu präsentieren. Vielmehr muss in P2P-Systemen begründet durch deren Charakteristik zumeist auf best-effort Lösungen zurückgegriffen werden. Ein Beispiel dafür stellen Top-N-Anfragen dar. Als Ergebnis wird ein gemäß der Ähnlichkeit zur gestellten Anfrage sortiertes Ergebnis erwartet, das die „besten“ zum gegebenen Zeitpunkt im Netzwerk vorhandenen Datenwerte widerspiegelt, ohne dabei eine kostenintensive Berechnung vorauszusetzen. Die dabei gestellten Anforderungen an die Anfrageverarbeitung umfassen zumeist eine möglichst geringe Bearbeitungsdauer sowie eine möglichst hohe Korrektheit des Ergebnisses bezüglich aller im Netzwerk vorhandenen Daten. In der vorliegenden Arbeit wird zum Einen vorgestellt, welche Anforderungen an eine Strategie, die Top-N-Anfragen bearbeitet, zu stellen sind. Zum Anderen werden konkrete Lösungsansätze vorgestellt, die die Bearbeitung derartiger Anfragen auf effiziente Art und Weise realisieren.

1 Einführung

Seit einigen Jahren werden schemabasierte Peer-to-Peer (P2P) Systeme im Rahmen aktueller Forschung untersucht. Im Gegensatz zu klassischen Systemen bestehen P2P Systeme, auch Peer Data Management System (PDMS) genannt, aus einer Menge von Peers, welche vollkommen unabhängig voneinander sind. Dies bedeutet unter Anderem, dass es sich um eine dynamische Netzwerkstruktur handelt, in der sich Peers unabhängig voneinander an- und abmelden bzw. ausfallen können. Zudem sind auch lokales Schema sowie die lokal verwalteten Daten selbst keinen Abhängigkeiten zu anderen Peers unterworfen.

Dadurch dass sich PDMS meist durch eine große Anzahl von Peers auszeichnen, ist eine vollständige und exakte Anfrageverarbeitung oftmals nicht möglich, da eine solche Bearbeitung zum Einen das Einbeziehen aller Peers, d.h. das Fluten des Netzwerkes, voraussetzen würde. Zum Anderen stellen unvollständige bzw. inkorrekte Mappings, sowie unvollständige Informationen über Datenverteilung und -lokalisation weitere erschwerende Aspekte dar. Deshalb wird oft auf *best-effort* Lösungen zurückgegriffen, die versuchen, eine Anfrage so exakt und kostenminimal wie möglich zu beantworten, jedoch ein vollständiges Ergebnis nicht garantieren können.

Dies spielt insbesondere bei der Berechnung von Top-N-Anfragen eine wichtige Rolle. Das Ziel dabei ist nicht ein vollständiges und exaktes Ergebnis zu ermitteln, sondern vielmehr *möglichst* alle N besten Ergebnistupel auf effiziente Weise zu finden, wobei natürlich ein Kompromiss zwischen Genauigkeit bzw. Vollständigkeit und Effizienz bzw. Kostenminimierung gefunden werden muss. Die Schwierigkeit besteht vor Allem darin, Kriterien bzw. Regeln zu entwickeln, anhand derer entschieden werden kann, welche Peers nicht in die Anfrageverarbeitung einbezogen werden, ohne dass die Genauigkeit d.h. die Qualität des Ergebnisses zu große Einbußen erfährt. Da in PDMS kein globales Wissen zur Verfügung steht, stellt dies besondere Anforderungen an die verwendeten Anfragebearbeitungsstrategien.

Nach der Vorstellung verwandter Arbeiten in Abschnitt 2 werden in Abschnitt 3 einige grundlegende Ansätze bzw. Strategien der Top-N-Anfrageverarbeitung vorgestellt, ihre Evaluierung folgt in Abschnitt 4. Abschließend umreißt Abschnitt 5 kurz, welche Aspekte und Optimierungen grundlegender Verfahren in aktueller und zukünftiger Forschungsarbeit untersucht werden.

2 Verwandte Arbeiten

Erste Ansätze zur Berechnung von Top-N-Anfragen wurden für die Anwendung in RDBMS konzipiert. Wie in [CG99, BCG02] vorgestellt, erscheint es zunächst sinnvoll, existierende Datenstrukturen wie zum Beispiel Indexe und Statistiken, beispielsweise in Form von Histogrammen, zu verwenden, um eine Top-N-Anfrage in eine traditionelle Select-Anfrage zu übersetzen und diese dann auf übliche Weise zu bearbeiten. Ein anderer Ansatz, der versucht, das Risiko eines Neustarts einer Anfrage auf probabilistische Weise zu quantifizieren, wird in [DR99] vorgestellt.

TPUT [CW04] ist ein Algorithmus, der für die effiziente Berechnung von Top-N-Anfragen auf Aggregaten in verteilten Systemen entwickelt wurde. Dieser Algorithmus stellt eine Weiterentwicklung des *Threshold Algorithmus (TA)* dar, welcher unabhängig von mehreren Gruppen entwickelt wurde [NR99, GBK00, LNF01]. Weitere Varianten wurden entworfen für Multimedia Repositories [CGM04], Distributed Preference Queries auf internetzugänglichen Datenbanken [MBG04] und für das Ranken von Anfrageergebnissen strukturierter Datenbanken [ACDG03]. Eine weitere Arbeit [TWS04] basierend auf TA führt eine Familie von approximativen Top-N Algorithmen ein, die auf probabilistische Art und Weise den Berechnungsaufwand zu minimieren versuchen. All diese Algorithmen benötigen mehrere Round-Trips, um das Endergebnis zu ermitteln, was im allgemeinen Fall als nachteilig angesehen werden muss.

Die in [NSTB04, BNST05] vorgestellten Algorithmen versuchen, die Bearbeitung von Top-N-Anfragen zu verbesseren, indem sie dynamisch Anfragestatistiken sammeln, die beim erneuten Bearbeiten der gleichen Anfrage verwendet werden können, um das Anfragerouting zu verbessern. Wird eine Anfrage ein erstes Mal gestellt, müssen jedoch alle Peers am Prozess der Anfrageverarbeitung teilnehmen, wobei mehrere Round-Trips zum Ermitteln des Endergebnisses benötigt werden.

3 Basisstrategien

Im Gegensatz zu einigen der im vorhergehenden Abschnitt vorgestellten Verfahren basieren die im Folgenden vorgestellten Strategien darauf, dass jeweils nur ein Durchlauf bzw. Round-Trip benötigt wird, um eine Anfrage vollständig zu beantworten. Dies bedeutet, dass jeder Peer eine Anfrage nur ein einziges Mal erhält und niemals mehrfach bezüglich der Bearbeitung einer einzigen Anfrage kontaktiert wird. Dies bringt unter Anderem die folgenden zwei Vorteile mit sich: erstens werden Kosten reduziert, und zweitens werden Probleme vermieden, die sich durch die Tatsache begründen, dass in PDMS aufgrund der Dynamik nicht garantiert werden kann, dass sich das Netzwerk im zweiten Durchlauf noch immer im gleichen Zustand befindet (An- und Abmeldevorgänge, Ausfälle, Updates,...). Zur Vereinfachung, ohne Beschränkung der Allgemeinheit, wird dabei in den folgenden Abschnitten die zugrunde liegende Netzwerkstruktur als Baum aufgefasst. Die vorgestellten Strategien können jeweils durch kleinere Modifikationen zur Zyklenerkennung erweitert werden.

Naiver Ansatz Die einfachste Strategie zur Beantwortung einer Top-N-Anfrage besteht darin, alle Peers in die Anfrageverarbeitung einzubeziehen, d.h. das Netzwerk zu fluten. Dabei bearbeitet jeder Peer die Anfrage lokal und sendet jeweils seine N lokal besten Ergebniselemente zum Initiator. Dieser berechnet nach Erhalt der Antworten das Endergebnis und gibt es als globales Top-N-Ergebnis aus.

Partieller Ansatz Eine Möglichkeit, diesen naiven Ansatz zu verbessern, besteht darin, die Tatsache auszunutzen, dass die Antworten in einem Netzwerk mehrere Peers durchlaufen müssen, bevor sie den Initiator letztendlich erreichen. Auf dem Weg zum Initiator fasst jeder Peer die Daten seiner Nachbarn zusammen, berechnet daraus ein Teilergebnis mit N Elementen und leitet dieses dann weiter. Schließlich verfährt der Initiator auf gleiche Weise und verfügt somit über das Endergebnis, das dem User präsentiert wird. Offensichtlich reduziert diese Strategie das zu sendende Datenvolumen, jedoch nicht die Anzahl involvierter Peers, da das Netzwerk immer noch geflutet werden muss, um alle Peers zu erreichen.

Global optimierender Ansatz Um zusätzlich auch die Anzahl der am Anfrageverarbeitungsprozess teilnehmenden Peers reduzieren zu können, müssen Indexstrukturen existieren, die Auskunft darüber geben, über welche Daten ein Nachbar verfügt bzw. auf welche Daten über einen Nachbarn zugegriffen werden kann. Ist eine Indexstruktur mit globalem Wissen, zum Beispiel auf Basis von DHTs, vorhanden

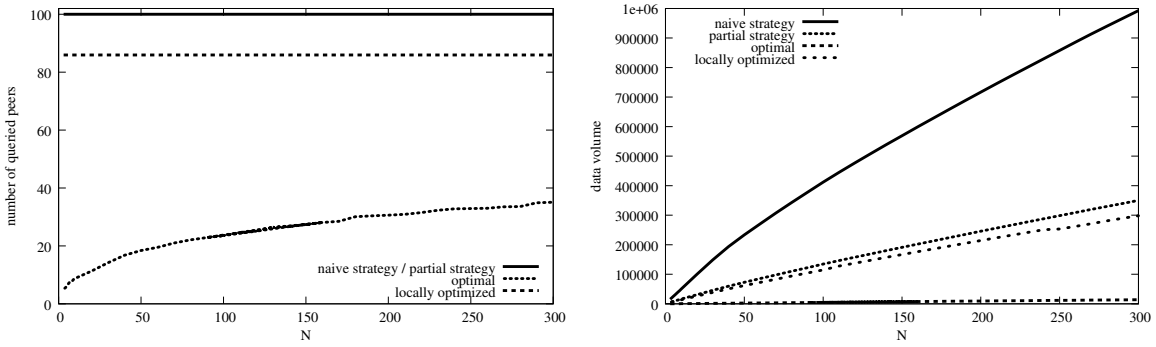
und auf dem angefragten Attribut definiert, so kann der Initiator die Anfrageverarbeitung insbesondere das Routing der Anfrage zentral planen, mittels des vorhandenen globalen Wissens optimieren und somit die Bearbeitungskosten reduzieren.

Lokal optimierender Ansatz Ist hingegen kein globales Wissen vorhanden, so muss jeder Peer, der die Anfrage bearbeitet, die Entscheidung über ein möglichst optimales Routing lokal treffen. Dies geschieht wiederum auf Basis von Indexen, wobei davon ausgegangen wird, dass die Informationen dieser Indexe innerhalb eines bestimmten Horizontes beschränkt sind und somit keine Informationen über Daten enthalten, die in sehr großer Entfernung liegen. Aufgrund der ihm vorliegenden Indexe entscheidet jeder Peer, der die Anfrage erhält, individuell welche seiner Nachbarn über relevante Daten bezüglich der gestellten Anfrage verfügen, leitet die Anfrage an diese Peers weiter und bezieht sie in die Anfrageverarbeitung ein. Dieser im Folgenden als lokal optimierend bezeichnete Ansatz basiert auf dem partiellen Ansatz, um gleichzeitig das Datenvolumen zu reduzieren.

4 Evaluierung

In diesem Abschnitt sollen die grundlegenden Performanceunterschiede der in Abschnitt 3 vorgestellten Basisstrategien aufgezeigt werden. Alle umgesetzten Algorithmen wurden im gleichen statischen Netzwerk bestehend aus 100 Peers ausgewertet. Wie bereits erwähnt, wurde eine Baumstruktur als Netzwerk gewählt, wobei jeder Peer ausgenommen der Blattknoten 4 Kinder besitzt. Als Testdaten wurden Astrodaten [Kha01] verwendet, wobei sich alle Anfragen zur Vereinfachung auf eine Dimension, d.h. ein Attribut beschränken. Ein Teil der Daten wurde geclustert auf die Peers verteilt, so dass jeder Peer viele Daten in einem bzw. auch mehreren Teilbereichen des gesamten Attributwertebereichs besitzt. Ein weiterer Teil wurde zufällig auf alle Peers verteilt. Bezüglich jeder Anfragestrategie wurden zehn Anfragen mit jeweils unterschiedlichen Attributwerten am gleichen Peer initiiert. Die im Folgenden dargestellten Kurven repräsentieren die Mittelung zwischen den Ergebnissen dieser zehn Anfragen, die den gesamten Wertebereich des angefragten Attributes überdecken.

Der Vergleich der Strategien erfolgt auf Basis von Datenvolumen und Anzahl angefragter Peers. Auf die Betrachtung der Anzahl zur Bearbeitung einer Anfrage benötigter Nachrichten wird an dieser Stelle aufgrund der starken Korrelation zur Anzahl involvierter Peers verzichtet. Der Grund für diese Korrelation besteht darin, dass alle umgesetzten Strategien darauf basieren, dass ein Peer auf die Antworten seiner Nachbarn wartet, bevor er eine eigene sendet. Für den lokal optimierenden Ansatz wurde eine Indexstruktur auf Basis von Histogrammen verwendet, deren Horizont stets alle Peers beinhaltet. (In der gewählten Netzwerkstruktur beträgt die maximale Entfernung eines beliebigen Peers vom Initiator 4)



(a) Anzahl involvierter Peers

(b) Benötigtes Datenvolumen

Abbildung 1: Betrachtungen in statischen Netzwerken

Abbildung 1 zeigt das grundsätzliche Verhalten von naivem, partiellem und lokal optimierendem Ansatz in statischen Netzen. Die als „optimal“ bezeichneten Kurven sollen lediglich zu Vergleichszwecken dienen. Sie stellen minimale Werte dar, die nur dann erreicht werden können, wenn (i) eine Anfrage auf optimalem Weg nur zu genau denjenigen Peers geleitet wird, die tatsächlich über die N besten Datensätze im ganzen Netzwerk verfügen, und (ii) diese Peers auch nur genau diejenigen Elemente zurückliefern,

die zum exakten Endergebnis gehören. Der global optimierende Ansatz wurde nicht weiter verfolgt, da der Fokus unserer Arbeit auf PDMS und den einhergehenden Problemstellungen der Dynamik und begrenztem Wissen liegt.

Abbildung 1(a) zeigt, wie sich die Anzahl der an der Anfrageverarbeitung teilnehmenden Peers bei steigendem N verhält: Bei naiver und partieller Strategie werden stets alle 100 Peers des Netzwerkes angefragt. Bei der lokal optimierenden Strategie fällt auf, dass immer gleichbleibend viele Peers involviert werden. Dies ist vor Allem dadurch begründet, dass stets alle Peers einbezogen werden müssen, die über Daten verfügen, die *eventuell* zu den N global besten gehören, d.h. Datensätze mit Attributwerten nahe dem angefragten Wert. Begründet durch die Datenverteilung zeigen sich bei veränderlichem N keine Unterschiede.

Das benötigte Datenvolumen, welches in Abbildung 1(b) dargestellt ist, bestätigt die Erwartung, dass die lokal optimierende Strategie weniger Datenvolumen benötigt als der partielle Ansatz. Dies lässt sich anhand der Tatsache begründen, dass die lokal optimierende Strategie weniger Peers in den Anfrageprozess involviert, siehe Abbildung 1(a), und somit auch weniger Datenvolumen zur Anfrageverarbeitung nötig ist. Im Gegensatz zur naiven Strategie, welche alle Daten zunächst am Initiator sammelt und deshalb das größte Datenvolumen verursacht, versendet jeder Peer bei Anwendung des lokal optimierenden oder partiellen Ansatzes maximal N Elemente als Antwort.

Zusammenfassend lässt sich sagen, dass der lokal optimierende Ansatz im Vergleich zur naiven bzw. partiellen Strategie weniger Kosten (Datenvolumen bzw. Peeranzahl) verursacht. Wie in weiteren hier aus Platzgründen nicht aufgeführten Tests gezeigt werden konnte, hat weder Datenverteilung noch Netzwerkstruktur Einfluss auf diese grundsätzliche Feststellung. Die lokal optimierende Strategie arbeitet in statischen Netzwerken derart, dass stets das exakte globale Top-N-Ergebnis zurückgeliefert wird. Dabei werden oftmals viele Peers angefragt, die zwar dem angefragten Schema entsprechende Daten besitzen, letztendlich jedoch nichts zum Endergebnis beitragen können. Der Grund dafür ist die Tatsache, dass Indexe stets eine Approximation der Realität darstellen und der Algorithmus dies durch das *zusätzliche* Anfragen weiterer Peers ausgleicht und so Fehler vermeidet. Folglich muss eine Strategie, deren Kosten niedriger als die der „lokal optimierenden“ Strategie sein sollen, d.h. sich denen als „optimal“ bezeichneten Kosten annäheren, „bewusst“ das Risiko eingehen, dass das Ergebnis nicht alle N besten Elemente des Netzwerkes widerspiegelt. Nur dann können Kostenreduktionen durch das Nichtanfragen von Peers erzielt werden. Dies stellt neben weiteren im nächsten Abschnitt vorgestellten Aspekten einen Schwerpunkt unserer momentanen Forschungsarbeit dar.

5 Ausblick

Wie in den vorhergehenden Abschnitten bereits aufgezeigt, ist es das Ziel weiterer Forschung, best-effort Lösungen zu entwickeln, die es erlauben, die Anzahl involvierter Peers zu reduzieren. Eine Möglichkeit, den dabei eingegangenen Fehler zu quantifizieren, stellen Wahrscheinlichkeitsgarantien dar. Zum Beispiel könnte ein Nutzer beim Stellen einer Anfrage eine Fehlergrenze angeben, die es dem zugrunde liegenden Algorithmus ermöglicht, Fehler einzugehen. Ein Anfrageergebnis könnte somit in folgender Form angegeben werden: „mit einer Wahrscheinlichkeit von 90% entspricht das ermittelte Ergebnis dem global exakten“.

Im Gegensatz zu den hier untersuchten Netzwerken zeichnen sich reale P2P-Systeme meist durch ein hohes Maß an Dynamik aus. Dieser wichtige Aspekt stellt besondere Anforderungen an Algorithmen, die in solchen Netzwerken eingesetzt werden sollen. Durch eine inkrementelle Arbeitsweise, die prinzipiell bei allen vorgestellten Basisstrategien realisiert werden kann, wird zwar u.U. Datenvolumen und Nachrichtenanzahl erhöht, jedoch sind diese Varianten robuster gegenüber dynamischen Veränderungen der Netzwerkstruktur [KHS04]. Im Vergleich der einzelnen inkrementellen Strategievarianten untereinander würde sich jedoch das in Abschnitt 4 aufgezeigte Kostenverhältnis nicht ändern.

Insbesondere die Tatsache, dass Indexe zumeist nur auf einen gewissen Horizont beschränkt sind, stellt einen interessanten in zukünftiger Arbeit zu untersuchenden Aspekt dar. Gleiches gilt für die Untersuchung der Korrektheit des Ergebnisses unter Einwirkung von verschiedenen Indexaktualisierungsstrategien (Query-Feedback, epidemische Protokolle, Gossiping...).

Allgemeine Einsparungen von Datenvolumen könnten durch das Mitsenden von lokalen Ergebnisinformationen erzielt werden. Dadurch dass jeder Peer, der eine Anfrage weiterleitet, Informationen darüber mitsendet, welche Ergebniswerte ihm bereits bekannt sind, können die Empfänger-Peers ihre Antworten dahingehend optimieren, dass sie nur solche Datenelemente enthalten, die *besser* sind als die bereits bekannten.

Aus Platzgründen können an dieser Stelle nicht alle Aspekte ausführlich erläutert werden. Zusammenfassend lässt sich sagen, dass sich unsere momentane Forschungsarbeit damit beschäftigt, eine Anfragestrategie zu entwickeln, welche die folgenden Charakteristika erfüllt: inkrementelle Arbeitsweise, Beantwortung einer Anfrage innerhalb eines Round-Trips, Nutzung vorhandener Indexstrukturen, Kostenminimierung unter Angabe von probabilistischen Garantien für die Vollständigkeit bzw. Korrektheit des Ergebnisses.

Literatur

[ACDG03] Sanjay Agrawal, Surajit Chaudhuri, Gautam Das, and Aristides Gionis. Automated ranking of database query results. In *CIDR*, 2003.

[BCG02] N. Bruno, S. Chaudhuri, and L. Gravano. Top-k selection queries over relational databases: Mapping strategies and performance evaluation. *ACM TODS, Vol. 27, No. 2*, 2002.

[BNST05] Wolf-Tilo Balke, Wolfgang Nejdl, Wolf Siberski, and Uwe Thaden. Progressive distributed top k retrieval in peer-to-peer networks. In *ICDE'05*, 2005.

[CG99] Surajit Chaudhuri and Luis Gravano. Evaluating top-k selection queries. In *VLDB'99*, pages 397–410, 1999.

[CGM04] Surajit Chaudhuri, Luis Gravano, and Amélie Marian. Optimizing queries over multimedia repositories. *IEEE TKDE*, 16(8):992–1009, 2004.

[CW04] Pei Cao and Zhe Wang. Efficient top-k query calculation in distributed networks. In *PODC '04*, pages 206–215, 2004.

[DR99] Donko Donjerkovic and Raghu Ramakrishnan. Probabilistic optimization of top n queries. In *VLDB'99*, pages 411–422, 1999.

[GBK00] Ulrich Güntzer, Wolf-Tilo Balke, and W. Kiessling. Optimizing multi-feature queries for image databases. In *VLDB 2000*, pages 419–428, 2000.

[Kha01] N. V. Kharchenko. All-sky compiled catalogue of 2.5 million stars (ascc-2.5), 2001. Kinematics and Physics of Celestial Bodies, 17, N 5, 409 - 423; ftp://ftp.mao.kiev.ua/pub/astro/cc.

[KHS04] M. Karnstedt, K. Hose, and K.-U. Sattler. Query Routing and Processing in Schema-Based P2P Systems. In *DEXA'04 Workshops*, pages 544–548. IEEE Computer Society, 2004.

[LNF01] Amnon Lotem, Moni Naor, and Ronald Fagin. Optimal aggregation algorithms for middleware. In *PODS'01*, March 03 2001.

[MBG04] Amélie Marian, Nicolas Bruno, and Luis Gravano. Evaluating top- queries over web-accessible databases. *ACM TODS'04*, 29(2):319–362, 2004.

[NR99] S. Nepal and M. V. Ramakrishna. Query processing issues in image (multimedia) databases. In *ICDE '99*, page 22, 1999.

[NSTB04] Wolfgang Nejdl, Wolf Siberski, Uwe Thaden, and Wolf-Tilo Balke. Top-k Query Evaluation for Schema-Based Peer-to-Peer Networks. In *Int. Semantic Web Conf.*, pages 137–151, 2004.

[TWS04] Martin Theobald, Gerhard Weikum, and Ralf Schenkel. Top-k query evaluation with probabilistic guarantees. In *VLDB 2004*, pages 648–659, 2004.

Transformation von SQL in XQuery-Anfragen innerhalb föderierter Informationssysteme

Heiko Jahnkuhn, Ilvio Bruder, Ammar S. Balouch
Institut für Informatik, Universität Rostock
E-Mail: {hj016,ilr,ab006}@informatik.uni-rostock.de

In föderierten Informationssystemen steht man vor dem Problem, Daten und Informationen, die in verschiedenen Formaten und Systemen gespeichert sind, auswerten zu müssen. Die Daten können dabei in relationalen, objektrelationalen oder auch XML-Datenbanken vorliegen. Dazu werden Anfrage-Transformationen benötigt, aber auch die Transformation von Daten und Ergebnissen in andere Formate ist erforderlich. Diese Arbeit beschäftigt sich mit der Anfragetransformation von SQL in XQuery innerhalb föderierter Informationssysteme. Der in dieser Arbeit entwickelte allgemeingültige Algorithmus führt eine solche Transformation unter bestimmten Vorbedingungen automatisch durch.

1 Einführung

Ein grundlegendes Prinzip der föderierten Datenbank- und Informationssysteme besteht darin, dass eventuell mehrere unterschiedliche Datenmodelle in die Föderation eingehen. Somit besteht die Möglichkeit, dass Föderierungsdienst und Teilnehmer der Föderation dementsprechend heterogen sein können. Wenn eine Anfrage in einer konkreten Anfragesprache gestellt wird, aber auch von anderen Datenbanksystemen mit anderen Anfragesprachen ausgewertet werden soll, muss eine Transformation dieser Anfrage erfolgen. Dabei sind für diese Arbeit prinzipiell zwei Szenarien denkbar. Im ersten Fall nimmt ein objektrelationales Datenbanksystem an einer Föderation teil, dessen Datenmodell auf XML beruht. Somit ist die Anfragesprache des Föderierungsdienstes XQuery. Wenn nun der Teilnehmer eine Anfrage bezüglich seines lokalen Schemas stellt, welche aber von der Föderation ausgewertet werden soll, muss diese Anfrage derart transformiert werden, dass sie in XQuery vorliegt und sich auf das globale Schema bezieht. Im zweiten Fall, auf den sich der hier vorgestellte Algorithmus auch bezieht, nutzt der Föderierungsdienst ein objektrelationales Datenmodell, auf das Anfragen mittels SQL gestellt werden. Wird nun eine SQL-Anfrage bezüglich des globalen Schemas gestellt, muss diese für jeden Teilnehmer der Föderation transformiert werden. Für eine XML-Datenbank muss somit die Anfrage in XQuery transformiert werden, welche eine Anfrage auf ein XML-Dokument oder eine XML-Dokumentenkollektion beschreibt. Nutzt man für eine derartige Transformation beispielsweise ein regelbasiertes System, besteht der Nachteil darin, dass ein solches System für jeden XML-Teilnehmer neu entworfen werden muss. Nutzt dieser Teilnehmer nicht nur eine sondern eine Vielzahl unterschiedlicher Schemabeschreibungen, steigt der Entwurfsaufwand sogar linear mit der Anzahl der Schemabeschreibungen, die der Föderation bereitgestellt werden. Um diesem Effekt entgegenzuwirken wird hier ein Algorithmus vorgestellt, welcher eine solche Transformation automatisch auch bezüglich verschiedener Schemabeschreibungen durchführen kann. An dieser Stelle wird allerdings nur der grundlegende Algorithmus vorgestellt, der sich auf einfache SQL-Anfragen beschränkt. Einfach bedeutet an dieser Stelle, dass für die Where- und Having-Klausel lediglich elementare Vergleichsoperatoren wie <, > und = erlaubt sind, welche mit den logischen Operatoren AND, OR und NOT verknüpft werden können. In der Having-Klausel sind weiterhin Aggregatfunktionen möglich. Auf die Umsetzung weiterer Konstrukte, die SQL bietet, sowie weitere Phasen und auftretende Probleme der Anfrageverarbeitung in föderierten Informationssystemen wird in [Jah05] genauer eingegangen.

2 Anfragetransformation

An dieser Stelle wird die Anfragetransformation bezüglich einer bestimmten Schemabeschreibung (DTD oder XML Schema) durchgeführt. Das bedeutet, dass die resultierende XQuery-Anfrage an eine

Dokumentenkollektion gestellt wird, die bezüglich dieser Schemabeschreibung gültig ist. Auf den allgemeinen Fall, dass einer XML-Datenbank mehrere Schemabeschreibungen zugrunde liegen, wird ebenfalls in [JBB05] eingegangen. Es existiert somit eine injektive Abbildung der Attribute des globalen Schemas auf eine Menge von Pfadausdrücken bezüglich des lokalen Schemas, welches eine Schemabeschreibung des XML-Komponentendatenbanksystems. Der Algorithmus zur Transformation läuft in vier separaten Schritten ab, die nachfolgend erklärt werden sollen.

Schritt 1
Zu Beginn wird die globale Anfrage syntaktisch analysiert. Sie wird in die Fragmente der SELECT-, FROM, WHERE-, GROUP BY-, HAVING- und ORDER BY-Klausel aufgeteilt. Unter Nutzung einer Zuordnungstabelle von globalen Attributen auf lokale Pfadausdrücke wird eine Liste sämtlicher Pfadausdrücke erstellt, welche in der globalen SQL-Anfrage nach Substitution der globalen Attribute auftreten. Diese Liste beschreibt den Strukturbaum der zugrunde liegenden Anfrage, welcher einen Teilbaum der zugrunde liegenden Schemabeschreibung darstellt. Darauf hin wird die WHERE-Klausel derart analysiert, dass sie in eine Liste von Konjunktionstermen aufgespaltet wird. Besteht die WHERE-Klausel also aus dem Term A AND B, wobei A und B wiederum Terme sind, wird dieser Term aus der Liste gelöscht und A und B separat in diese Liste aufgenommen. Anschließend werden A und B auf dieselbe Art untersucht, so dass die Liste am Ende nur atomare Vergleiche oder Disjunktionen enthält. Anschließend wird für jedes Element dieser Liste der größte gemeinsame Pfad (gcp – greatest common path) ermittelt, der im Term T_i nach Substitution der globalen Attribute enthalten ist. Somit ist jedem Term T_i ein $gcp(T_i)$ zugeordnet. Im letzten Teil dieses Transformationsschrittes wird der ermittelte Strukturbaum der Anfrage mit drei Knotenarten markiert, den Where-Elementen, den Return-Elementen und den Splitting-Knoten. Die Where-Elemente repräsentieren dabei die ermittelten $gcp(T_i)$. Die Return-Elemente ergeben sich aus den Attributen der SELECT-, GROUP BY-, HAVING- und ORDER BY-Klausel. Die Splitting-Knoten lassen sich durch die konkreten Beziehungen zwischen den WHERE- und RETURN-Elementen ermitteln. Abbildung 1 zeigt dabei die möglichen Beziehungsklassen, aus denen sich Splitting-Knoten ermitteln lassen. XML-Attribute werden hierbei ebenfalls als XML-Elemente angesehen. Alle anderen Beziehungen lassen

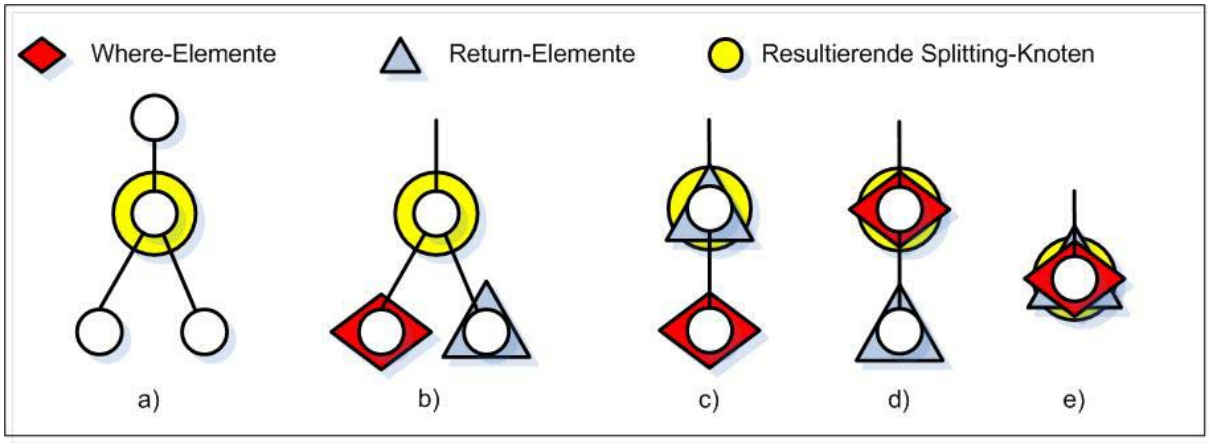


Abb. 1: Beziehungsklassen für Splitting-Knoten

sich durch diese Klassen darstellen. Die Linien repräsentieren dabei unbekannte Pfade, auf denen keine markierten Knoten liegen. Die Bedeutung der einzelnen Klassen lautet wie folgt, wobei SK für den resultierenden Splitting-Knoten steht:

a) SK ist der erste Knoten, der mehr als ein Kindelement hat.
b) SK hat sowohl Where- als auch Return-Elemente in verschiedenen descendent-Achsen.
c) SK ist ein Return-Element und hat ein Where-Element in einer descendent-Achse.
d) SK ist ein Where-Element und hat ein Return-Element in einer descendent-Achse.
e) SK ist sowohl Where- als auch Return-Element.

Schritt 2
Im zweiten Transformationsschritt wird aus dem markierten Strukturbaum ein XQuery-Ausdruck hergeleitet, welcher diesen repräsentiert. Jeder Splitting-Knoten des Strukturbaums entspricht dabei einem zu generierenden FLWR-Ausdruck, wobei die Hierarchie erhalten bleibt. Somit wird ein Top-Down-Verfahren eingesetzt, welches ausgehend von der Wurzel des Baumes den ersten Splitting-Knoten ermittelt. Für diesen Splitting-Knoten wird ein FLWR-Ausdruck erzeugt, der den Splitting-Knoten als Kontextknoten in der For-Klausel referenziert. Die Where-Klausel wird durch die Konjunktion sämtlicher Where-Elemente gebildet, die in der self-or-descendent-Achse des Kontextknotens liegen. Für die Erzeugung der Return-Klausel werden drei Fälle unterschieden:

1. Ist der Splitting-Knoten auch ein Return-Element wird der Kontextknoten direkt übernommen.
2. Für jeden Folgepfad, der ausschließlich Return-Elemente enthält, wird der Pfad dieser Return-Elemente übernommen.
3. Für jeden Folgepfad, der Splitting-Knoten enthält, wird dieses Verfahren rekursiv angewandt, wobei der jeweils neue Kontextknoten immer relativ zum übergeordneten Kontextknoten gesetzt wird.

Im ersten und zweiten Fall werden die Return-Elemente zusätzlich mit Tags umschlossen, deren Name sich aus dem korrespondierenden globalen Attributnamen des jeweiligen Pfadausdrucks ergibt. Außerdem wird der Inhalt der ersten Return-Klausel mit zwei tuple-Tags umschlossen, so dass das

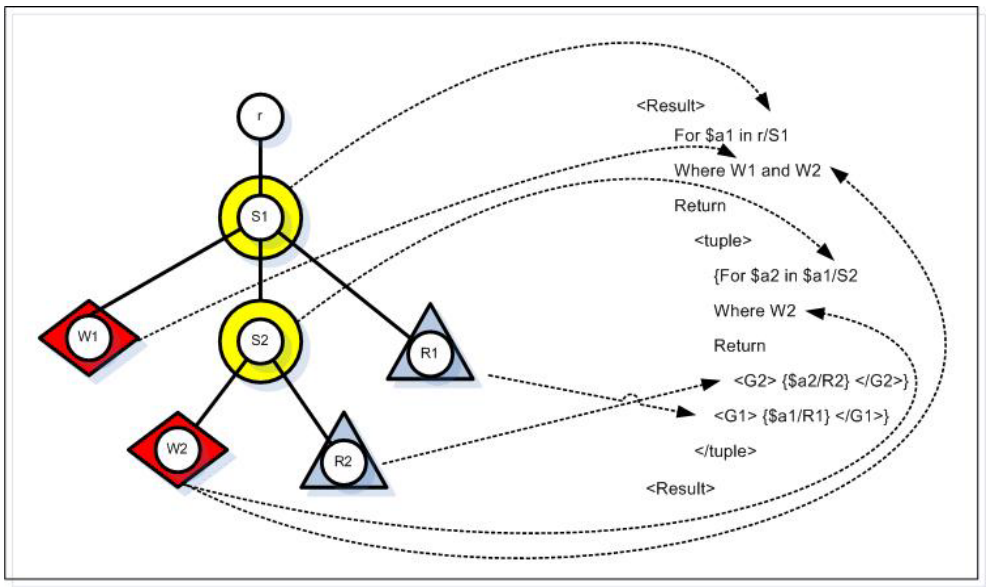


Abb. 2: Abbildung des Strukturbaumes auf XQuery-Anfrage

Gesamtergebnis als SET OF TUPLE OF dargestellt wird. In Abbildung 2 wird versucht diesen Schritt graphisch darzustellen. Die Ausgabe dieses zweiten Schrittes realisiert interessanterweise bereits eine relationale Darstellung des Strukturbaums, da die erzeugte Struktur gültig bezüglich folgender generischer DTD wäre:

<! ELEMENT Result (tuple*)>
<! ELEMENT tuple = any-node*>
(any-node bezeichnet hier ein beliebiges Element des Anfragebaums).

Schritt 3
Im dritten Schritt folgt die Realisierung der GROUP BY- und HAVING-Klausel, sofern diese in der globalen Anfrage verwendet wurden. Diese beiden Klauseln werden in diesem Schritt jedoch separat bearbeitet, da unter Umständen trotz GROUP BY- keine HAVING-Klausel existiert. Das Grundprinzip der Gruppierung basiert wiederum auf einem FLWR-Ausdruck. Die For-Klausel besteht dabei aus n Zuweisungen der Form $\$groupBy_i$ in result/tuple/Att_i, wobei n die Anzahl der Gruppierungsattribute und Att_i ein bestimmtes Gruppierungsattribut darstellen. Dadurch wird ein n-

dimenstionaler Vektorraum aufgespannt, der durch die For-Klausel sequentiell durchlaufen wird. In der Let-Klausel wird nun jedes tuple-Element einem bestimmten Punkt in diesem Raum zugeordnet und in der Return-Klausel einem konkreten group-Element zugewiesen. Dieses group-Element enthält zum einen die n Gruppierungsattribute und zum anderen eine Menge von tuple-Elementen, in denen die jeweiligen Attribute mit den Gruppierungsattributen identisch sind. Die Ergebnisstruktur dieses Teils des dritten Schrittes ist somit gültig bezüglich folgender DTD:

```
<! ELEMENT result (group* | tuple*)>
<! ELEMENT group (any-node*, tuple*)>
<! ELEMENT tuple = any-node*>
```

Auf diese Struktur kann nun die Selektion bezüglich der Having-Klausel durchgeführt werden. Dies geschieht ebenfalls mittels einem FLWR-Ausdruck, in der die group-Elemente sequentiell durchlaufen werden. Die Where-Klausel ergibt sich dabei direkt aus der globalen HAVING-Klausel. Es muss dabei lediglich jedem globalen Attribut der Pfad /tuple/ vorangestellt werden, da die Attribute nun mit dem absoluten Pfad Result/group/tuple/Attribut erreichbar sind. In die Return-Klauseln werden die group-Elemente übernommen, die sich durch die HAVING-Klausel qualifiziert haben. Abbildung 3 stellt diesen Schritt graphisch dar.

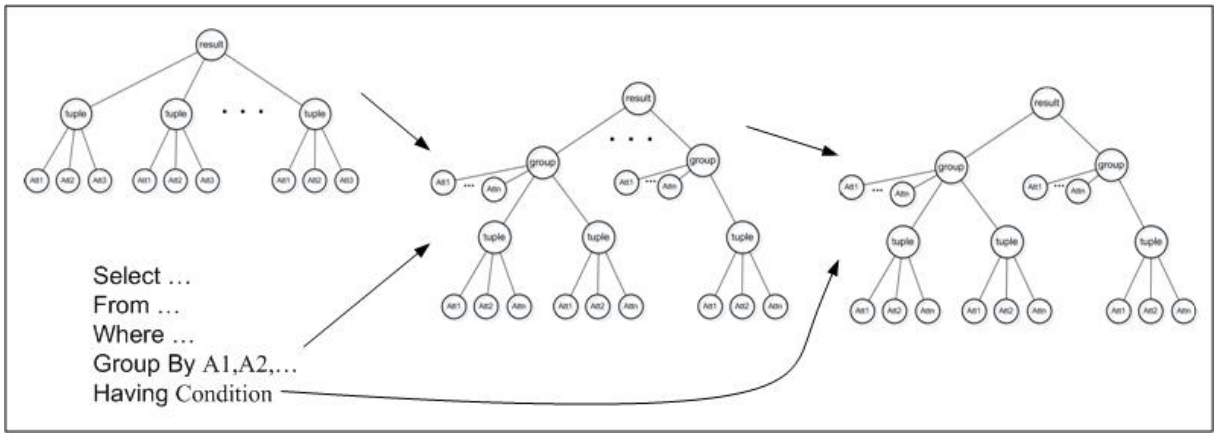


Abb. 3: Umsetzung der Group By- und Having-Klausel

Schritt 4

Im vierten und letzten Transformationsschritt werden die Attribute der globalen SELECT-Klausel berechnet und die eventuell spezifizierte ORDER BY-Klausel umgesetzt. Dies geschieht ebenfalls durch einen neuen FLWR-Ausdruck. Um festzustellen, worauf sich die Zuweisung in der For-Klausel bezieht, müssen prinzipiell drei Fälle unterschieden werden:

1. Die globale SELECT-Klausel enthält nur Aggregatfunktionen und keine Attribute. Falls eine Gruppierung existiert, beziehen sich die Aggregatfunktionen auf das group-Element des Ergebnisses, andernfalls auf das result-Element, also die ganze Ergebnismenge. In beiden Fällen jedoch auf das Vaterelement eines tuple-Elements.
2. Die globale SELECT-Klausel enthält nur Attribute und keine Aggregatfunktion. Falls eine Gruppierung existiert, müssen die Attribute in der GROUP BY-Klausel spezifiziert sein, somit beziehen sie sich auf das group-Element. Ohne GROUP BY-Klausel beziehen sich die Attribute auf jedes einzelne tuple-Element. In beiden Fällen erfolgt der Bezug jedoch auf alle direkten Kindelemente des result-Elements.
3. Die globale SELECT-Klausel enthält sowohl Attribute als auch Aggregatfunktionen. Daraus folgt, dass eine GROUP BY-Klausel spezifiziert wurde und zwar bezüglich der Attribute, die in der SELECT-Klausel als direkte Attribute vorkommen. Daraus folgt weiterhin, dass diese Attribute in allen tuple-Elementen der Gruppierung identisch sind. Somit ist dieser Fall analog zum zweiten Fall zu behandeln.

Die Order By-Klausel in diesem FLWR-Ausdruck entspricht der ORDER BY-Klausel der globalen Anfrage. Die textuelle Repräsentation wäre somit ORDER BY /{Sortierungsattribut(e)} ASCENDING | DESCENDING.

Die Return-Klausel besteht aus den Aggregatfunktionen und den Attributen der globalen SELECT-Klausel. Die Aggregatfunktionen haben dabei die Form AggFkt(/tuple/*Attribut*). Die Pfadnamen der Attribute lauten /*Attribut*. Diese werden jeweils mit zwei Tags umschlossen. Der Name dieser Tags ist entweder die Aggregatfunkion auf das Attribut angewandt, der globale Attributname oder der jeweilige, eventuell spezifizierte Aliasname. Der gesamte Inhalt der Return-Klausel wird ebenfalls wieder mit den tuple-Tags umschlossen.

Das Ergebnis des vierten Schrittes bildet den Abschluss der Anfragetransformation. Diese generierte Anfrage kann nun an das jeweilige XML-Datenbanksystem zusammen mit der Zugehörigen Schemabeschreibung, auf die sich diese Transformation bezog, geschickt werden. Die Ergebnismenge, die der Föderierungsdienst enthält liegt dann in einer XML-Form vor, die ebenfalls äquivalent zur SET OF TUPLE OF – Darstellung ist. Somit stellt eine Transformation der Ergebnismenge in das globale Datenmodell keine weitere Schwierigkeit mehr dar. Für weitere Schemabeschreibungen, die das XML-Komponentendatenbanksystem der Föderation eventuell bereitstellt, wird dieser Algorithmus jeweils separat wiederholt. Auf die darauf aufbauende Phase der Nachbearbeitung der Einzelergebnisse wird ebenfalls in [Jah05] genauer eingegangen.

3 Zusammenfassung und Ausblick

Das Ziel des hier vorgestellten Algorithmus ist eine Transformation einer globalen SQL-Anfrage in einen äquivalenten XQuery-Ausdruck, der bezüglich einer XML-Dokumentenkollektion an ein XML-Komponentendatenbanksystem übergeben werden kann. Vorraussetzung dafür ist lediglich eine semantikerhaltende Abbildung der lokalen Schemabeschreibung auf das globale Schema des Föderierungsdienstes. Ausgehend von dieser Abbildung und der aktuellen SQL-Anfrage wird der korrespondierende XQuery-Ausdruck abgeleitet. Der Algorithmus ist in der hier vorgestellten Form allerdings nur auf sehr einfach strukturierte Anfragen beschränkt. Die Umsetzung weiterer Konstrukte wie den Like-Operator oder verzahnt geschachtelte Anfragen wird in [Jah05] untersucht. Auch die Anwendung auf SQL-Erweiterungen ist denkbar. Eine in diesem Kontext interessante SQL-Erweiterung ist beispielsweise MM/Text. Dadurch ermöglichte Information Retrieval-Techniken, die Vergleiche mit Stammwortreduktion oder distanzbasierte Anfragen ermöglichen, könnten womöglich durch die Volltext-Erweiterung von XQuery ebenfalls durch diesen Algorithmus umgesetzt werden. Ein anderer Ansatzpunkt für diese Arbeit wäre beispielsweise ein mögliches Pipelining der Transformation zu untersuchen. Da die Einzelschritte völlig unabhängig vom vorangegangenen Schritt sind, sollte eine solche Parallelisierung verschiedener Anfragetransformationen bezüglich unterschiedlicher Schemabeschreibungen prinzipiell Möglich sein. Beispielsweise wäre der Einsatz von drei Prozessoren denkbar, wobei der dritte und der vierte Transformationsschritt von einem Prozessor bearbeitet werden muss. Da der dritte Schritt unter Umständen nicht durchgeführt wird, bestünde sonst die Gefahr, dass eine Transformation ohne GROUP BY-Klausel eine Transformation mit GROUP BY-Klausel "einholen" könnte.

Bibliography

[Con97] Stefan Conrad. *Föderierte Datenbanksysteme*. Springer, 1997

[EEL04] Francisco J.C. Escobar, Enrique D. Espinosa, Rafael Lozano. *XML Information Retrieval Using SQL2Xquery*. Departmento do Computación, Tecnológico de Monterrey-Campus cd. De México

[ERS99] Ahmed Elmagarmid, Marek Rusinkiewicz, Amit Sheth. *Management of Heterogeneous and Autonomous Database Systems.* Morgen Kaufmann Publishers, Inc., 1999

[Jah05] Heiko Jahnkuhn. *Transformation zwischen SQL- und XQuery-Anfragen innerhalb föderierter Informationssysteme.* Studienarbeit, Universität Rostock, 2005

[KM03] Meike Klettke, Holger Meyer. *XML und Datenbanken*. dpunkt Verlag GmbH, 2003

Modellierung mathematischer und kausaler Maßzahlen-Beziehungen in der multidimensionalen Datenanalyse

Sascha Koch
Oldenburger Forschungs- und Entwicklungsinstitut
für Informatik-Werkzeuge und -Systeme (OFFIS)
koch@offis.de

Zusammenfassung

Unterschiedliche Organisationen setzen zunehmend auf organisationsweites Performance Management zur strategischen Steuerung. Dabei wird angestrebt, die Strategie zu operationalisieren und die Leistung (Performance) der Organisation gezielt zu beeinflussen. Zugrunde gelegt werden vermutete Ursache-Wirkungsbeziehungen zwischen strategischen Zielen. Zudem werden die einzelnen Ziele mit Indikatoren verknüpft, um die Zielerreichung überwachen zu können. Für die Spezifikation von auf Maßzahlen (Kennzahlen) basierenden Indikatoren kann in vielen Organisationen auf vorhandene integrierte Datenbestände (z.B. Data Warehouses) zurückgegriffen werden, die eine multidimensionale und analyseorientierte Sicht auf die Daten bieten.

Da Organisationen ihre Strategie aufgrund veränderter Ziele oder Rahmenbedingungen kontinuierlich anpassen, kann die Etablierung von Performance Management kein einmaliger Vorgang sein. Stattdessen müssen Ziele, vermutete Ursache-Wirkungsbeziehungen zwischen Zielen, die Verknüpfung von Zielen mit Indikatoren sowie Indikatoren und die zugrunde gelegten Maßzahlen selbst in einem nicht endenden Zyklus hinterfragt und angepasst werden.

In diesem Beitrag wird der Performance-Management-Kreislauf vorgestellt und auf dieser Basis ein Metamodell abgeleitet, das die Zusammenhänge von Zielen, Indikatoren und Maßzahlen im Performance Management beschreibt.

1 Einleitung

Neuere Managementansätze wie beispielsweise die Balanced Scorecard, die im folgenden unter dem Oberbegriff Performance Management zusammengefasst werden, propagieren die Operationalisierung der Strategie einer Organisation unter Verwendung von Maßzahlen (Kennzahlen). Da für diese Maßzahlen aggregierte Daten betrachtet werden müssen, können Organisationen auf vorhandene integrierte Datenbestände zurückgreifen, die sie oft bereits in Form eines Data Warehouses aufgebaut haben, um entscheidungsrelevante Informationen zu gewinnen.

In diesem Beitrag wird in Abschnitt 2 zunächst verdeutlicht, inwiefern die bei der multidimensionale Datenanalyse betrachteten Maßzahlen als Grundlage für Performance Management dienen können. Anschließend wird in Abschnitt 3 beschrieben, wie Maßzahlen bei der Operationalisierung von Strategien genutzt werden. In Abschnitt 4 wird daraus ein Metamodell abgeleitet, das eine Verbindung der Strategie einer Organisation mit Maßzahlen auf multidimensionalen Daten herstellt und den kleinsten

gemeinsamen Nenner der verschiedenen Performance-Management-Ansätze repräsentiert. Der Beitrag schließt in Abschnitt 5 mit einer Zusammenfassung und einem Ausblick.

2 Multidimensionale Datenanalyse

In verschiedenen Organisationen sammeln sich wertvolle digitale Datenbestände an, deren interaktive oder (semi-)automatisierte Auswertung nützlich erscheint. Viele Unternehmen integrieren beispielsweise ihre vorhandenen operativen Datenbestände zu Data Warehouses, um diese durch On-line Analytical Processing (OLAP) interaktiv oder auch durch Data Mining (semi-)automatisiert analysieren zu können. Dadurch sollen entscheidungsrelevante Informationen gewonnen werden, die letztlich zu einem Wettbewerbsvorteil führen (Inmon (1996)). Das Konzept des Data Warehouse hat sich zunächst im betriebswirtschaftlichen Umfeld etabliert, lässt sich aber auch auf andere Anwendungsdomänen wie beispielsweise das Gesundheitswesen übertragen (Meister u. a. (2003)).

Das Data Warehouse als physische Datenbank bietet eine integrierte und analyseorientierte Sicht auf die Daten. Der oft erhebliche Aufwand für die Datenintegration wird ausschließlich für den Zweck der Analyse der Daten betrieben. Als adäquater Modellierungsansatz für die Strukturierung der zu analysierenden Daten hat sich das multidimensionale Datenmodell etabliert (Wietek (2000)), welches die Denkweise des Analysten in Dimensionen und Klassifikationshierarchien widerspiegelt und die Betrachtung aggregierter Daten vorsieht.

Viele Anwender beschränken sich bei der Nutzung entsprechender Analyse-Werkzeuge hinsichtlich der Interaktion darauf, in regelmäßigen Abständen standardisierte Berichte zu den für sie relevanten Themenbereichen einzusehen, beispielsweise über ein Portal. Dabei greifen sie sich auf Maßzahlen eines vordefinierten Maßzahlsystems zurück. Wird das multidimensionale Datenmodell in den Mittelpunkt der Analyse gestellt, lassen sich die Daten unter Verwendung von OLAP-Operatoren navigierend entlang der Klassifikationshierarchien untersuchen.

Auf der Suche nach Strukturen, Mustern und einfachen Zusammenhängen in den Daten mit dem Ziel der Hypothesen- und Modellformulierung werden von Analysten neben dem Detaillierungsgrad auch weitere Parameter wie beispielsweise den relevanten Datenraum oder das anzuwendende statistische Verfahren variiert. Insbesondere wird dann nicht zwingend auf eine vordefinierte Menge an Maßzahlen zurückgegriffen, vielmehr stoßen Analysten möglicherweise auf neue interessante Maßzahlen.

3 Operationalisierung von Strategien mittels Maßzahlen

Unterschiedliche Organisationen setzen zunehmend auf organisationsweites Performance Management zur strategischen Steuerung (Daum (2002); Klingebiel (1999); Scherer und Alt (2002)). Dabei wird angestrebt, die Strategie zu operationalisieren und die Leistung (Performance) der Organisation gezielt zu beeinflussen. Hierzu werden Ziele mit Indikatoren (ausgewählte Maßzahlen/Kennzahlen) verknüpft, um die Zieleinhaltung zu überwachen. Für die Berechnung dieser Indikatoren kann auf vorhandene integrierte Datenbestände und deren aggregierte und multidimensionale Sicht auf die Daten zurück gegriffen werden.

Performance Management integriert Ziele, Strategien und Steuerungsgrößen in einem permanenten Führungssystem (Brunner (1999)). Das am weitesten verbreitete Instrument zur Umsetzung dieser Idee ist die Balanced Scorecard (Kaplan und Norton (2001)). Darüber hinaus gibt es jedoch eine Vielzahl weiterer Performance-Management-Ansätze (Brunner (1999); Gleich (2001); Klingebiel (1999); Lynch und Cross (1995)). Die Konzepte des Performance Management stammen ursprünglich aus der betriebswirtschaftlichen Domäne, werden aber zunehmend auch auf andere Domänen übertragen (Klingebiel (1999); Scherer und Alt (2002); Tropp (2002)).

Die Etablierung von Performance Management ist kein einmaliger Vorgang. Stattdessen werden die Phasen Zielformulierung, Modellierung, Performance Measurement sowie Hinterfragen von Zielen in einem nicht endenden Zyklus durchlaufen (siehe Abbildung 1). Im inneren Zyklus erfolgt die tatsächliche Messung der Leistung (Performance Measurement). Die Rückkopplung im äußeren Zyklus ermöglicht eine kontinuierliche Anpassung der Strategie aufgrund veränderter Ziele oder Rahmenbedingungen. Im Sinne des Double Loop Learning wird hierbei nicht nur die Frage „Are we doing the things right", sondern im äußeren Zyklus auch die Frage „Are we doing the right things" thematisiert (Argyris u. a. (2002); Weber und Schäffer (2000)).

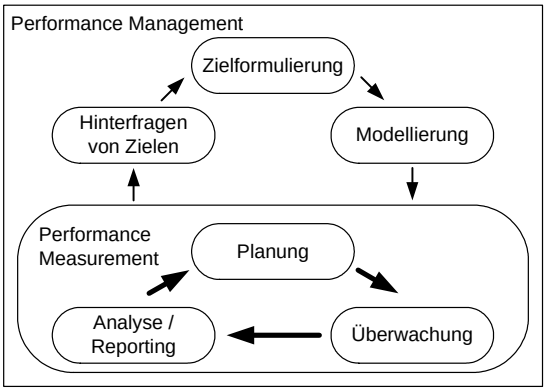


ABBILDUNG 1: Performance-Management-Kreislauf (Meister und Koch (2004))

Die hieraus resultierende kontinuierliche Anpassung einer Strategie ist insbesondere erwünscht, da eine Strategie ein System von Hypothesen über Ursache-Wirkungsbeziehungen ist (Kaplan und Norton (2001)), das sich nicht in einem „mathematischen Totalmodell" beschreiben lässt (Weber und Schäffer (2000)). Somit werden in der Modellierungsphase Kausalbeziehungen zwischen strategischen Zielen betrachtet, die sich zwar empirisch stützen (z.B. durch Sensitivitätsanalysen), jedoch nicht beweisen lassen.

4 Performance-Management-Metamodell

Ausgehend von der Definition von Maßzahlen im multidimensionalen Datenmodell lässt sich ein Metamodell für das Performance Management spezifizieren (siehe Abbildung 2), das die in Abschnitt 3 beschriebene Einbettung von Maßzahlen in den Kontext einer Strategie explizit widerspiegelt. Dieses Metamodell bildet den kleinsten gemeinsamen Nenner der verschiedenen Ansätze des Performance Management und kann gegebenenfalls um Spezifika bestimmter Ansätze erweitert werden.

Die Strategie als Hypothese über vermutete Ursache-Wirkungs-Zusammenhänge zwischen Teilzielen lässt sich auf verschiedene Weise repräsentieren. In der Regel werden Ziele und die zwischen Zielen vermuteten Wirkungen in Form von gerichteten azyklischen Graphen dargestellt, in denen kausale Verbindungen durch Pfeile von der Ursache hin zum Effekt dargestellt werden. Basierend auf den Grundmodellen Gemeinsame-Ursache-Modell, Gemeinsamer-Effekt-Modell und Kettenmodell lassen sich Kausalmodelle zusammensetzen (Hagmeyer (2001)).

Ziele werden mit Aktivitäten verknüpft, die zur Zielerreichung führen sollen. Um die Messung der Leistung einer Organisation zu ermöglichen, werden Ziele zudem mit ausgewählten Maßzahlen (Indikatoren) verknüpft, welche die Zielerreichung möglichst treffend und eindeutig wiedergeben (Eisenführ und Weber (2003)). Zur Interpretation der Indikatorwerte werden Referenzbereiche definiert. Die Grenzen der Referenzbereiche können fix (d.h. feste Zielvorgabe) oder relativ (z.B. durch Vergleich mit anderen Geschäftseinheiten) definiert sein. Oft genügt für ein Ziel genau ein Indikator, zur Balancierung können gegebenenfalls aber auch mehrere Indikatoren erforderlich sein. Diese müssen innerhalb der Indikator-Menge dann geeignet logisch verknüpft sein, so dass beispielsweise verlangt wird, dass die Werte aller Indikatoren in ihrem Referenzbereich liegen.

Grundlage des Metamodells bildet die Definition der mathematischen Beziehungen in Form von Maßzahlen auf multidimensionalen Daten, deren Gesamtheit ein Maßzahlsystem bildet. Maßzahlen, die die

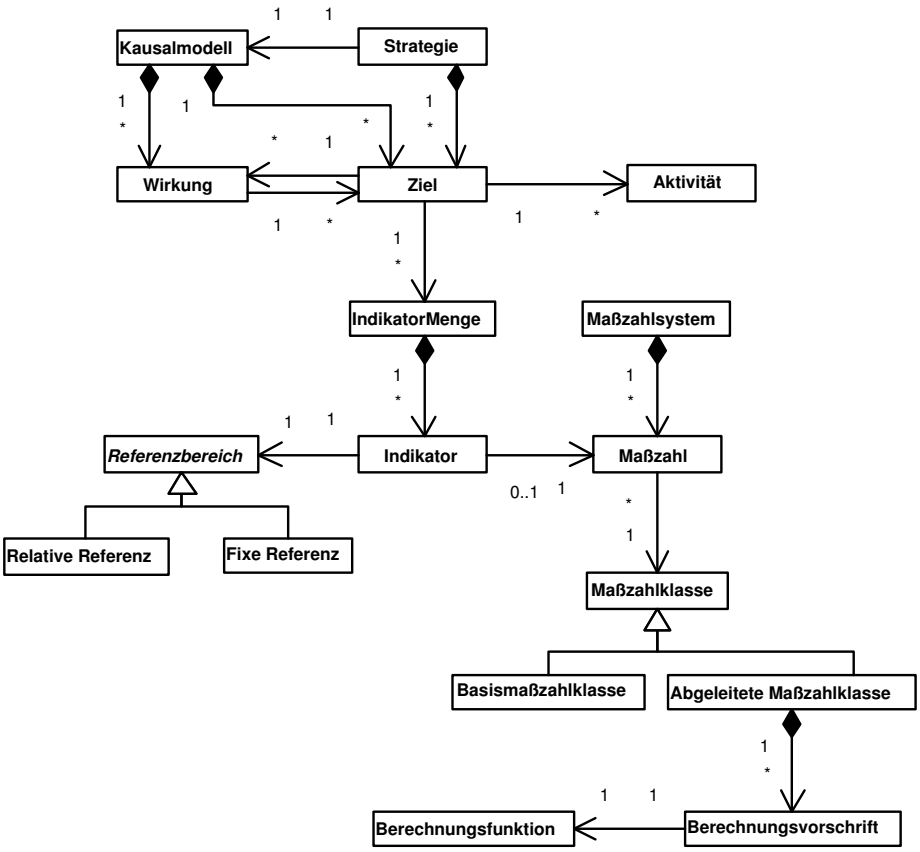


ABBILDUNG 2: Performance-Management-Metamodell

gleiche Berechnungsvorschriften besitzen, werden zu einer Maßzahlklasse (z.B. Anteil, Median) zusammengefasst. Dabei werden auf Grundlage von integrierten Datenbeständen bereitgestellte Basismaßzahlklassen sowie daraus abgeleitete Maßzahlklassen unterschieden. Eine Berechnungsvorschrift spezifiziert die (möglicherweise aggregierende) Berechnung aus Quellmaßzahlen und umfasst insbesondere auch die Berechnungsfunktion, die die Berechnung von Zellinhalten definiert.

5 Zusammenfassung und Ausblick

In diesem Beitrag wurde ein Metamodell vorgestellt, das unabhängig von bestimmten Performance-Management-Ansätzen das Konzept der Operationalisierung der Strategie einer Organisation mit Hilfe von Maßzahlen auf multidimensionalen Daten repräsentiert. Dieses Metamodell kann dazu genutzt werden, bestehende Analysewerkzeuge wie beispielsweise die in OFFIS entwickelte Analyseplattform MUSTANG (Koch u. a. (2003); Meister u. a. (2003)) um ein entsprechendes Grundgerüst für das Performance Management zu erweitern, so dass der Performance-Management-Kreislauf weitestgehend durch Werkzeug-Unterstützung begleitet werden kann.

Der vorgestellte Performance-Management-Kreislauf setzt sich aus zwei nicht endende Zyklen zusammen. Dies hat zur Folge, das in einer Organisation Indikatoren, Ziele sowie deren Verknüpfung untereinander ständig weiterentwickelt werden. Daher ist es sinnvoll, nicht nur entsprechend dem Data-Warehouse-Prinzip integrierte Daten zu historisieren, sondern darüber hinaus auch die auf dem Performance-Management-Metamodell basierende Strategierepräsentation. Entsprechende Konzepte zur Historisierung und Versionierung sind zukünftig noch unter Nutzung des Metamodells zu entwickeln.

Literatur

[Argyris u. a. 2002] ARGYRIS, Chris ; SCHÖN, Donald A. ; RHIEL, Wolfgang: *Die lernende Organisation – Grundlagen, Methode, Praxis*. Klett-Cotta, 2002

[Brunner 1999] BRUNNER, Jürgen: *Value-Based Performance Management – Wertsteigernde Unternehmensführung: Strategien, Instrumente, Praxisbeispiele*. Gabler Verlag, 1999

[Daum 2002] DAUM, Jürgen H.: *Intangible Assets*. Galileo Press, 2002

[Eisenführ und Weber 2003] EISENFÜHR, Franz ; WEBER, Martin: *Rationales Entscheiden*. Springer-Verlag, 2003

[Gleich 2001] GLEICH, Ronald: *Das System des Performance Measurement*. Verlag Vahlen, 2001

[Hagmeyer 2001] HAGMEYER, York: *Denken mit und über Kausalmodelle*, Georg-August-Universität Göttingen, Dissertation, 2001. – URL http://webdoc.sub.gwdg.de/diss/2001/hagmayer/

[Inmon 1996] INMON, William H.: *Building the Data Warehouse*. John Wiley & Sons, 1996

[Kaplan und Norton 2001] KAPLAN, Robert S. ; NORTON, David P.: *Die strategiefokussierte Organisation: Führen mit der Balanced Scorecard*. Schäffer-Poeschel Verlag, 2001

[Klingebiel 1999] KLINGEBIEL, Norbert: *Performance Measurement: Grundlagen - Ansätze - Fallstudien*. Gabler Verlag, 1999

[Koch u. a. 2003] 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

[Lynch und Cross 1995] LYNCH, Richard ; CROSS, Kelvin F.: *Measure Up! – How to Measure Corporate Performance*. Blackwell Publishers, 1995

[Meister und Koch 2004] MEISTER, Jürgen ; KOCH, Sascha: Konzeption einer Performance-Management-Plattform. In: BAUER, Andreas (Hrsg.) ; BÖHNLEIN, Michael (Hrsg.) ; HERDEN, Olaf (Hrsg.) ; LEHNER, Wolfgang (Hrsg.): *Internationales Symposium: Data-Warehouse-Systeme und Knowledge-Discovery*, Shaker Verlag, Juni 2004, S. 43–52

[Meister u. a. 2003] MEISTER, Jürgen ; ROHDE, Martin ; APPELRATH, Hans-Jürgen ; KAMP, Vera: Data-Warehousing im Gesundheitswesen. In: *it - Information Technology* 4 (2003), August, S. 179–185

[Scherer und Alt 2002] SCHERER, Andreas G. ; ALT, Jens M.: *Strategische Steuerung und Balanced Scorecard*. Veröffentlichung des Bundesverwaltungsamtes. Juli 2002

[Tropp 2002] TROPP, Gerhard: *Kennzahlensysteme des Hochschul-Controlling: Fundierung, Systematisierung, Anwendung*. Bayer. Staatsinst. für Hochschulforschung und Hochschulplanung. 2002

[Weber und Schäffer 2000] WEBER, Jürgen ; SCHÄFFER, Utz: *Balanced Scorecard & Controlling*. Gabler Verlag, 2000

[Wietek 2000] WIETEK, Frank: *Intelligente Analyse multidimensionaler Daten in einer visuellen Programmierumgebung und deren Anwendung in der Krebsepidemiologie*, Carl von Ossietzky Universität Oldenburg, Dissertation, 2000. – URL http://docserver.bis.uni-oldenburg.de/publikationen/dissertation/fb10.h%tml

Konzepte zur Datenqualitätssicherung in analytischen Anwendungen

Mathias Körbs Eike Schallehn
Fakultät für Informatik
Universität Magdeburg, Postfach 4120, 39016 Magdeburg
mathias@koerbs.de,eike@iti.cs.uni-magdeburg.de

Zusammenfassung

Mit der zunehmenden Verwendung von Datenbanksystemen und stetig wachsenden Datenvolumen wird die Frage nach der Qualität der verwalteten Daten immer bedeutender. Entsprechend wurden die Fragen, was genau Datenqualität ist und wie man sie messen und bewerten kann, in den letzten Jahren zu einem aktuellen Thema in der Forschung und in der industriellen Praxis. Inbesondere bei analytischen Anwendungen ist eine Bewertung der Qualität der zu Grunde liegenden Daten wichtig, um eine Aussage über die Güte der abgeleiteten Analyseergebnisse machen zu können. Dies wird in Anwendungen, die auf einem Data Warehouse oder ähnlichen Ansätzen basieren, zunehmend problematisch, da Informationen zur Qualität integrierter Datenbestände nur schwer ableitbar sind.

Während grundlegende Ansätze zu spezifischen Problemen der Bewertung und Messung von Datenqualität existieren, ist die Integration entsprechender Funktionalität in Informationssystemen immer noch ein weitgehend ungelöstes Problem. Diese Aufgabenstellung wird in unserer aktuellen Forschung in Kooperation mit Industriepartnern angegangen.

1 Einleitung

Mit zunehmenden Einsatz von datenintensiven Anwendungen gewinnt auch der Begriff der Datenqualität an Bedeutung. Die Qualität im Allgemeinen ist ein Schlüsselfaktor bei der Herstellung beliebiger Produkte und dem Angebot von Dienstleistungen. Handelsketten, Automobilkonzerne und viele andere Unternehmen verlangen von ihren Zulieferern ein klar definiertes Mindestmaß an Qualität. Dabei beziehen sich Qualitätsanforderungen nicht nur auf die Produkte an sich, sondern auch auf alle Prozesse im Unternehmen. Vieles muss dokumentiert werden, wird nachgemessen und kontrolliert. Über die Qualität von Daten ist hingegen oftmals wenig bekannt. Häufig beschränken sich Aussagen über die Datenqualität auf Abschätzungen. Dennoch ist der Umgang mit Daten unter Qualitätsaspekten von Bedeutung, gerade dann, wenn auf Basis der Daten wichtige Entscheidungen getroffen werden sollen.

Die Datenqualität ist ein besonders kritischer Aspekt bei der Datenintegration, wo Daten aus verschiedenen autonomen Quellen zusammengeführt werden, und es häufig zu schwer abschätzbaren Fehlern und Inkonsistenzen kommt. Bei Data Warehouse-Systemen handelt es sich um einen speziellen und sehr erfolgreichen Ansatz zur Datenintegration. Dazu werden aus verschiedenen Systemen eines Unternehmens Daten extrahiert und materialisiert, um diese als Grundlage für Analysen in einem übergreifenden Kontext zu verwenden. Jedoch hat eine schlechte Qualität der integrierten Ausgangsdaten oft auch erheblichen Einfluß auf die Qualität der Analysergebnisse.

Für die Sicherstellung der Datenqualität haben Forschung und Praxis zu zahlreichen Lösungen für spezielle Probleme geführt. Jedoch ist Datenqualität jeweils nur durch anwendungsspezifische Kriterien und Verfahren zu erreichen. Ein weiteres Problem ist die einheitliche Er-

fassung und Verwaltung von Qualitätsdaten sowie die Interpretation entsprechend eines einheitlichen Modells.

Im Rahmen der hier dargestellten Forschungsarbeiten sollten am Beispiel eines Systems zur Durchführung von Analysen zur Fahrzeugsicherheit bei einem großen Automobilhersteller Konzepte zur Integration von Datenqualitätsfunktionalität untersucht werden. Dabei sollte einerseits eine enge Integration mit der bestehenden Anwendung erreicht werden, und andererseits soll die Lösung für zahlreiche relevante Datenqualitätsaspekte offen und erweiterbar sein.

2 Stand der Forschung

Angelehnt an die Qualittsdefiniton der ISO-Norm 9000:2000 [EN00] lässt sich die Qualität von Daten ähnlich wie bei natürlichen Produkten oder Dienstleistungen als eine Menge von Merkmalen dieser Daten ausdrücken. Solche Merkmale sind den Daten inhärente Eigenschaften, also unveränderlich mit ihnen verbunden. Bei einer Schraube ist die Tiefe ihres Gewindes ein Merkmal [Gie01], bei einer Menge von Daten ist dies zum Beispiel die Vollständigkeit. Ob eine konkrete Merkmalsausprägung hohe Qualität bedeutet hängt dabei von Anforderungen ab, die von interessierten Parteien, den Daten-Nutzern, Anbietern oder Produzenten, aufgestellt werden. Problematisch ist, dass in der Literatur viele Defintionen des Begriffes Datenqualität auf den Konzepten des Total Quality Management basieren, das im Gegensatz zur ISO 9000:2000 keine scharfe Definiton für Qualität kennt [GD03]. Die Folge ist eine uneinheitliche Terminologie. So werden die Begriffe Dimension, Attribut und Merkmal synonym verwendet. Lediglich in [Hin02] kommt eine Definition nach der ISO-Norm 9000:2000 zur Anwendung.

Um eine hohe Qualität von Daten zu erlangen ist es notwendig aktiv darauf Einfluss zu nehmen, das heißt, nicht auf Folgen mangelnder Datenqualität zu reagieren, sondern lenkend einzugreifen. Ein solcher Qualitätsmanagementprozess besteht im wesentlichen aus den vier Phasen

1. Anforderungen aufstellen,
2. Ist-Zustand ermitteln,
3. Ist-Zustand analysieren und
4. Verbesserungsmaßnahmen ableiten und durchführen.

Um eine ständige Qualitätskontrolle und -verbesserung zu erreichen werden die Phasen immer wieder iterativ ausgeführt. Selbst wenn ein gewünschtes Maß an Qualität erreicht ist, ist dieser Prozess zur Kontrolle notwendig. Auf Datenqualität angepasste Qualitätsmanagementprozesse werden unter anderen im Rahmen der Total Data Quality Management (TDQM) [Wan98] und Data Warehouse Quality (DWQ) [YV97, VBQ99] Projekte vorgestellt.

3 Konzepte zur Integration der Qualitätsicherung

Um die oben genannten Zielsetzungen des Qualitätsmanagements zu realisieren, sind im Folgenden vor allem Fragen der Integration des DQ-Managements in einem gegebenen Anwendungskontext sowie die flexible und erweiterbare Unterstützung verschiedener Metriken von Interesse.

3.1 Einbettung des DQ-Frameworks

Während sich für das Qualitätsmanagement Anforderungen und notwendige Verbesserungsmaßnahmen anwendungsspezifisch zumeist relativ einfach bestimmen lassen, ist die Ermittlung des

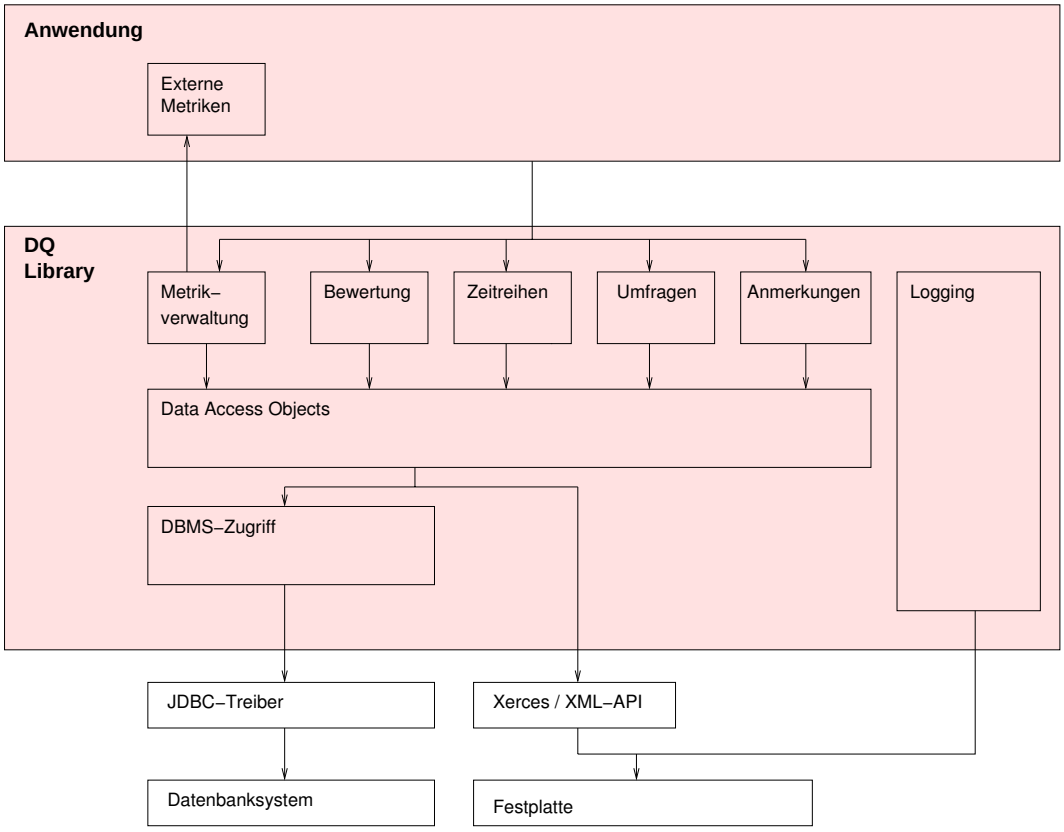


Abbildung 1: Architektur des DQ-Frameworks

Ist-Zustandes problematisch, weil Informationen über die Qualität aus den Daten gewonnen werden müssen.

Um die Qualität von Daten zu ermitteln, müssen die Ausprägungen relevanter Merkmale gemessen und die Messwerte daraufhin mit vorher aufgestellten Anforderungen verglichen und bewertet werden. Gerade die Messung ist schwierig, weil umfangreiches Wissen über die Daten notwendig ist, die Datenmenge wegen ihrer Größe häufig unhandlich ist, und weil viele Merkmale von der subjektiven Meinung eines Experten abhängen.

Um unter diesen Bedingungen dennoch Qualitätsaussagen zu erhalten, lassen sich Umfragen unter Angehörigen der interessierten Parteien durchführen oder Metriken speziell für einen bestimmten Anwendungsfall erstellen. Ein solches Vorgehen ist aber sehr teuer und daher, gerade durch den zyklischen QM-Prozess, häufig nicht durchführbar.

Dabei stellt sich die Frage, ob die Ermittlung der Qualität von Daten nicht mit einfachen und vor allem wiederverwendbaren Metriken ausreichend ist, gerade wenn das Verhältnis aus Kosten und Nutzen relevant ist und dadurch Ungenauigkeiten in Kauf genommen werden können. Ausgehend von dieser Problematik wurde ein Software-Framework entwickelt, um eine möglichst einfache Messung und Bewertung von Datenqualitätsmerkmalen zu ermöglichen. Die grundlegende Architektur des Frameworks ist in Abbildung 1 dargestellt.

3.2 Flexible Unterstützung von Metriken

Besondere Aspekte des Frameworks sind eine Metrikschnittstelle, die Normalisierung und Aggregation der ermittelten Messwerte um diese in einen Aggregationsbaum darstellen zu können, Zeitreihen um die Entwicklung der Merkmalsausprägungen über einen Zeitraum zu verfolgen, Umfragen um subjektive Merkmalsausprägungen zu ermitteln und die Möglichkeit der Markierung von Daten, damit den Daten-Nutzern ein Feedback über Fehler in den Daten gegeben

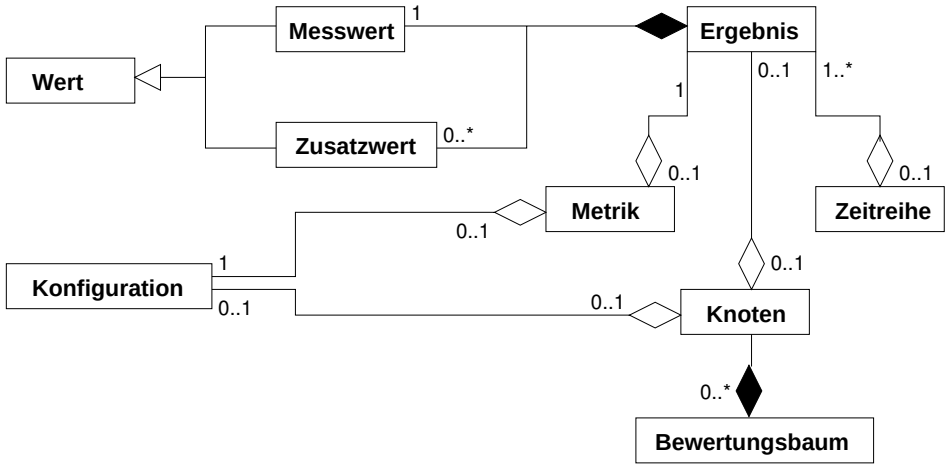


Abbildung 2: Konzeptionelles Schema des DQ-Frameworks

werden kann.

Im Einzelnen sind die betrachteten Daten enstprechend des in Abbildung 2 dargestellten Schemas strukturiert. Im Mittelpunkt stehen hierbei Metriken, durch deren Anwendungen eine konkrete Messung durchgeführt wird. Hierzu benötigt die Metrik eine Konfiguration, welche zum Beispiel festlegt, auf welche Daten (Relationen) die Metrik angewandt wird, und welche spezifischen Parameter gesetzt wurden. Das Ergebnis der Messung besteht nun aus einem einzelnen Messwert und gegebenenfalls weiteren Zusatzwerten, die zum Beispiel der Interpretation des Ergebnisses dienen können.

Zeitreihen werden zur kontinuierlichen und wiederholten Ausführung von Messungen verwaltet. Durch diese Sammlung von Messergebnissen und zugehörigen Zeitstempeln wird eine wichtige Grundlage für das Datenqualitätsmanagement gegeben, da auf diese Art und Weise die Bewertung durchgeführter Maßnahmen zur Verbesserung der Datenqualität möglich wird.

In einem Bewertungsbaum können die Ergebnisse verschiedener Messungen zusammengefasst werden und hierarchisch als Knoten dargestellt werden. Eine entsprechende Darstellung eines Bewertungsbaums in der Applikation zur Auswertung von Messeregebnissen ist in Abbildung 3 gegeben.

4 Zusammenfassung

Die hier skizzierten Ansätze wurden im Rahmen einer Diplomarbeit in der Forschungsabteilung eines Automobilherstellers implementiert und befinden sich momentan im Einsatz.

Durch eine Messung auf den Daten der dort eingesetzten analytischen Anwendung wurde ermittelt, inwieweit sich das Framework in diese Anwendung integrieren lässt und ob sich vorher benannte Daten quantifizieren lassen. Dabei wurden Messwerte für die Vollständigkeit auf Attribut und auf Tupelebene, der referentiellen Integrität und der Einhaltung von Konsistenzregeln ermittelt. Eine weitere Metrik wurde zur Messung eines in der Anwendung vorkommenden Spezialfalles der referentiellen Integrität verwendet.

Die Auswertung hat gezeigt, dass die Messwerte ausreichend genau waren, um das Ausmaß der vorher benannten Datenfehler zu bestimmen. Auch hat sich gezeigt, dass sich das Framework einfach in die Anwendung integrieren ließ.

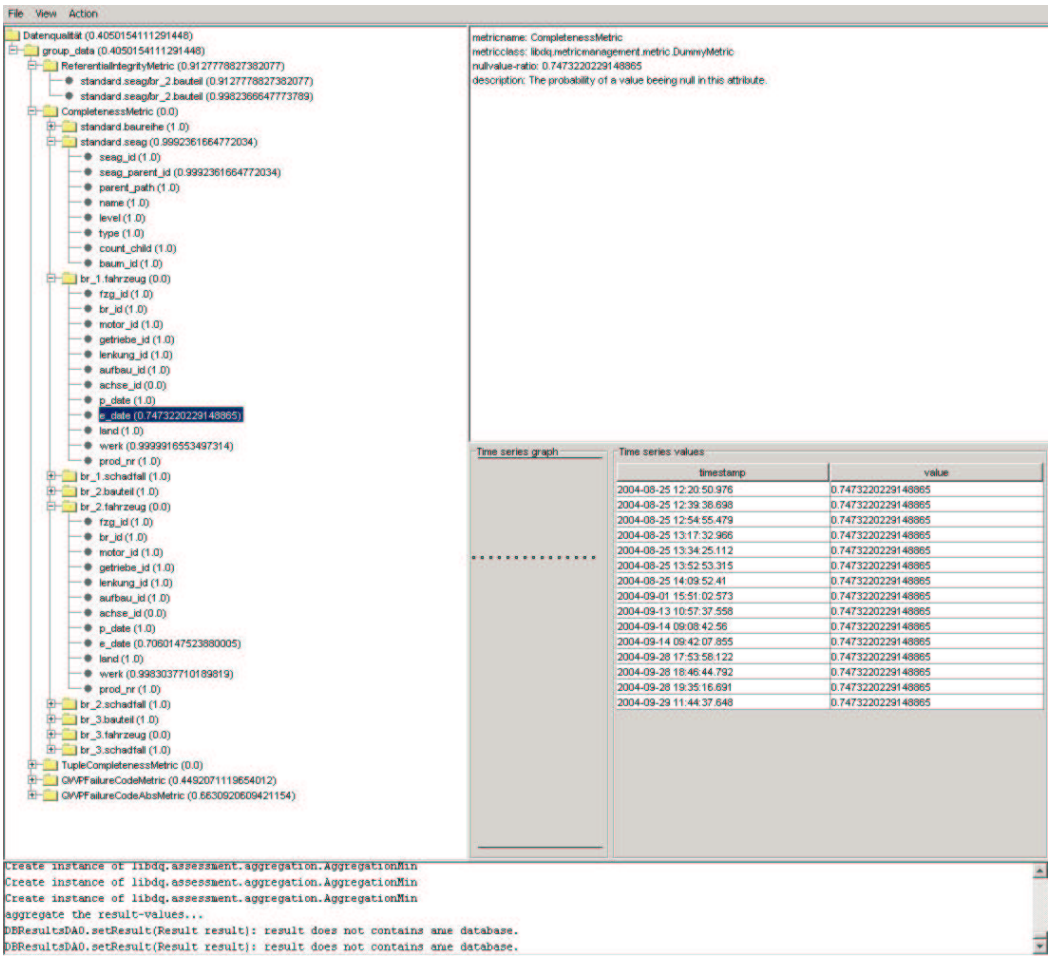


Abbildung 3: Applikation zur Auswertung und Kombination von Messergebnissen

Literatur

[EN00] DIN EN. ISO 9000 : 2000 Qualitätsmanagementsysteme Grundlagen und Begriffe., 2000.

[GD03] D.L. Goetsch and S. B. Davis. *Quality management: introduction to total quality management for production, processing and services.* Pearson Education, Inc., New Jersey, 2003.

[Gie01] D.G. Gietl. *Qualitätsmanagement: Begriffe und Definitionen.* Dr. Ingo Resch GmbH, 2001.

[Hin02] H. Hinrichs. *Datenqualitätsmanagement in Data-Warehouse-Systemen.* 2002.

[VBQ99] P. Vassiliadis, M. Bouzeghoub, and C. Quix. Towards quality-oriented data warehouse usage and evolution. pages 164–179, 1999.

[Wan98] R.Y. Wang. A product perspective on total data quality management. 41(2):58–65, 1998.

[YV97] M. Yarke and Y. Vassiliou. Data warehouse quality: A review of the dwq project. pages 299–313, 1997.

Ein merkmalsorientierter Speichermanager für eingebettete Systeme

Thomas Leich und Sven Apel
{*leich*|*apel*}*@iti.cs.uni-magdeburg.de*
Fakultät für Informatik, Otto-von-Guericke-Universität Magdeburg

Zusammenfassung

Der Bereich der eingebetteten Systeme ist ein bedeutender Markt. Einsatzgebiete sind beispielsweise Autosteuerungen oder Sensornetzwerke. Häufig benötigen derartig eingebettete Rechnersysteme Infrastruktursoftware zur Datenhaltung. Auf Grund der Heterogenität der Hard- und Software, sowie der extremen Ressourcenbeschränkungen, ist eine Adaption der klassischen Mehrzwecksysteme aus Großrechnern oder dem PC-Bereich nicht möglich. Eine Maßschneiderung der Datenmanagementfunktionalität auf den jeweiligen Anwendungskontext, sowie auf die Hard- und Softwareumgebung ist unumgänglich. Die hierfür benötigten Komponenten müssen leichtgewichtig, minimal und feingranular sein. Eine Kombination von Komponententechniken und merkmalsorientierter Programmierung bietet Möglichkeiten zur Überwindung der genannten Probleme. Im Folgenden werden der Entwurf und die Implementierung eines Speichermanagers im Sinne einer Programmfamilie mittels merkmalsorientierter Domänenanalyse und merkmalsorientierter Programmierung beschrieben. Als Bewertungsgrundlagen dienen die Anzahl der Variationspunkte und die potentielle Anzahl sinnvoller Systemvarianten.

1 Einleitung und Motivation

Der Markt für eingebettete Systeme wächst stark [4]. Bei der Entwicklung eingebetteter Rechnersysteme sind die Kosten für Hardware auf Grund der zumeist hohen Stückzahlen ein sehr wichtiger Faktor. Daher kommt es bei der Softwareentwicklung darauf an, den Ressourcenbedarf zu minimieren, um so preisgünstige Hardware einsetzen zu können. Das Ergebnis ist typischerweise die Entwicklung von Spezialzwecksoftware, die für den konkreten Anwendungsfall zugeschnitten ist und somit keinen unnötigen Speicher oder Rechenzeit verbraucht. Dies erschwert die systematische Wiederverwendung, Wartung, Erweiterung und Anpassung der Software enorm. Heutige Infrastruktursoftware zur Datenhaltung, die üblicherweise im Großrechner- und PC-Bereich eingesetzt wird, kann auf die Heterogenität der Hard- und Software und auf die starken Ressourcenbeschränkungen nicht adäquat reagieren. Neue Softwaremethoden aus dem Bereich der Komponententechniken, Programmfamilien und der merkmalsorientierten Entwicklung können helfen diese Schwierigkeiten zu überwinden ohne die Vorteile eine Spezialzwecksoftware zu verlieren [5]. Dieser Beitrag zeigt erste Ergebnisse aus dem Entwicklungsprozess eines merkmalsorientierten Speichermanagers (SM) in Form einer Programmfamilie. Aus der SM-Familie können speziell für den Anwendungskontext zugeschnittene leichtgewichtige Datenhaltungskomponenten abgeleitet werden. Die Ergebnisse in Hinsicht auf eine feingranuale Maßschneiderbarkeit des SM sollen an einem Beispielszenario aus dem Bereich der Sensornetzwerke demonstriert werden. Der Beitrag ist wie folgt gegliedert: Abschnitt 2 erläutert ein Beispielszenario aus dem Bereich der Sensornetzwerke und arbeitet unterschiedliche Anforderungen an einen SM in diesem Bereich heraus. In Abschnitt 3 werden die verwendeten Softwaretechniken vorgestellt. Der folgende Abschnitt präsentiert den Entwurf und die Implementierung des merkmalsorientierten SM. Abschnitt 5 diskutiert und vergleicht die erreichten Ergebnisse. Eine Zusammenfassung und einen Ausblick gibt Abschnitt 6.

2 Beispielszenario

Im Bereich der drahtlosen Sensornetzwerke konzentrierte sich die Forschung der letzten Jahren vor allem auf Anfragebearbeitung in Netzwerken und (Vor-) Aggregationsalgorithmen [6]. Zentralisierte Datenhaltung und Analyse ermöglichen den Einsatz preiswerter Sensorenknoten mit geringem Ressourcenverbrauch. Oft sind Sensornetzwerke für den Langzeitbetrieb entwickelt und somit auf Grund der knappen Ressourcen in der Kommunikation stark eingeschränkt [6]. Voraggregationen von Rohdaten auf den Sensorknoten selbst sollen helfen den Speicherverbrauch zu minimieren und die Kommunikationsbeschränkungen zu überwinden. Dies ist nur möglich, wenn die zu untersuchenden Merkmale a priori bekannt sind. Gerade in wissenschaftlichen Anwendungsbereichen wie der Biotopüberwachung ist dieses nicht immer möglich [6]. In Anlehnung an [12, 6] wird im Folgenden ein Versuchsaufbau zur Überwachung eines Biotops vorgestellt. Einfache *Sensorknoten (1)* messen und speichern Daten wie z. B. Temperatur oder Lichtintensität. Die jeweilige Record-Länge ist für einen Sensorknotentyp fest. Des Weiteren benötigen die Sensoren eine einfache Speicherstruktur mit Einfüge- und Suchfunktionalität. Eine Löschfunktion ist z. B. nicht notwendig, da die im Hauptspeicher gehaltenen Daten periodisch mit einem „Reset“ zurücksetzt werden. *Datenkollektoren (2)* bilden die zweite Gerätegruppe, welche die Daten verschiedener Sensoren je nach Bedarf der Wissenschaftler „zusammentragen“, und Informationen für interne und externe Analysen bereitstellen. Für eine sichere und dauerhafte Speicherung wird eine Sekundärspeicherung verwendet, welche durch ein Buffermanagement optimiert wird. Die Speicherstruktur muss für die jeweilige Analyse geeignet sein und entsprechende Operationen anbieten. Des Weiteren werden Integritätschecks benötigt. Verfügbare Datenmanagementdienste können ein solches Spektrum unter Berücksichtigung der limitierten Ressourcen nicht abdecken. Einige Dienste bieten eingeschränkte Adaptionsmöglichkeiten. Diese sind nicht ausreichend, um auf die extremen Ressourcenbeschränkungen zu reagieren.

3 Softwaretechnischer Hintergrund

Für die Umsetzung des SM in Form einer Programmfamilie wurden die Konzepte der merkmalsorientierten Domänenanalyse (engl. Feature-Oriented Domain Analysis (FODA)), der schrittweisen Verfeinerung und der merkmalsorientierten Programmierung (engl. Feature-Oriented Programming (FOP)) angewendet. Für die konkrete Implementierung wurden Mixin Layer verwendet.

Merkmalsorienierte Domänenanalyse: Ziel der FODA ist die Eingrenzung der Zieldomäne und die Identifizierung der gemeinsamen und unterschiedlichen Merkmale der Zielapplikationen [5]. Die Ergebnisse werden in einem Baumdiagramm zusammengefasst. Diese abstrakte und implementierungsunabhängige Darstellung gibt einen Überblick über die Variationspunkte der Zielanwendungen. Optionale *(C,D)* und zwingende *(B)* Merkmale werden durch einen leeren bzw. gefüllten Kreis graphisch gekennzeichnet. Des Weiteren können Merkmale zu Oder- (G,H) und Alternativ- (E,F) Gruppierungen zusammengefasst werden, welche durch leere bzw. gefüllte Halbbögen dargestellt werden.

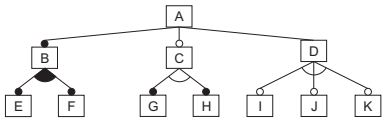


Abbildung 1: Merkmalsbaum

Programmfamilien und Schrittweise Verfeinerung: Den Begriff der Programmfamilie führte Parnas [9] ein. Familienmitglieder sollen aus einer gemeinsamen Basis abgeleitet werden. Eine Umsetzung kann durch *schrittweise Verfeinerung* [10] erfolgen. Beginnend mit einer minimalen Basis wird durch Hinzufügen von Schichten die Funktionalität inkrementell erweitert. Durch ein einfaches Hinzufügen, Austauschen und Weglassen einzelner Schichten wird Erweiterbarkeit, Wartbarkeit und Anpassbarkeit ermöglicht. Batory et al. [10] überführten diesen Ansatz in die Objekt-Orientierte Welt und erkannten, dass neue *Software-Merkmale* oft eine Erweiterung oder Modifikation mehrerer Klassen nach sich zieht. Basierend auf diesen Beobachtungen erkannten sie, dass Merkmale *Kollaborationen von Klassen und/oder Objektfragmenten* sind.

Abb. 2 zeigt Schichten von Kollaborationen. Klassen werden vertikal (c_1 – c_3) und Kollaborationen horizontal (f_1 – f_3) angeordnet. Verschiedene Merkmale bilden als Schichten von Kollaborationen die gewünschte Software. Im Sinne von FOP [10] implementiert eine Kollaboration von Objekten ein Merkmal [10][1]. *Mixin Layer* stellen eine mögliche Implementierungstechnik für die Umsetzung der Merkmale in Form von Kollaborationen dar [10]. Sie kapseln Fragmente verschiedener Klassen statisch. Dadurch wird ein hoher Grad an Modularisierung und ein einfaches Zusammensetzen der Schichten gewährleistet. AHEAD [1] stellt z. B. eine Spracherweiterung für Java bereit, um Klassen in Form von Mixins zu organisieren.

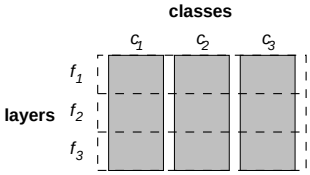


Abbildung 2: Kollaborationsschichten

4 Analyse, Entwurf und Implementierung

Im Folgendem wird die Analyse, der Entwurf und die Implementierung des SM beschrieben.

FODA: Abb. 3 zeigt einen Ausschnitt des Merkmalsdiagramms, welches sich aus der Domänenmodellierung ergibt. Die Merkmale in den grauen Boxen können weiter verfeinert werden. Der SM teilt sich in vier zwingende Merkmale: 1. *Data Type (DT)* repräsentiert die unterstützten Datentypen , 2. *Buffer Management (BM)* organisiert die primäre bzw. sekundäre Speicher- und die Freispeicherverwaltung, 3. *Storage Organisation (SO)* übernimmt die Zugriffsverwaltung der Daten und 4. *Record (Rec)* repräsentiert den internen Aufbau der Daten. Optional können verschiedene Integritätschecks *Integrity Checks (IC)* und Dateitypen *File Types (FT)* ausgewählt werden. Der B^**-Baum* mit seinen Operationen als Submerkmalen ist ein Beispiel für den feingranularen Entwurf. Nur dieser feingranulare Entwurf erlaubt eine Maßschneidung für stark ressourcenbeschränkte Umgebungen. Eine genauere Untersuchung würde hunderte Merkmale hervorbringen und führt schnell zu unübersichtlichen Merkmalsbäumen. Ein weiteres Problem ist, dass Merkmalsdiagramme in ihrer Aussagekraft beschränkt sind. So benötigen Änderungsoperationen eine Sucheoperation. Hierfür gibt es unterschiedliche Erweiterungen der Merkmalsdiagramme. Für den hier vorgestellten Speichermanager wurden diese Abhängigkeiten durch das Konzept der *Design Rule Checks (DRC)* [3] modelliert. DRC werden auf Implementierungsebene eingesetzt. Das gesamte Merkmalsmodell der SM-Familie umfasst 93 Merkmale. Auf eine Untersuchung von speziellen Datentypen, Transaktionsmechanismen oder Mehrbenutzerverfahren wurde verzichtet. Zur Bestimmung der Anzahl möglicher Varianten, die aus der SM-Familie entstehen können, werden die in Tab. 1 beschriebenen Annahmen getroffen. So werden z. B. durch die SM-Familie vier unterschiedliche Datentypen unterstützt. Mit Hilfe einer GenVoca-Grammatik [2] können für die präsentierte SM-Familie

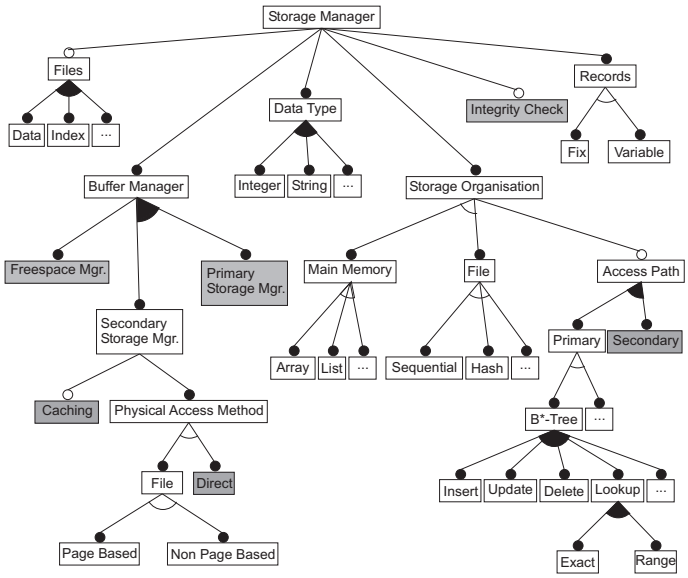


Abbildung 3: Merkmalsdiagramm des SM

[1] Im Folgenden werden Schichten und Merkmal deshalb synonym verwendet.

Parameter	Merkmal	# für Berechnung
d	Data Types	4
f	File Type	2
m	Main Memory Organisation	2
s	Data File Structures	2
a	Access Structure	2
o	B^* Tree	6

Tabelle 1: Parameter und Werte für die Berechnung

8.164.800 Varianten berechnet werden. Die Unterschiede zwischen einzelnen Varianten sind minimal, so dass die Forderung nach einer feingranularen Konfiguration erfüllt ist.

$$\#SM = \underbrace{(2^f - 1)}_{FT} * \underbrace{(15)}_{BM} * \underbrace{(2^n - 1)}_{DT} * \underbrace{((m + s) * (6 * a * (2^o - 1)))}_{SO} * \underbrace{(2)}_{IC} * \underbrace{(2)}_{Rec}$$

Entwurf und Implementierung: Zu Evaluierungszwecken wurde der SM mit Hilfe der *AHEAD-ToolSuite* umgesetzt. Abb. 4 zeigt einen Ausschnitt der implementierten Schichten. Beginnend mit der Basis wird der SM in jeder Schicht erweitert bzw. modifiziert. Dies geschieht durch das Hinzufügen von Klassen und/oder das Verfeinern bereit existierender Klassen. Für den SM wurden 36 Klassen und wenige Hilfsklassen implementiert. Im Durchschnitt wurden 3 Klassen pro Schicht verfeinert. Ein Beispiel hierfür ist das Merkmal *Page Based Storage*. In dieser Schicht werden die Klassen: `File`, `FreeSpaceMgr`, `SecStorageMgr` verfeinert und die Klasse `Page` hinzugefügt. Die AHEAD-ToolSuite generiert aus den in Abb. 4 dargestellten Klassen eine Vererbungshierarchie.

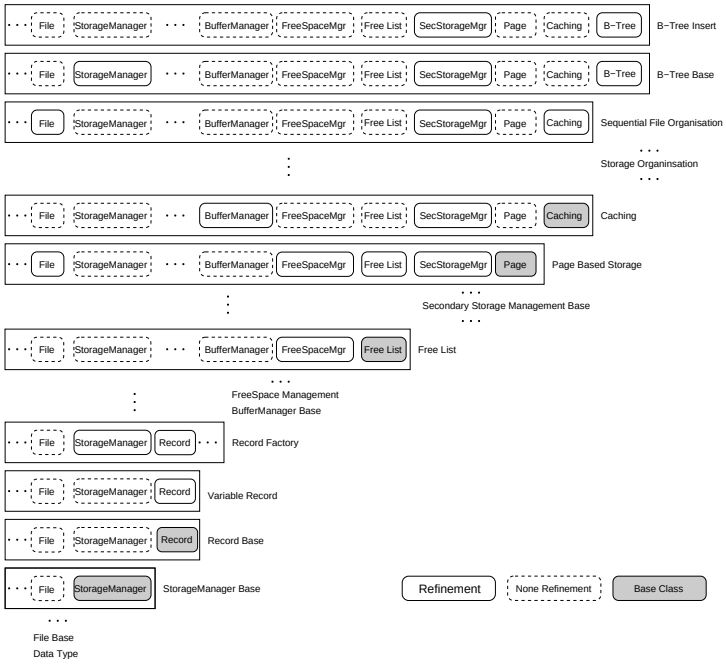


Abbildung 4: Layer Stack

5 Ergebnisse und Vergleich

Zu Evaluierungszwecken wurde für die im Abschnitt 2 eingeführten Szenarien eine Prototypenfamilie implementiert, von der verschiedene Konfigurationen abgeleitet werden können. Expemplarisch werden die Sensorknoten- und Datenkollektorenkonfiguration vorgestellt. *Sensorknoten:* Es wurden zwei Varianten konfiguriert. Beide Varianten sind Hauptspeicherdatenbanken mit einer statischen Speicherverwaltung (feste Record-Längen). In der ersten Sensorknotenvariante werden einfache Datentypen (integer, number) in einem Array gespeichert. Diese Konfiguration besteht aus 11 Schichten. Für die zweite Variante wurde statt einem Array eine Hash-Speicherstruktur verwendet. Dadurch können effiziente Such- und Änderungsoperationen unterstützt werden. Des Weiteren ist ein einfacher Integritätscheck vorhanden. Dieser SM besteht aus 16 Schichten. *Datenkollektoren:* Auf Grund der in Abschnitt 2 beschriebenen Anforderungen ist die Funktionalität der Datenkollektoren komplexer. Für die Sekundärspeicherverwaltung wurde eine seitenbasierte Speicherverwaltung konfiguriert und zur Überwindung der Zugriffslücke ein Cache-Management

eingeführt. Die Dateien sind sequenziell geordnet. Als Zugriffstruktur wurde ein B^*-Baum ausgewählt. Im Gegensatz zu den Sensorknoten wird im Datenkollektor ein variable Record-Länge verwendet, um auf die verschiedenen Datenformate der Sensoren effizienter reagieren zu können. Der eingefügte Integritätscheck ist im Vergleich zu dem des Sensorknotens komplexer. Die gesamte Konfiguration dieses SM wird durch 38 Schichten realisiert. Die semantische Richtigkeit der Konfigurationen wird durch DRC sichergestellt. Durch DRC konnten ca. 60 % aller Konfigurationen, die aus dem Merkmalsdiagramm möglich sind, ausgeschlossen werden.
Ein Vergleich zu Berkley DB soll Unterschiede zu bisherigen Lösungen verdeutlichen. Berkley DB liegt in einer C- und Java-Implementierung vor. Das DBMS ist auf der Basis seiner 4 Teilsysteme konfigurierbar: dem Zugriffspfadmanagement, der Speicher-, Transaktions- und Sperrverwaltung. Durch Kombination bzw. Weglassen einzelner Subsysteme wird die Konfiguration ermöglicht. Dieses ist eine vergleichsweise grobe Variabilität zu dem präsentierten Ansatz. Änderungen in einem Teilsystem wie z. B. der Zugriffpfadverwaltung sind möglich, aber auf Grund der Abhängigkeiten in den Modulen sehr schwierig. Tesanovic et. al [11] bestätigten diese Aussage durch Untersuchungen im Bereich der crosscutting concerns(CC). CC sind bekannt dafür, dass sie eine Erweiterung in Softwaresystemen behindern. Durch die Auslagerung einiger CC in Aspekte konnte die Codegröße um 57 % reduziert werden. Dies zeigt die hohe zyklomatische Komplexität in DBMS und die damit verbundenen Schwierigkeiten für die Konfigurier- und Erweiterbarkeit. Im COMET-Projekt [7] wurden viele Abhängigkeiten eines SM in Module und Aspekte getrennt. Auch bei der Entwicklung mit AOP besteht ein Merkmal typischerweise aus Klassen und Aspekten. Kohäsive Module, bei denen Klassen und Aspekte ein Merkmal zusammenfassen sind laut Lopez-Herrejon et al. [8] mit AOP nicht möglich. Dies führt vor allem in großen Projekten wie z. B. bei der Entwicklung von SM zu Problemen.

6 Zusammenfassung und Ausblick

Merkmalsorienierte Softwareentwicklung und schrittweise Verfeinerung ermöglichen eine variable Implementierung von Datenmanagementdiensten für eingebettete Systeme. Hierzu wurde die Kombination von FODA, FOP und Mixin Layer zur Implementierung eine SM-Familie beschrieben. Ein kleine Anzahl an Grundfunktionen wurden analysiert und implementiert, um einen hohen Grad an Variabilität und Maßschneiderung an einem Beispielszenario aus dem Bereich der Sensornetzwerke nachzuweisen. Durch einen einfachen Konfigurationsprozess konnten drei verschiedenen Varianten des SM auf verschiedene Anwendungsszenarien zugeschnitten werden. Zukünfige Arbeiten konzentrieren sich auf eine Umsetzung des SM mit C++. Hierzu wird C++ um FOP-Funktionalitäten erweitert. Des Weiteren werden Kombinationen von FOP und AOP für die Umsetzung evaluiert.

Literatur

[1] D. Batory. Feature-Oriented Programming and the AHEAD Tool Suite.
[2] D. Batory and S. O'Malley. The design and implementation of hierarchical software systems with reusable components. *ACM Transactions on Software Engineering and Methodology*, 1(4):355–398, 1992.
[3] D. Batory, J. N. Sarvela, and A. Rauschmayer. Scaling Step-Wise Refinement. In *Proc. of the 25th Int. Conf. on Software Engineering*, 2003.
[4] Business Communications Company. Future of Embedded Systems Technology, 2000. BCC Press release on market study RG-229.
[5] K. Czarnecki and U. Eisenecker. *Generative Programming: Methods, Tools, and Applications*. Addison-Wesley, 2000.
[6] D. Ganesan et. al. An evaluation of multi-resolution storage for sensor networks. In *Proceedings of the ACM SenSys Conference*, pages 89–102, Los Angeles, California, USA, November 2003. ACM.
[7] D. Nyström et. al. Comet: A component-based real-time database for automotive systems. In *ICSE'04)*, Edinburgh, Scotland, May 2004. IEEE Computer Society Press.
[8] R. Lopez-Herrejon, D. Batory, and W. Cook. Ëvaluating Support for Features in Advanced Modularization Technologies. In *(ECOOP)*, 2005.
[9] D. L. Parnas. Designing Software for Ease of Extension and Contraction. *IEEE*, SE-5(2), 1979.
[10] Y. Smaragdakis and D. Batory. Mixin Layers: An Object-Oriented Implementation Technique for Refinements and Collaboration-Based Designs. *ACM TOSEM*, 11(2), 2002.
[11] A. Tešanović, K. Sheng, and J. Hansson. Application-tailored database systems: a case of aspects in an embedded database. In *IDEAS*, Coimbra, Portugal, July 2004. IEEE Computer Society Press.
[12] Alec Woo, Sam Madden, and Ramesh Govindan. Networking support for query processing in sensor networks. *Commun. ACM*, 47(6):47–52, 2004.

Integration of Web Data Sources: A Survey of Existing Problems

Le Quang Hieu

Institute of Computer Science
Heinrich-Heine-University Düsseldorf
D-40225 Düsseldorf, Germany
lqhieu@cs.uni-duesseldorf.de

Abstract

Integration of Web Data Sources is difficult because of the heterogeneous nature of the Web. A problem is that web data sources come in and come out frequently. Another problem is that an exact comparison between data elements in several data sources is not feasible since the data sources are owned by different organizations and therefore there are usually subtle differences in describing the same data. Or often queries need to access data sources in more than one domain. Furthermore new web standards and technologies such as XML, web services, the OWL Web Ontology Language give new opportunities and pose new challenges. In this paper, we investigate/survey on these problems and related studies as well as give our general ideas on how we could work on these problems.

1 Introduction

There are many web data sources available in the Internet. They are websites like Citeseer or Ebay that allow users to query information over certain criteria and answer users with data embedded in structured HTML pages. The web data sources have the following characteristics:

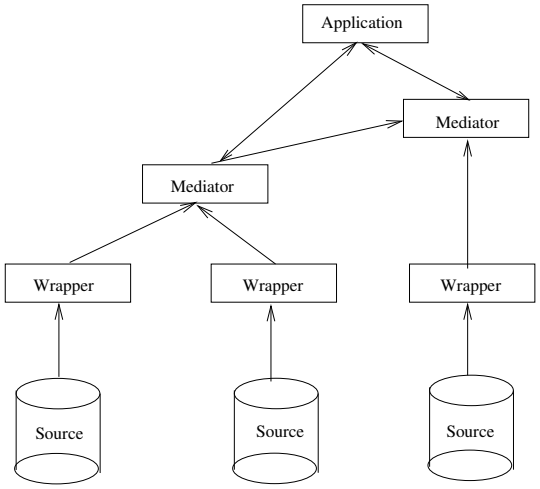
- Autonomous: Users have no control over the sources.
- Heterogeneous and Overlapping: Each source stores data in its own way, but the same kind of data is often stored in several sources; each source has different query capabilities.
- Frequently changing: Sources appear/disappear at a rapid rate; Data and layout of each source is updated continuously.
- The number of sources is very large, and it is increasing rapidly with the fast growth of the Internet.
- Distributed: sources are located all over the Internet which has an unpredictable condition. For example, connection between two points may be lost temporally or the connection speed varies from time to time. Moreover, there is no or little communication between the sources.

The integration of web data sources is to provide uniform access to multiple web data sources [9]. The integration of web data sources has many applications. It is also an active research field with many existing problems originated from the above web data sources' characteristics. The integration of web data sources is closely related to and sometimes has the similar meaning as "the integration of heterogeneous autonomous sources". For brevity, we will use "web data integration" for "integration of web data sources", and "souce" for "web data source" unless explicitly indicating otherwise.

In general web data integration obviously shares many problems and techniques of traditional legacy data integration. But it has its own set of problems and techniques. The common problems in web data integration are as follows:

- Selection of architecture and data model.
- Answering queries using views and query optimization.
- Source mapping and wrapper construction.

Figure 1: A mediator architecture for web data integration



- Object similarity identification.

These above problems in turn (except of the following one) form respectively the next sections of the paper before the conclusions. In each section, we will briefly review the problem, and then give our comments on that problem. The comments are our personal views of the strength and weakness of existing solutions, or of open problems under the impacts of XML, web services, the OWL Web Ontology Language.

The two problems of answering queries using views and query optimization are not addressed in this paper. They are also the two very important problems in web data integration. An comprehensive survey for the former is [6], while a brief review for the latter can be found in [5].

2 Architecture and Data Model

The selection of architecture and data model play a central role for web data integration. Here we selected three systems for our following review: TSIMMIS [1], Information Manifold (IM) [10] and Ariadne [7] because they are pioneer and successful systems in this area.

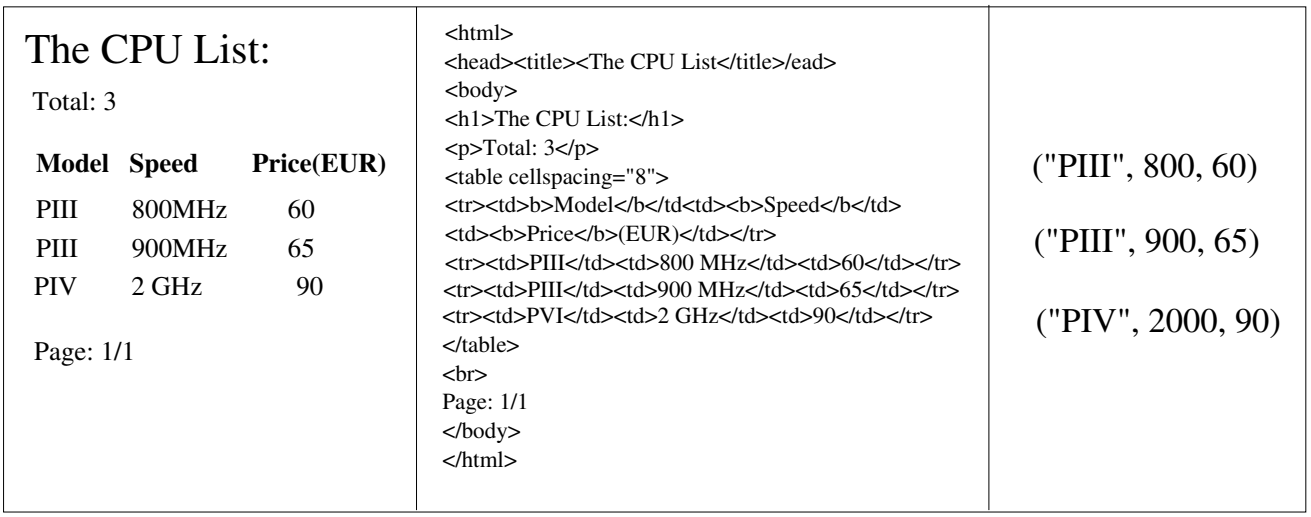
There three architectures for data integration are Federated system, Mediator system [15], and Data Warehouse. It is interesting that TSIMMIS and Ariadne use the Mediator system. Though IM system has a global mediated schema, its architecture can also be considered as a mediator system since the global schema is virtual, storing no data and the IM uses wrappers to get data from sources. Figure 1 illustrates a mediator system for web data integration.

The Mediator architecture is the most suitable for web data integration since it is the most flexible architecture. It can deal better with the autonomous and frequently changing web data sources. On the other hand, both the Federated system and the Data Warehouse can not cope with the problem of adding/removing data sources frequently, and they require certain control of data sources in case of the Data Warehouse or communication between data sources in case of the Federated system.

In contrast to the use of the architecture, the three systems devise and use three different data models. TSIMMIS has the OEM (Object Exchange Model) [12], that is a simple and self-describing data model, and supports object nesting and identity. TSIMMIS has also the OEM-QL which is a SQL-like language for OEM. The data model for IM is the relational model enhanced with object-oriented features. IM use this data model along with a record for each source to describe the content of the source and the source capabilities [10]. Finally Ariadne based on SIMS uses Loom to model domains. Loom is a knowledge representation language.

The differences of data models in the three systems shows that the choice of an adequate data model is an open problem. However, one interesting note is that TSIMMIS and IM have parts of their roots in

Figure 2: Data exaction process



database logic [14]. This note is also quite true for Ariadne because of its use of Loom. This fact may explain that database logic makes it easier to represent recursive relations and to re-formulate queries.

Among the three systems, IM emphasized most in the source content and capability description. However, the method used by IM may not suitable for complex sources such as Ebay. Ebay has many product categories with different sets of attributes. Therefore using the IM's method to describe Ebay contents will result in many complex views. Furthermore, the IM's method for capability descriptions is good for the sources accepting user queries in form of HTML pages, but is not applicable to sources exposed their content in form of web services like the Ebay's web service.

With the adoption of XML and especially OWL (Web Ontology Language), it is interesting to see how data models that have been created for web data integration evolve. Since XML and OWL are designed for data exchange over the Web, they have in many ways important impacts to web data integration. However OWL seems not suitable for web data integration since the data model for web data integration should be not only expressive but also simple [12] and efficient [9]. In the context of OWL adoption for the Web, a certain data model which could be evolved from the TSIMMIS's OEM may be a good one, partly because OEM is already a simple object-oriented model.

3 Source Mapping and Wrapper Construction

Source mapping or matching is a typical problem in data integration. That is given two schemas of two sources, find a mapping between elements of the two schemas. However, in web data integration, there is little source meta-data. Therefore sources must be learned more. The large number of web data sources makes the problem harder since manual mapping seems impractical. The source mapping should be as automated as possible. Many researches are towards that goal and they use approaches such as linguistics, constrain-based at different schema or instance level. An excellence survey of automatic schema mapping is [13].

Different from the source mapping problem, wrapper construction is a quite specific problem in web data integration. Figure 2 illustrates the problem.

Figure 2 shows that, a user asks for a list of CPUs cheaper than EUR 100. The user then gets the result as a list of CPUs in form a HTML page displayed in the left column. The HTML page is designed for humans, not for computers. The HTML page actually consists of the HTML tags, text and data shown in the middle column. To automate the extraction of data for a integration system, we need to build a wrapper that automatically parses the HTML page in the middle column to get the embedded data that are three tuples shown in the right column. As in the source mapping problem, the main problem in building wrappers is to make it as automated as possible. The reason is that there is a large

number of sources, and each source does not only frequently update its data but also its layout. There are also many approaches in the field, which use HTML-syntax analysis, Natural Language Processing, Wrapper induction, Modeling-based and so on. A good brief survey of wrapper construction is [8].

Among the approaches for source mapping, the approach in the LSD system [4] is an interesting one. In LSD, there are a meta-learner which is capable of combining the result of other learner. In that way the system may be able to extend to take advantage of other approaches.

An observation is that the increasing use of XML and OWL will ease the two problems. XML will make the process of extracting data a lot easier since XML allows the separation of data and presentation. XML and especially OWL will also reduce the difficulty in the process of schema mapping because of their self-description feature and the possibility of sharing the common data model and terminologies for a specific application domain by adopting OWL. However, the need for automatically wrapper building and source mapping remains because of the number and diversity of sources

Another observation is current methods of wrapper construction are not designed for web service data sources. On the other hand, the use of web services is increased quickly. Web sites, which make use of web services, are often large sites and store a very large amount of data such as Ebay(.com), Yahoo(.com). Among current wrapper construction approaches, the approach used in the XWRAP system [11] can be modified for web service source since it provide a framework with the separation of tasks for each source and tasks repetitive for any sources and a two-phase code generation.

4 Object Similarity Identification

Object similarity identification problem is that given two objects or data items of two sources, how to decide whether the two objects are "similar". This problem is quite similar to the source mapping problem, but harder. The problem is difficult for many reasons: how to define the 'similarity' meaning; the same data item is often stored in subtly different ways in two sources; the huge amount of data and so on.

There are not many researches in the problem. The two approaches frequently mentioned are [3] and [2]. The latter using textual similarity with techniques from Information Retrieval seems more suitable for web environment. One note is that the second approach is still being developed, while there are no new results for the first approach after the mentioned paper.

5 Conclusions

The Integration of Web Data Sources is a large research field. While it shares many problems with the traditional data integration, it also has its own set of problems because of the characteristics of the web data sources. Web data integration research is evolving with the development of the Web as well as of the Data Integration methods.

Currently web data integration only solves the problems relating to answering queries, but not with transactions between sources. The reason may be that most nowadays sources are autonomous. However, building a full featured data integration system over the Web could also a good way since it can take the advantages of Web infrastructure and matured technologies.

References

[1] Chawathe, S., Garcia-Molina, H., Hammer, J., Ireland, K., Papakonstantinou, Y., Ullman, J.D., Widom, J.: The TSIMMIS project: Integration of heterogeneous information sources. In: 16th Meeting of the Information Processing Society of Japan, pp. 7–18. Tokyo, Japan (1994)

[2] Cohen, W.W.: Integration of heterogeneous databases without common domains using queries based on textual similarity. pp. 201–212 (1998)

[3] D. McLeod D. Fang, J.H.: The identification and resolution of semantic heterogeneity. In: In Pro. First International Workshop on Interoperability in Multidatabase Systems. Kyoto, Japan (1991)

[4] Doan, A., Domingos, P., Levy, A.Y.: Learning source description for data integration. In: WebDB (Informal Proceedings), pp. 81–86 (2000)

[5] Florescu, D., Levy, A.Y., Mendelzon, A.O.: Database techniques for the world-wide web: A survey. SIGMOD Record **27**(3), 59–74 (1998)

[6] Halevy, A.Y.: Answering queries using views: A survey. VLDB Journal: Very Large Data Bases **10**(4), 270–294 (2001)

[7] Knoblock, C.A., Minton, S., Ambite, J.L., Ashish, P.J.M.N., Muslea, I., Philpot, A.G., Tejada, S.: Modeling web sources for information integration. In: Proc. Fifteenth National Conference on Artificial Intelligence (1998)

[8] Laender, A.H.F., Ribeiro-Neto, B.A., da Silva, A.S., Teixeira, J.S.: A brief survey of web data extraction tools. SIGMOD Rec. **31**(2), 84–93 (2002). DOI http://doi.acm.org/10.1145/565117.565137

[9] Levy, A.Y.: Logic-based techniques in data integration pp. 575–595 (2000)

[10] Levy, A.Y., Rajaraman, A., Ordille, J.J.: Querying heterogeneous information sources using source descriptions. In: Proceedings of the Twenty-second International Conference on Very Large Databases, pp. 251–262. VLDB Endowment, Saratoga, Calif., Bombay, India (1996)

[11] Liu, L., Pu, C., Han, W.: XWRAP: An XML-enabled wrapper construction system for web information sources. In: ICDE, pp. 611–621 (2000)

[12] Papakonstantinou, Y., Garcia-Molina, H., Widom, J.: Object exchange across heterogeneous information sources. In: P.S. Yu, A.L.P. Chen (eds.) 11th Conference on Data Engineering, pp. 251–260. IEEE Computer Society, Taipei, Taiwan (1995)

[13] Rahm, E., Bernstein, P.A.: A survey of approaches to automatic schema matching. VLDB Journal: Very Large Data Bases **10**(4), 334–350 (2001)

[14] Ullman, J.D.: Information integration using logical views. In: ICDT '97: Proceedings of the 6th International Conference on Database Theory, pp. 19–40. Springer-Verlag, London, UK (1997)

[15] Wiederhold, G.: Mediators in the architecture of future information systems. IEEE Computer **25**(3), 38–49 (1992)

Anfrageoptimierung in objektrelationalen Datenbanken mit bedingten Termersetzungen

Mazeyar E. Makoui, Udo W. Lipeck
{mem, ul}@dbs.uni-hannover.de
Universität Hannover

Zusammenfassung

Die bisherige Optimierung von Anfragebäumen mit Hilfe von Termersetzungen in der Relationenalgebra liefert durch bekannte Heuristiken (z. B. frühzeitiges Ausführen von Projektionen und Selektionen) die Grundlage eines Termersetzungssystems, welches aber kaum eine Möglichkeit für weitere Optimierungsstrategien zur Anfrageoptimierung in objektrelationalen Datenbanken bietet. In diesem Artikel werden algebraische Termersetzungen um kostenbasierte Bedingungen erweitert. Als Basis dafür dienen sowohl Metadaten der vorhandenen Relationen, als auch der einzusetzenden Operatoren und Zugriffsmethoden. Diese werden durch ein erweitertes Kostenmodell während der Optimierung berücksichtigt und verbessern damit den Entscheidungsprozess verschiedener Anfrageoptimierungsalgorithmen. Als Beispiel hierfür wird der von Vance/Meier [Van96] vorgeschlagene Join-Ordering-Algorithmus in seinen Suchmöglichkeiten stark reduziert und dadurch beschleunigt.

1 Einleitung

In objektrelationalen Datenbanken sind die althergebrachten Heuristiken zur Anfrageoptimierung hinsichtlich der frühen Ausführung von Selektionen und Projektionen in den meisten Fällen nicht mehr sinnvoll. Vielmehr sollten kostspielige Operationen wie z. B. Selektionen auf räumlichen Daten in einem Zugriffsplan später ausgeführt werden, also nicht nur auf Reduzierung der zu betrachtenden Tupel geachtet werden. Demzufolge sollte ein Optimierer Operatoren und deren Implementierungskosten auch in früheren Phasen mit in Betracht ziehen können. Folgende drei Aspekte wären ideal:

1. Alle möglichen Operatoren mit ihren verschiedenen Implementierungen sollten mit einbezogen werden.
2. Alle möglichen Reihenfolgen einzelner Operatoren müssen betrachtet werden können.
3. Es muss eine Basis dafür geliefert werden, dass beliebige Optimierungsstrategien angewendet werden können.

Dieses wird von dem Konzept der bedingten Termersetzungen ermöglicht, ohne dabei den kompletten Suchraum betrachten zu müssen.

Im folgenden Abschnitt wird das Konzept der bedingten Termersetzung vorgestellt und motiviert. Danach folgt die Herleitung einer bedingten Termersetzungsregel für den Selektionsoperator, um im dritten Abschnitt für den Join-Ordering-Algorithmus verwendet zu werden. Im vierten Teilabschnitt wird ein Beispiel präsentiert. Schließlich befasst sich der fünfte Abschnitt mit Erweiterungen und einem Ausblick der vorgestellten Arbeit.

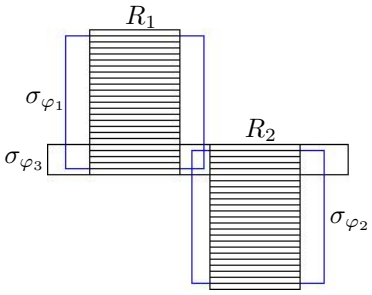
2 Konzept der kostenbedingten Termersetzungen

Die Idee der bedingten Termersetzung besteht darin, Termersetzungen nicht mehr rein heuristisch vorzunehmen, sondern von einer Kostenfunktion abhängig zu machen. Dabei erweitern wir die bisherigen algebraischen Ersetzungsregeln durch Bedingungen, so dass die Regeln auf Basis aller zusätzlichen Informationen, die in ein erweitertes Kostenmodell eingeflossen sind, alternative Ausführungspläne in Beziehung zueinander setzen können.

2.1 Motivation einer kostenbedingten Termersetzung

Gegeben seien zwei Relationen R_1 und R_2 und drei darauf agierende Selektionen ($\overline{\mathsf{R}}_1 := \sigma_{\varphi_1}(\mathsf{R}_1), \overline{\mathsf{R}}_2 := \sigma_{\varphi_2}(\mathsf{R}_2)$ und $\mathsf{R}_1 \bowtie_{\varphi_3} \mathsf{R}_2 = \sigma_{\varphi_3}(\mathsf{R}_1 \times \mathsf{R}_2)$). Die Joinbedingung ($\sigma_{\varphi_3}$), wie im folgenden Beispiel gezeigt, kann so stark selektierend sein, dass es sich nicht lohnt, erst die Ausgangsrelationen durch die zugehörigen Selektionen einzuschränken und dann zu joinen. Dabei soll vorausgesetzt werden, dass keine Indexe oder Sortierungen existieren, die man zusätzlich dazu berücksichtigen müsste (siehe nachfolgende Abbildung):

Beispiel für einen stark selektierenden Verbund:



Dazu seien folgende Statistiken gegeben[1]:

Selektivität		Kardinalität
$\mathsf{sel}(\varphi_1, \mathsf{R}_1)$	$= 0,90$	$\lvert\mathsf{R}_1\rvert = 30$
$\mathsf{sel}(\varphi_2, \mathsf{R}_2)$	$= 0,90$	$\lvert\mathsf{R}_2\rvert = 30$
$\mathsf{sel}(\varphi_3, \mathsf{R}_1 \times \mathsf{R}_2)$	$= 0,01$	
$\mathsf{sel}(\varphi_3, \overline{\mathsf{R}}_1 \times \overline{\mathsf{R}}_2)$	$= 0,01$	

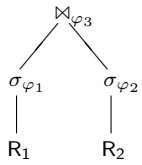
Die aus der Standardliteratur (siehe [Kem04, S.233]) bekannte Selektionsverschiebungsregel für Verbunde, hier L-Rule genannt:

L-Rule $$\tau_1 := \sigma_{\varphi_2}(\sigma_{\varphi_1}(\mathsf{R}_1 \bowtie_{\varphi_3} \mathsf{R}_2)) \rightarrow \tau_2 := \sigma_{\varphi_1}(\mathsf{R}_1) \bowtie_{\varphi_3} \sigma_{\varphi_2}(\mathsf{R}_2),$$
$$\textbf{falls } \mathsf{attr}(\varphi_1) \subseteq \mathsf{sch}(\mathsf{R}_1) \textbf{ und } \mathsf{attr}(\varphi_2) \subseteq \mathsf{sch}(\mathsf{R}_2)$$

liefert laut Heuristik immer den Ausführungsplan, der zuerst die Basisrelationen selektiert und dann erst den Verbund berechnet. Genau diese Regel scheint aber hier nicht zu passen.

Nachfolgend betrachten wir beide allgemein möglichen Ausführungspläne τ_1 und τ_2. Dabei benutzen wir ein Kostenmodell (siehe dazu [Mak03]), das nur die Anzahl der Tupel, die in den Join „hineingehen“ und ihn wieder „verlassen“, in Beziehung zueinander setzt. Vorausgesetzt, dass die zu selektierenden Tupel gleichverteilt in den Selektionen vorkommen und somit die Selektivitäten von σ_{φ_1} und σ_{φ_2} konstant bleiben, folgt:

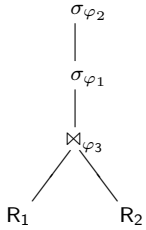
- 1. Heuristisch optimierter Ausführungsplan τ_1:



1. $\lvert\overline{\mathsf{R}}_1\rvert = \mathsf{sel}(\varphi_1, \mathsf{R}_1) \cdot \lvert\mathsf{R}_1\rvert = 0,9 \cdot 30 = 27$
2. $\lvert\overline{\mathsf{R}}_2\rvert = \mathsf{sel}(\varphi_2, \mathsf{R}_2) \cdot \lvert\mathsf{R}_2\rvert = 0,9 \cdot 30 = 27$
3. $\lvert\overline{\mathsf{R}}_1 \bowtie_{\varphi_3} \overline{\mathsf{R}}_2\rvert = \mathsf{sel}(\varphi_3, \overline{\mathsf{R}}_1 \times \overline{\mathsf{R}}_2) \cdot \lvert\overline{\mathsf{R}}_1\rvert \cdot \lvert\overline{\mathsf{R}}_2\rvert$
$= 0,01 \cdot (27 \cdot 27) = 7,29$

$\Rightarrow \mathsf{cost}(\tau_1) = 27 + 27 + 7,29 = 61,29$

- 2. Nicht heuristisch optimierter Ausführungsplan τ_2:



1. $\lvert\mathsf{R}_1 \bowtie_{\varphi_3} \mathsf{R}_2\rvert = \mathsf{sel}(\varphi_3, \mathsf{R}_1 \times \mathsf{R}_2) \cdot \lvert\mathsf{R}_1\rvert \cdot \lvert\mathsf{R}_2\rvert$
$= 0,01 \cdot (30 \cdot 30) = 9$
2. $\mathsf{sel}(\varphi_1, \mathsf{R}_1) \cdot \lvert\mathsf{R}_1 \bowtie_{\varphi_3} \mathsf{R}_2\rvert = 0,9 \cdot 9 = 8,1$
3. $\mathsf{sel}(\varphi_2, \mathsf{R}_2) \cdot \lvert\mathsf{sel}(\varphi_1, \mathsf{R}_1) \cdot \lvert\mathsf{R}_1 \bowtie_{\varphi_3} \mathsf{R}_2\rvert\rvert = 0,9 \cdot 8,1 = 7,29$

$\Rightarrow \mathsf{cost}(\tau_2) = 9 + 8,1 + 7,29 = 24,39$

Unter den vorangehenden Einschränkungen muss τ_2 - also ohne Anwendung der Standardheuristik - weniger als $39,79\,\%$ der Tupel von τ_1 berücksichtigen.

2.2 Herleitung einer kostenbedingten Termersetzungsregel

Angestrebt wird jetzt eine Ersetzungsregel $\tau_1 \rightarrow \tau_2$ mit einer Bedingung, die $\mathsf{cost}(\tau_1) \geq \mathsf{cost}(\tau_2)$ gewährleistet. Um diese Bedingung vorab auszurechnen und angemessen zu vereinfachen, vergleichen wir die Kostenformeln der beiden Ausführungspläne:

$$\begin{aligned}\mathsf{cost}(\tau_1) &= \mathsf{sel}(\varphi_1, \mathsf{R}_1) \cdot \lvert\mathsf{R}_1\rvert + \mathsf{sel}(\varphi_2, \mathsf{R}_2) \cdot \lvert\mathsf{R}_2\rvert + \mathsf{sel}(\varphi_3, \overline{\mathsf{R}}_1 \times \overline{\mathsf{R}}_2) \cdot (\mathsf{sel}(\varphi_1, \mathsf{R}_1) \cdot \lvert\mathsf{R}_1\rvert \cdot \mathsf{sel}(\varphi_2, \mathsf{R}_2) \cdot \lvert\mathsf{R}_2\rvert) \\ &= \mathsf{sel}(\varphi_1, \mathsf{R}_1) \cdot \lvert\mathsf{R}_1\rvert + \mathsf{sel}(\varphi_2, \mathsf{R}_2) \cdot \lvert\mathsf{R}_2\rvert + \mathsf{sel}(\varphi_1, \mathsf{R}_1) \cdot \mathsf{sel}(\varphi_2, \mathsf{R}_2) \cdot \mathsf{sel}(\varphi_3, \overline{\mathsf{R}}_1 \times \overline{\mathsf{R}}_2) \cdot \lvert\mathsf{R}_1\rvert \cdot \lvert\mathsf{R}_2\rvert\end{aligned}$$

$$\begin{aligned}\mathsf{cost}(\tau_2) &= \mathsf{sel}(\varphi_3, \mathsf{R}_1 \times \mathsf{R}_2) \cdot \lvert\mathsf{R}_1\rvert \cdot \lvert\mathsf{R}_2\rvert + \mathsf{sel}(\varphi_1, \mathsf{R}_1) \cdot (\mathsf{sel}(\varphi_3, \mathsf{R}_1 \times \mathsf{R}_2) \cdot \lvert\mathsf{R}_1\rvert \cdot \lvert\mathsf{R}_2\rvert) + \mathsf{sel}(\varphi_2, \mathsf{R}_2) \cdot (\mathsf{sel}(\varphi_1, \mathsf{R}_1) \cdot \\ &(\mathsf{sel}(\varphi_3, \mathsf{R}_1 \times \mathsf{R}_2) \cdot \lvert\mathsf{R}_1\rvert \cdot \lvert\mathsf{R}_2\rvert)) = (\mathsf{sel}(\varphi_3, \mathsf{R}_1 \times \mathsf{R}_2) \cdot \lvert\mathsf{R}_1\rvert \cdot \lvert\mathsf{R}_2\rvert) \cdot (1 + \mathsf{sel}(\varphi_1, \mathsf{R}_1) \cdot (1 + \mathsf{sel}(\varphi_2, \mathsf{R}_2)))\end{aligned}$$

[1] Das Bild liefert zur besseren Visualisierung nicht maßstabsgetreu die Selektivitäten, die in der nachfolgenden Statistik angenommen werden.

Da wir die Situation suchen, in der beide Ausführungspläne die gleichen Kosten besitzen, setzen wir die beiden Kostenfunktionen gleich:

$$\mathsf{cost}(\tau_1) = \mathsf{cost}(\tau_2)$$

Aufgelöst nach $\mathsf{sel}(\varphi_3, \mathsf{R}_1 \times \mathsf{R}_2)$ ergibt sich die Grenzselektivität $\mathsf{g}_{\mathsf{sel}}(\varphi_1, \varphi_2, \varphi_3, \mathsf{R}_1, \mathsf{R}_2)$, bei der beide Ausführungspläne die gleichen Kosten verursachen:

$$\mathsf{g}_{\mathsf{sel}}(\varphi_1, \varphi_2, \varphi_3, \mathsf{R}_1, \mathsf{R}_2) := \frac{|\mathsf{R}_1| \cdot (|\mathsf{R}_2| \cdot \mathsf{sel}(\varphi_2, \mathsf{R}_2) \cdot \mathsf{sel}(\varphi_3, \overline{\mathsf{R}}_1 \times \overline{\mathsf{R}}_2) + 1) \cdot \mathsf{sel}(\varphi_1, \mathsf{R}_1) + |\mathsf{R}_2| \cdot \mathsf{sel}(\varphi_2, \mathsf{R}_2)}{|\mathsf{R}_1| \cdot |\mathsf{R}_2| \cdot (\mathsf{sel}(\varphi_1, \mathsf{R}_1) \cdot (\mathsf{sel}(\varphi_2, \mathsf{R}_2) + 1) + 1)}$$

Wir haben jetzt eine Ersetzungsregel, die bedingt angewandt werden kann:

L2-Rule

$$\begin{array}{l|l} \sigma_{\varphi_2}(\sigma_{\varphi_1}(\mathsf{R}_1 \bowtie_{\varphi_3} \mathsf{R}_2)) \to \sigma_{\varphi_1}(\mathsf{R}_1) \bowtie_{\varphi_3} \sigma_{\varphi_2}(\mathsf{R}_2), & \sigma_{\varphi_1}(\mathsf{R}_1) \bowtie_{\varphi_3} \sigma_{\varphi_2}(\mathsf{R}_2) \to \sigma_{\varphi_2}(\sigma_{\varphi_1}(\mathsf{R}_1 \bowtie_{\varphi_3} \mathsf{R}_2)), \\ \text{wenn } \mathsf{sel}(\varphi_3, \mathsf{R}_1 \times \mathsf{R}_2) \geq \mathsf{g}_{\mathsf{sel}}(\varphi_1, \varphi_2, \varphi_3, \mathsf{R}_1, \mathsf{R}_2) & \text{wenn } \mathsf{sel}(\varphi_3, \mathsf{R}_1 \times \mathsf{R}_2) \leq \mathsf{g}_{\mathsf{sel}}(\varphi_1, \varphi_2, \varphi_3, \mathsf{R}_1, \mathsf{R}_2) \end{array}$$

Bislang sind wir davon ausgegangen, dass in jedem Zeitpunkt alle Selektivitäten bekannt sind. Das ist aber in der Realität nicht so. Vielmehr besitzen wir in unserem konkreten Fall, wenn eine Selektion an einem Join vorbei geführt werden soll, die Selektivität des eigentlichen Joins nicht mehr, da sie sich in diesem Fall verändern kann. Aus diesem Grund müssen wir eine Unabhängigkeit der einzelnen Selektionen annehmen und können dazu die alte Selektivität des Joins für die Berechnung benutzen. Somit wird im Folgenden $\mathsf{sel}(\varphi_3, \mathsf{R}_1 \times \mathsf{R}_2) = \mathsf{sel}(\varphi_3, \overline{\mathsf{R}}_1 \times \overline{\mathsf{R}}_2)$ gesetzt. In unserem Beispiel ergibt sich für den rechten Kostenterm:

$$\mathsf{g}_{\mathsf{sel}}(\varphi_1, \varphi_2, \varphi_3, \mathsf{R}_1, \mathsf{R}_2) = \frac{30 \cdot (30 \cdot \frac{9}{10} \cdot \frac{1}{100} + 1) \cdot \frac{9}{10} + 30 \cdot \frac{9}{10}}{30 \cdot 30 \cdot (\frac{9}{10} \cdot (\frac{9}{10} + 1) + 1)} = \frac{681}{27100} \approx 0,025129$$

Mit $\mathsf{sel}(\varphi_3, \mathsf{R}_1 \times \mathsf{R}_2) = \frac{1}{100}$ folgt, da $0,01 = \frac{1}{100} < \frac{681}{27100} \approx 0,025129$, der richtige Ausführungsplan, da die bedingte Regel dazu rät, die Selektionen erst nach dem Join anzuwenden.

3 Erweiterung des Join-Ordering-Algorithmus um Selektionen

Im Folgenden zeigen wir wie der Join-Ordering-Algorithmus von Vance/Meier mit der L2-Rule beschleunigt werden kann. Dazu beschreiben wir zunächst die Arbeitsumgebung:

Zu optimieren ist ein Term bestehend aus natürlichen Joins und Selektionen über einer Menge von n verschiedenen Basisrelationen $\mathcal{R} = \{\mathsf{R}_\mathsf{i} \mid \mathsf{i} = 1, ..., \mathsf{n}\}$. Die Selektionen lassen sich darstellen als eine Menge von (konjunktiv verknüpfbaren, möglichst atomaren) Selektionsbedingungen $\mathcal{P} = \{\varphi_\mathsf{j} \mid \mathsf{j} = 1, ..., \mathsf{m}\}$, von denen jede sich auf eine oder mehrere Basisrelationen bezieht. Wenn eine Bedingung φ nur Attribute einer Teilmenge $\widetilde{\mathsf{R}}$ von $\mathcal{R}$ enthält, schreiben wir $(\widetilde{\mathsf{R}}, \varphi) \in \mathcal{P}$.

Gegenstand der Optimierung ist einerseits die Anordnung der Joins und andererseits die Positionierung der Selektionen, die nach den Überlegungen im Abschnitt 2 eben nicht grundsätzlich wie bei der üblichen Heuristik so früh wie möglich ausgeführt werden sollten.

Die entstehenden *Optimierungsterme* können dann rekursiv dargestellt werden als $\sigma_\Phi(\tau_{\mathsf{left}} \bowtie \tau_{\mathsf{right}})$, wobei Φ eine Menge äußerer Selektionsbedingungen (und-verknüpft) ist, welche leer sein kann (entspricht „true“), und τ_{left}, τ_{right} wieder Terme aus den oben genannten Operationen sind, die alle anderen Selektionsbedingungen enthalten. Basisterme sind von der Form $\sigma_\varphi(\mathsf{R}_\mathsf{i})$, falls $(\varphi, \mathsf{R}_\mathsf{i}) \in \mathcal{P}$, oder $\sigma_{\mathsf{true}}(\mathsf{R}_\mathsf{i})$ wobei R_i eine Basisrelation ist.

Wir orientieren uns an dem in [Van96] vorgeschlagenen Vorgehen der Optimierung von Join-Reihenfolgen durch dynamische Programmierung, bei dem aus den Optimierungstermen für immer größer werdende Teilmengen von $\mathcal{R}$ schließlich ein optimaler Term konstruiert wird. Wir arbeiten allerdings mit durch Selektionen geschachtelten Mengen von Relationen, um auch die Positionierung der Selektionen zu optimieren.[2]

Dazu wird eine Tabelle `plantable` aufgebaut, deren Einträge durch solche eventuell geschachtelten Mengen von Relationen indiziert sind; ein Eintrag liefert den zugehörigen Optimierungsterm durch Angabe der äußeren Selektionsbedingungen `selections` (oben: Φ) und des Eintrags `leftSide` für den linken Teilterm (oben: τ_{left}, aber in Mengendarstellung); der Eintrag für den rechten Teilterm (oben: τ_{right}) ist daraus rekonstruierbar.

Zusätzlich werden zu jedem Eintrag - zur schrittweisen Berechnung - folgende Attribute festgehalten:

[2] Alle Erweiterungen gegenüber dem Original-Algorithmus werden nachfolgend blau dargestellt.

- `cardinality` die Kardinalität der durch den Optimierungsterm dargestellten Relation
- `selectivity` die Selektivität äußerer Joinbedingungen des Optimierungsterms[3]
- `cost` die Gesamtkosten des Optimierungsterms

Input: Relationenmenge $\mathcal{R}$, Menge aller Selektionsbedingungen $\mathcal{P}$
Output: Liefert den optimierten Anfrageplan

```
optimize(Relationenmenge R, Bedingungsmenge P) {
  // Folgender Abschnitt ist die Eintragung der Basisrelationen
  // in die planTable:
  planTable := ∅;
  for each Ri ∈ R do {
    planTable[Ri].cardinality := |Ri|;
    planTable[Ri].leftSide := ∅;
    // κ(τ) liefert die Kosten eines Operators, wobei
    // Indexe und Sortierungen berücksichtigt werden.
    planTable[Ri].cost := κ(σ_true(Ri));
    planTable[Ri].selectivity := 1;
    planTable[Ri].selections := ∅;
    if (S ≠ ∅) {
    S := {φ | (Ri, φ) ∈ P} alle Bedingungen, die auf Ri agieren;
      ψ sei deren Konjunktion
      planTable[σψ{Ri}].cardinality := |σψ(Ri)|;
      planTable[σψ{Ri}].leftSide := ∅;
      planTable[σψ{Ri}].cost := κ(σψ(Ri));
      planTable[σψ{Ri}].selectivity := 1;
      planTable[σψ{Ri}].selections := S;
    }
  }

  // Im folgenden Abschnitt findet die eigentliche Opt. statt:
  for (jede Teilmenge R̃ := R̃1 ∪ R̃2, R̃1 ∩ R̃2 = ∅, für die
    zu R̃1 und R̃2 Einträge in der planTable[R̃] existieren{
    planTable[R̃].cardinality := |R̃|;
    planTable[R̃].leftSide := find_best_split(R̃);
    planTable[R̃].cost := planTable[τleft].cost +
                         planTable[τright].cost + κ(R̃1 ⋈ψ R̃2);
    P̃ := natürliche Verbundbedingungen für R̃
    planTable[R̃].selectivity := compute_selectivity(P̃);
    planTable[R̃].selections := ∅;
    S := {φ | (R̃, φ) ∈ P und φ kommt in R̃ noch nicht vor}
    alle weiteren Bed., die auf R̃ agieren; ψ sei deren Konj.
    if (Eintrag R̃ ist optimierbar mit L2-Regel) {
      remove(R̃, planTable);
    }
    // Entfernt Einträge, bei denen die Selektion zu "tief"
    // im Baum ist; z. B. wird R̃ = {R, σρ{S, T}} daraufhin ge-
    // prüft, ob der Term R ⋈ σρ(S ⋈ T) nach der L2-Rule zu
    // σρ(R ⋈ (S ⋈ T)) optimiert wurde; dann kann der Eintrag
    // gelöscht werden, weil zu σρ{R, S, T) an anderer Stelle ein
    // Eintrag eingeführt wird.
    // Hier folgt die Betrachtung von aufsteigenden Selektionen:
    if (S ≠ ∅) {
      planTable[σψ{R̃}].cardinality := |σψ{R̃}|;
      planTable[σψ{R̃}].leftSide := planTable[R̃].leftSide;
      planTable[σψ{R̃}].cost := κ(σψ{R̃}) + planTable[R̃].cost;
      planTable[σψ{R̃}].selectivity := 1;
      planTable[σψ{R̃}].selections := S;
      // Entfernt Einträge, bei denen die Selektion zu "hoch"
      // im Baum ist. Hier wird geprüft, ob im Term
      // σψ(R ⋈ σρ(S ⋈ T)) die Selektion absinken
      // kann; dann gäbe es andere Einträge.
      if (Eintrag σψ{R̃} ist optimierbar mit L2-Regel) {
        remove(σψ{R̃}, planTable);
      }
    }
  }
  return get_result(R, planTable);
}
```

```
compute_selectivity(Verbundbedingungen P̃) {
  s := 1;
  for each φ1 ∈ P do {
    s = s · sel(φ1);
  }
  return s;
}
```

Input: Relationenmenge $\tilde{R}$
Output: Relationenmenge $\tilde{R}_{best}$ (Beste linke Seite des Vebundes leftSide)

```
// Kombiniert alle möglichen nicht leeren Teilmengen der überge-
// benen Relationenmenge und liefert die Kombination zurück, die
// insgesamt die geringsten Kosten verursacht.
find_best_split(Relationenmenge R̃) {
  R̃best := ∅;
  best_cost := ∞;
  for each R̃1, ∅ ⊂ R̃1 ⊂ R̃ do {
    R̃2 := R̃ \ R̃1;
    cost = compute_cost(R̃1, R̃2);
    if (cost ≤ best_cost) {
      best_cost = cost;
      R̃best = R̃;
    }
  }
  return R̃best;
}
```

Durch die Anwendung der L2-Rule lassen sich Einträge in der Optimierungstabelle löschen, die dann beim Erzeugen von weiteren Einträgen nicht berücksichtigt werden müssen. Somit ist dieser Algorithmus auch schneller als die von [Ste97], [Sch97] und [Sch98] vorgeschlagene Optimierung mit Hilfe von Ranking-Verfahren, da unser Algorithmus schon bei der Erzeugung weniger Teilbäume betrachten muss.

4 Beispiel

Gegeben sei folgende Anfrage

$$\pi_{\mathsf{P.Name}}\left(\sigma_{\mathsf{S.Semester}='5'}\left(\mathsf{H} \bowtie_{\mathsf{H.MatrNr}=\mathsf{S.MatrNr}} \mathsf{S} \bowtie_{\mathsf{V.VorlNr}=\mathsf{H.VorlNr}} \mathsf{P} \bowtie_{\mathsf{P.ID}=\mathsf{V.gelesenVon}} \mathsf{V}\right)\right)$$

für die nachfolgenden Relationen und Metadaten[4]

H(<u>VorlNr</u>, MatrNr),

Metadaten	Größe
|H|	10.000
#(MatrNr, H)	2000
#(VorlNr, H)	300
|I(MatrNr, H)|	3
|I(VorlNr, H)|	3

S(<u>MatrNr</u>, Semester),

Metadaten	Größe
|S|	2000
#(MatrNr, S)	2000
#(Semester, S)	20
|I(MatrNr, S)|	3
|I(Semester, S)|	3

P(<u>ID</u>, Name),

Metadaten	Größe
|P|	100
#(ID, P)	100
#(Name, P)	75
|I(ID, P)|	2

V(<u>VorlNr</u>, gelesenVon)

Metadaten	Größe
|V|	300
#(VorlNr, V)	300
#(gelesenVon, V)	100
|I(VorlNr, V)|	2

Unser erweiterter Join-Ordering-Algorithmus erzeugt folgende Einträge:

[3] Dabei wird davon ausgegangen, dass sie entweder gegeben wird oder berechnet werden kann.

[4] Die Selektivität der gegebenen Selektionsbedingung berechnet sich durch den Kehrwert der Kardinalität des dazugehörigen Attributes; also $\mathsf{sel}(\mathsf{S.Semester} = '5') = \frac{1}{\#(\mathsf{Semester,S})} = \frac{1}{20}$.

Relation Set	cardinality	leftSide	cost	selectivity	jointype	selections
{H}	10.000	∅	10	1	Basisrelation	-
{P}	100	∅	10	1	Basisrelation	-
{S}	2.000	∅	200	1	Basisrelation	S.Semester =′ 5′
{V}	300	∅	30	1	Basisrelation	-
{σ(S)}	100	∅	106	1	Index-Selektion	-
{H, P}	1.000.000	{P}	21.012	1	Nested-Loop-Join	-
{H, V}	10.000	{V}	10.736	0.00333	Index-Index-Join	-
{H, S}	pruned	-	-	-	-	-
{H, σ(S)}	500	{σ(S)}	639	0.0005	Rel-Index-Join	-
{P, V}	300	{P}	349	0.01	Rel-Index-Join	-
{P, σ(S)}	10.000	{σ(S)}	1.318	1	Nested-Loop-Join	-
{P, S}	pruned	-	-	-	-	-
{S, V}	pruned	-	-	-	-	-
{V, σ(S)}	30.000	{σ(S)}	3.742	1	Nested-Loop-Join	-
{σ(H, S)}	pruned	-	-	-	-	-
{σ(P, S)}	pruned	-	-	-	-	-
{σ(S, V)}	pruned	-	-	-	-	-
{H, P, S}	pruned		-	-	-	-
{H, P, V}	10.000	{P, V}	10.780	0.00003	Rel-Index-Join	-
{H, P, σ(S)}	50.000	{H, σ(S)}	6.649	0.0005	Nested-Loop-Join	-
{H, S, V}	pruned		-	-	-	-
{H, V, σ(S)}	500	{H, σ(S)}	1.170	0	Rel-Index-Join	-
{H, σ(P, S)}	pruned	-	-	-	-	-
{H, σ(S, V)}	pruned	-	-	-	-	-
{P, S, V}	pruned	-	-	-	-	-
{P, σ(H, S)}	pruned	-	-	-	-	-
{P, σ(S, V)}	pruned	-	-	-	-	-
{P, V, σ(S)}	30.000	{P, V}	4.061	0.01	Nested-Loop-Join	-
{V, σ(H, S)}	pruned	-	-	-	-	-
{V, σ(P, S)}	pruned	-	-	-	-	-
{σ(H, P, S)}	pruned	-	-	-	-	-
{σ(H, S, V)}	pruned	-	-	-	-	-
{σ(P, S, V)}	pruned	-	-	-	-	-
{H, P, V, σ(S)}	500	{H, V, σ(S)}	1.701	0	Rel-Index-Join	-
{H, P, σ(S, V)}	pruned	-	-	-	-	-
{H, V, σ(P, S)}	pruned	-	-	-	-	-
{H, σ(P, S, V)}	pruned	-	-	-	-	-
{P, V, σ(H, S)}	pruned	-	-	-	-	-
{P, σ(H, S, V)}	pruned	-	-	-	-	-
{V, σ(H, P, S)}	pruned	-	-	-	-	-
{σ(H, P, S, V)}	pruned	-	-	-	-	-

Die mit „pruned“ bezeichneten Kardinalitäten geben an, welche Teilmengen nicht mehr weiter in die Berechnung einfließen. Natürlich liefert das bisherige Join-Ordering dasselbe Ergebnis. Dabei muss es aber 41 Einträge (entsprechend obiger geschachtelten Mengen) anstatt 16 erzeugen, um das Optimum zu finden. Das ist eine Ersparnis von $\frac{25}{41} \approx 61\%$ gegenüber dem Original-Algorithmus.

5 Erweiterungen und Ausblick

Ebenso wie die bedingten Termersetzungsregel für Selektionen, wird die entsprechende Ersetzungsregel für Projektionen ohne Duplikateliminierung zur Korrektur der Kosten eingesetzt. Dabei muss allerdings ein Kostenmodell eingesetzt werden, welches nicht nur die Anzahl von Tupeln, sondern auch deren Länge bzw. deren Speicherbedarf in Seiten betrachtet.

Aus diesem Grund genügt das einfache Kostenmodell aus [Van96] nicht mehr und muss durch ein anderes Modell ersetzt werden, welches Lesen und Schreiben von Seiten betrachtet (siehe dazu [War04]). Danach müssen drei Fälle untersucht werden:[5]

1. Das Einfügen einer Projektion verringert lokal sofort die Verbundkosten, da nicht mehr alle Attribute betrachtet werden müssen und somit mehr Tupel auf eine Seite passen.

2. Das frühe Einfügen einer Projektion senkt zwar nicht lokal die Kosten eines Teilbaumes des Ausführungsplanes (es kann ihn sogar verteuern), aber global werden die Kosten für weitere Verbunde so weit verringert, dass es sich insgesamt rentiert.

[5] Aus Platzgründen wird auf die komplette Herleitung einer bedingten Termersetzungsregel für die Projektion verzichtet. Vielmehr soll hier eine skizzenhafte Einführung geboten werden.

3. Das Einfügen ist in jedem Fall mit höheren Kosten verbunden und sollte nicht in Betracht gezogen werden.

Aus diesen Grund muss man das Einfügen einer Projektion in Beziehung zu den noch weiter im Ausführungsplan benötigten Attributen setzen:

τ	:=	originaler Ausführungsplan
$\overline{\tau}$	:=	optimierter Ausführungsplan
$\overline{\mathsf{attr}}(\tau)$	:=	noch benötigte Attribute von τ
$\mathsf{neededattr}(\mathsf{R})$	:=	liefert die im darüber liegenden Ausführungsplan noch ben. Attr.

M2-Rule

$$\mathsf{R}_1 \bowtie_\varphi \mathsf{R}_2 \rightarrow \pi_{\mathsf{l}_1}(\mathsf{R}_1) \bowtie_\varphi \pi_{\mathsf{l}_2}(\mathsf{R}_2),$$
$$\textbf{wenn } \mathsf{cost}(\tau) > \mathsf{cost}(\overline{\tau}) \textbf{ und}$$
$$\mathsf{l_i} := \mathsf{sch}(\mathsf{R_i}) \cap \mathsf{neededattr}(\mathsf{R_i}) \textbf{ mit } \mathsf{i} = 1, 2$$

Verglichen mit der L2-Rule kann diese Regel erst angewandt werden, wenn der gesamte Ausführungsplan generiert worden ist. Somit können Teilbäume nur optimiert werden, wenn man parallel zum Aufbau des Ausführungsplanes auch eine Menge von noch benötigten Attributen mitführt ($\mathsf{neededattr}(\mathsf{R})$).

Die Funktion $\mathsf{neededattr}(\mathsf{R})$ wird mit Hilfe einer simulierten Projektion berechnet, die man anfangs auf den kompletten Teilbaum anwendet. Dabei werden alle Attribute sowohl von noch ausstehenden Selektionen und Joins, als auch der abschließenden Projektion in einer Korrekturmenge zusammengefasst, welche die Projektionen auf die Teilbäume einschränkt bzw. erweitert ($\pi_{\mathsf{sch}(\mathsf{R}_1 \bowtie_\varphi \mathsf{R}_2) \cap \overline{\mathsf{attr}}(\tau) - \mathsf{sch}(\varphi)}(\mathsf{R}_1 \bowtie_\varphi \mathsf{R}_2)$).

Danach wird top-down das Schema genutzt, um die Rückgabewerte der Funktion $\mathsf{neededattr}(\mathsf{R})$ zu berechnen. Daraufhin wird die Anfangsprojektion herausgenommen und man erhält die einzelnen lokalen Projektionen. Abschließend berechnet man den Kostenvorteil bzw. Nachteil jeder einzelnen Projektion relativ zum gesamten Ausführungsplan. Hierfür muss man das Join-Ordering mit Hilfe der L2-Rule einmal anwenden, um daraus einen Kostenwert bzw. einen Ausführungsplan zu bekommen, mit dem man vergleichen kann.

Ein klares Ziel unserer Arbeit ist es, weitere Operatoren mit in die Optimierung einfließen zu lassen. Dazu gehören nicht nur reine Implementierungsarten, sondern auch zusammengesetzte relationale Operatoren wie der Semi- bzw. Antisemijoin. Er bietet bei allen Anfragen, bei denen eine Reduzierung einer Menge gefordert ist, ein immenses Optimierungspotential.

Durch das hier vorgestellte Konzept eines Termersetzungssystems zur Anfrageoptimierung sollte man den Antisemi- bzw. Semijoin als weiteren Operator mit eigens dazu gehörenden Kosten und Regeln hinzufügen können. Schwierig zu beantworten ist dabei nur die Frage, wie sich dieser Operator anwenden lässt. Dabei müssen teilweise Projektionen zusätzlich eingefügt und somit betrachtet werden. Klar ist auch, dass die zu betrachtenden Zwischenmengen wieder größer werden und wiederum durch bedingte Termersetzungen reduziert werden müssen.

Die in diesem Artikel vorgestellten bedingten Termersetzungen wurden im Zuge von Diplom- ([Mak03]), BSc- ([War04],[Die04]) und Studienarbeiten ([Zle05]) hinsichtlich eines Simulators für Anfrageoptimierung implementiert. Näheres zum Programm `RELOpt` kann man unter `www.dbs.uni-hannover.de/~makoui/-RELOpt.html` erfahren.

Literatur

[Van96] **Bennet Vance / David Maier:** Rapid Bushy Join-order Optimization with Cartesian Products, p. 35-46, SIGMOD 1996

[Ste97] **Michael Steinbrunn / Guido Moerkotte / Alfons Kemper:** Heuristic and randomized optimization for the join ordering problem, VLDB Journal, p. 191-208, 1997

[Sch97] **Wolfgang Scheufele / Guido Moerkotte:** On the Complexity of Generating Optimal Plans with Cross Products, PODS, p. 238-248, 1997

[Sch98] **Wolfgang Scheufele / Guido Moerkotte:** Efficient Dynamic Programming Algorithms for Ordering Expensive Joins and Selections, EDBT, p. 201-215, 1998

[Mak03] **Mazeyar E. Makoui:** Heuristische Anfrageoptimierung in Relationalen Datenbanken, Diplomarbeit, Institut für Informationssysteme, Universität Hannover, 2003

[War04] **Hendrik Warneke:** Erweiterung eines Simulators für relationale Anfrageoptimierungen, Bachelorarbeit, Institut für Informationssysteme, Universität Hannover, 2003

[Die04] **Moritz Diehle:** Erweiterung eines relationalen Anfragesimulators um eine regelbasierte Steuerung von physischen Optimierungsregeln, Bachelorarbeit, Institut für Informationssysteme, Universität Hannover, 2004

[Kem04] **Alfons Kemper / Andre Eickler:** Datenbankssysteme - Eine Einführung, Oldenbourg Verlag, 5. Auflage, 2004

[Zle05] **Alexander Zlenko:** Implementierung von Termersetzungsregeln zur regelgesteuerten Anfrageoptimierung in relationalen Datenbanken, Studienarbeit, Institut für Informationssysteme, Universität Hannover, 2005

A Query Processing Approach for XML Database Systems

Christian Mathis, Theo Härder

University of Kaiserslautern
{mathis | haerder}@informatik.uni-kl.de

Abstract: Besides the storage engine, the query processor of a database system is the most critical component when it comes to performance and scalability. Research on query processing for relational database systems developed an approach which we believe should also be adopted for the newly proposed XML database systems. It includes a syntactic and semantic analyzation phase, the mapping onto an internal query representation, algebraic and cost-based optimization, and finally the execution on a record-oriented interface. Each step hides its own challenges and will therefore be discussed throughout this paper. Our contribution can be understood as a road-map that reveals a desirable set of functionalities for an XML query processor.

1 Introduction

After relational, network-based, hierarchical, object-oriented, object-relational, and deductive database systems, academic research and businesses raise their attention to database-driven processing of XML documents, resulting in a new kind of information system, namely the *(native) XML database system* (XDBS). This development is reasonable, because for the management of a possibly large collection of XML documents, the classical advantages of database systems over file systems still hold: convenient use of XML data through a standardized application programming interface (API); transactional warranties for all operations on XML data; processing of large volumes of data, measured in number of documents as well as document size.
In [1], Michael Haustein outlines the realization for a subset of these concepts—the *XML Transaction Coordinator* (XTC), a native XDBS. Currently, XTC provides an internal node interface called taDOM, which includes the features of the Document Object Model enhanced with user transactions. Every sequence of DOM operations can be encapsulated by a transaction and can thus benefit from the ACID warranties. For declarative language access, an XQuery processor resides on top of this interface. Its implementation follows the concepts given in the XQuery formal semantics, thereby neglecting important optimization techniques, which were crucial for the success of relational database systems in the past thirty years.
Throughout this paper, we focus on the problem of query evaluation for declarative XDBS access using XQuery. Our contribution can be understood as a road-map that reveals a desirable set of functionalities for the XTC query processor.

2 Levels of Abstraction in XML query processing

To handle the complexity of query processing, several levels of abstraction between a declarative query expression and its procedural evaluation using a set of low-level operations can be identified. These levels are depicted in Table 1. To facilitate comprehension, the new XML-related concepts are compared to their well-known counterparts of relational query processing. The most abstract view of a query is its formulation in a way that only describes the desired result in a certain declarative language. The same query may be represented using an algebra expression, whose operators express the query in the *Logical Access Model*. Optimization techniques at this level only rely on the expression itself, but do not cope

Level of Abstraction	XDBS	RDBS
Language Model	XQuery	SQL
Logical Access Model	XML Query Algebra	Relational Algebra
Physical Access Model	Physical XML Query Algebra	Physical DB-Operators
Storage Model	XTC, Natix, Shredded Documents	Record-oriented DB-Interface

Table 1. XML Query processing abstraction levels

with system-specific information. In general, this is the task of the layer below—the *Physical Access Model*. Finally, the bottom layer accomplishing the storage of XML documents plays also an important role, because the efficiency of operations is critically dependent on the chosen storage structure. An explicit separation of this abstraction level helps to cope with mapping requirements when multiple heterogenous storage models are present. Each of the depicted layers has its own associated tasks for query evaluation and hides its own challenges. Therefore, we will elaborate on them.

2.1 The Language Model

So far, several declarative XML query languages have been proposed, among them Lorel, XML-QL, XML-GL, and XQuery. A survey of these languages in [2] singles XQuery out as the most universal language, measured by the demands posed in [3][1]. Furthermore, XQuery is likely to be standardized by the W3C and will therefore presumably play a similar role for XML data as SQL does for relational data. The language was designed to meet the demands of both the "document-centric" community—notably text search functionality and document-order awareness—and the "data-centric" community—expecting powerful selections and transformations[2]. The design efforts resulted in a complex, strongly typed language allowing nested subexpressions at almost every position. Contribution [4] shows that XQuery has the same expressive power as μ-recursive functions, and is thus Turing-Complete. Because of an inherent trade-off between expressiveness and evaluation complexity, the question which sublanguages of XQuery may effectively be evaluated, gains significant importance. For XPath, Gottlob et. al. answered this question in [5] stating that XPath is evaluable in polynomial time and space. However, for special extensions of XPath towards complete XQuery, this complexity determination is still an open problem.

```
<result>{
   for $dep in doc('dept.xml')//dept,
       $emp in doc('emp.xml')//emp
   where $dep/@depnr = $emp/@depnr and
       $dep/loc = 'Kaiserslautern'
       and $emp/sal > '50000'
   return
       <person>{
          $pers/name,
          $pers/sal
       }</person>
}</result>
```

Figure 1. Example Query

Furthermore, several practical problems regarding the language model arise: the syntactic and semantic analysis of a query and its transformation into a convenient internal representation to be used throughout the subsequent optimization steps. As observed in [2], XQuery may be syntactically analyzed using an LR parser. The semantic processing requires a specific phase for static type and reference checking to recognize user errors as early as possible. As demanded in [6], an internal query representation should be efficiently accessible, flexible w. r. t. subsequent transformation steps, and should reflect a kind of procedural evaluation strategy. So far, we have only dealt with operator trees and we may legitimately ask whether there is the need for a further refinement of the internal representation structure. We conclude this section with our running XQuery example depicted in Figure 1. Given the departments and

[1] However, unlike Lorel, XQuery does not support document modification operations, which certainly will be added in future versions.

[2] These expectations resemble the different qualities represented by the object-oriented model for "vertical search" and the relational model for "horizontal search"

employees of an organization in the documents `dept.xml` and `emp.xml`, it returns a list of all persons who work in Kaiserslautern and earn more than 50000.

2.2 The Logical Access Model

After the transformation of a query into its internal representation, the optimization phase can begin. In general, optimization goals are the reduction of query processing time or the maximization of throughput. The main obstacle is the possibly large number of equivalent evaluation strategies for a given query, originating from varying operator orders, different operator implementations, the existence of indices, and so on. In a first step, the query may be optimized regarding only the logical query structure, neglecting all further system-specific issues. This process is called non-algebraic optimization or query restructuring. Its key idea is to minimize intermediary results by executing the most selective operations as early as possible. To achieve this goal, the query has to be transformed in a semantics-preserving way. A general approach to identify these transformations is the use of algebra expressions onto which queries are mapped.

$$e_1 = \sigma_{[\phi_{loc}(\$dept)=\text{"}KL\text{"}\wedge\phi_{sal}(\$emp)>50000]}$$
$$\Big[\beta_{\$dept:\phi_{//dept}}(S(\text{"}dept.xml\text{"}))$$
$$\bowtie_{[\phi_{@depnr}(\$dept)=\phi_{@depnr}(\$emp)]}$$
$$\beta_{\$emp:\phi_{//emp}}(S(\text{"}emp.xml\text{"}))\Big]$$

$$C_{result}(C_{pers,\phi_{name}(\$emp)\circ\phi_{sal}(\$emp)}(e_1))$$

Figure 2. Algebraic Expression

To facilitate the operation tracking of our running example across the various abstraction levels, we will use our own algebra notation here, although several others have been published over the past years. Figure 2 contains the algebraic equivalence for the query in Figure 1. Each algebra operator relies on a set of ordered sequences of tuples, depending on the arity of the operator, and produces a single result sequence containing n-ary tuples. For example, a binary join operator processes two input sequences and produces one output sequence containing composed tuples.

$$e_1 = \Big[\beta_{\$dept:\phi_{//dept[loc=\text{"}KL\text{"}]}}(S(\text{"}dept.xml\text{"}))$$
$$\bowtie_{[\phi_{@depnr}(\$dept)=\phi_{@depnr}(\$emp)]}$$
$$\beta_{\$emp:\phi_{//emp[sal>50000]}}(S(\text{"}emp.xml\text{"}))\Big]$$

$$C_{result}(C_{person,\phi_{name}(\$emp)\circ\phi_{sal}(\$emp)}(e_1))$$

Figure 3. Optimized Expression

The expression in Figure 2 can be read as follows: The innermost operator S ("source") provides the document node of the specified documents in a singleton sequence. Then the *follow* operator (ϕ) evaluates the specified relative path expressions and its result sequence is bound (β) to the variables $\$dep$ and $\$emp$. Afterwards, the join operator can be evaluated, generating a sequence of binary tuples which obeys the join order. The *selection* operator filters these tuples by its predicate before the result is produces by two construct operators (C). So far, various proposals for an XQuery algebra have not lead to a standardization. In fact, the published approaches differ a lot in the following features: underlying data model, operator input and output format, existence of a designated evaluation operator (like ϕ) for XPath, handling of order, expressive power, representation of XQuery expressions, treatment of query nesting, etc. Nevertheless, each approach identifies certain algebraic equivalence rules which permit a heuristics-based optimization to reduce the intermediate result. This heuristics is primarily based on selection push down. Figure 3 holds an optimized version of the query in Figure 2, where the selection predicate is split up and embedded in the two path expressions that generate the departments and employees of interest.

2.3 The Physical Access Model

At the next lower level of abstraction, system-specific issues become visible. Each operator of the logical algebra can be composed of one or more physical operators. Those operators embody a specific evaluation algorithm that possibly relies on the existence of indices, document structure, and element order. The overall goal during this step of optimization—called query transformation or non-algebraic optimization—is a query execution plan (QEP) for which appropriate physical operators have to be cho-

sen and subsequently arranged in a sequential manner. As in the relational case, we expect the various system-dependent parameters to span a large search space for possible QEPs.
In Figure 3 two critical parts of the non-algebraic optimization can be identified: the path expressions and the join operator. Because XPath is a little bit older than XQuery[3], it has been the subject of more intense research. Several evaluation strategies can be pointed out: The *pure algorithmic evaluation* as presented by Gottlob et. al. relies on a dynamic programming technique avoiding duplicate nodes, which may be produced by most axes[4] and, thus, may cause the repeated evaluation of certain path steps. The algorithm is the basis for a complexity estimation which reveals that XPath may be evaluated in combined complexity of $O(|D|^4 * |Q|^2)$, where $|D|$ is the size of the input document and $|Q|$ the size of the query. This strategy may be chosen, when no indices are present. However, it remains to be explored whether or not the presented algorithm can benefit from index support.
A second alternative is the *index-based evaluation*. For example, the T-Index tries to evaluate a path expression at once, i. e., without further decomposition. Because path expressions can extremely vary and it seems utopian to support each type of expression by a single index, the appearance of an expression has to be limited using path templates. Only those queries that match such templates can be evaluated using the associated index. The knowledge of which kinds of expression have to be supported may either be derived from the structure of the underlying documents (using heuristics) or has to be provided from outside (e. g., by the database administrator). Another possible application area could be the use of ad-hoc indices.
Finally, further *algebraic techniques* as in [7] may be applied. This approach is listed for completeness only, because it actually belongs to the logical access model. As part of the mapping of a query expression to an algebraic equivalence, all path expressions are also decomposed into operators which can be considered under non-algebraic optimization.
Creating indices for a class of path expressions seems to be too restrictive. Therefore, we believe that a combination of all three strategies results in a conceivable solution. XPath expressions should be included in the algebraic optimization and further processed keeping the findings of Gottlob et. al. in mind. To speed up certain common evaluation tasks, indices may be used.
In relational query processing, there are three major physical operator classes for the join operation: nested-loop-, sort-merge-, and hash-based algorithms. When considering a join in XQuery, order plays an important role, because XML documents are inherently ordered by their textual representation. Therefore, when a sort-merge join or hash join is used—both do not obey document order in general—or when the join order is altered to minimize the intermediary result size, a sort operator has to process the generated result sequence. We will explore whether there are further circumventions of this non-commutativity and whether their cost is lower than the quadratic bound of the nested-loop join.

2.4 Storage Model

A critical input for the QEP optimizer is the cost model, because it builds the foundation of the QEP rating. It has to include information about the following four issues: I/O costs where the (physical) page references should be counted rather than the (logical) node references; CPU costs reflecting the processor usage during query execution; storage costs for intermediary results and, finally, communication costs, which are especially relevant in a distributed system environment. Of course, the overall costs of a QEP can only be estimated, and because many different system properties as well as document-related statistics (meta-data) have to be taken into account, there will be a trade-off between cost model accuracy and meta-data maintenance overhead. If, e. g., the system gathers data about the names and occurrences of child nodes for each node in the document, rather than only the average number of children for that type of node, the selectivity estimation of a child axis in a path expression may be more accurate. However, when the document is modified, the more accurate information becomes obsolete much earlier, requiring a recalculation of the meta-data. Therefore “stable” meta-data items have to be identified.

[3] XPath was released as a standard by the W3C in 1999 whereas the first working draft of XQuery was proposed in 2001.
[4] The parent axis is an example for this problem.

So far, several alternatives for DBS-supported storing of XML documents have been explored, from simple LOBs (*Large Objects*), over certain XML-to-relational mappings (*shredding*), as well as the use of object-oriented DBS, to specifically tailored (native) storage formats. Certainly, the costs for I/O and CPU usage heavily depend on the underlying storage model. The evaluation of the path expression `//departement[location="KL"]` from Figure 3 requires a possibly large number of physical page lookups, depending on how many pages have to be fetched to evaluate a child axis. In turn, this number depends on information contained in the specific storage model like, for example, the node numbering scheme and the storage layout of the document. Therefore, if the QEP evaluator is parameterized by the different cost models resulting from the specific storage models of heterogenous XDBS, we are able to optimize a query on behalf of each of those systems.

3 Conclusion

In this article, we introduced an approach to XML query evaluation, inspecting queries on four different layers of abstraction. For each layer, we highlighted several tasks: On the layer of the *Language Model* we have chosen XQuery. Because of its complexity, a first step is the search for an "optimizable" sublanguage. Furthermore, a suitable internal representation has to be found. For the *Logical Access Model* various XML query algebras have been proposed so far, but none has lead to a standard. A comparative survey of these algebras with respect to expressive power, underlying data model, and the set of equivalence rules has still to be done. For our own algebra, we have shown a brief example together with its algebraic optimization. Each operator of the logical access model has to be mapped onto its implementation in the *Physical Access Model* regarding information about existing index structures, object orders, document structure, etc. Interchangeable physical operators lead to different QEPs that have to be rated using a cost model. Because path expressions and joins are frequent operations, their effective implementation is crucial. Finally, cost models are heavily influenced by the assumed *Storage Model*. Parameterizing the rating of QEPs by cost models enables query optimization on behalf of different XML database systems.
In the future, we will focus our work on the two lowermost layers. Therefore we will assume a limited algebra that handles only central XQuery constructs like path expressions, joins, etc. Regarding the physical access model, we will elaborate on effective operator implementation, index support, and optimal operator order. Furthermore, the creation of a QEP enumeration and rating framework, customizable by different cost models, is our aim.

References

[1] Michael P. Haustein. Eine XML-Programmierschnittstelle zur transaktionsgeschützten Kombination von DOM, SAX und XQuery. In *11. GI-Fachtagung für Datenbanksysteme in Business, Technologie und Web (BTW)*, 2005. `http://wwwdvs.informatik.uni-kl.de/pubs/papers/Hau04.BTW.html`.

[2] Christian Mathis. Anwendungsprogrammierschnittstellen für XML-Datenbanksysteme. Master's thesis, Kaiserslautern University of Technology, 2004. (german only).

[3] David Maier. Database Desiderata for an XML Query Language, 1998. `http://www.w3.org/TandS/QL/QL98/pp/maier.html`.

[4] Stephan Kepser. A Simple Proof for the Turing-Completeness of XSLT and XQuery. In *Extreme Markup Languages 2004, Montreal Quebec*, 2004. `http://tcl.sfs.uni-tuebingen.de/~kepser/papers/EML2004Kepser01.pdf`.

[5] G. Gottlob, C. Koch, and R. Pichler. Efficient Algorithms for Processing XPath Queries. In *Proc. of the 28th International Conference on Very Large Data Bases (VLDB 2002)*, 2002.

[6] Bernhard Mitschang. *Anfrageverarbeitung in Datenbanksystemen, Entwurfs- und Implementierungskonzepte*. Vieweg Verlag, 1995. (german only).

[7] Matthias Brantner, Sven Helmer, Carl-Christian Kanne, and Guido Moerkotte. Full-fledged Algebraic XPath Processing in Natix. Technical report, University of Mannheim, 2005.

SQL-basierte Datenbankzugriffe und XML: Verarbeitung von Anfrageergebnissen in Anwendungsprogrammen

Thomas Müller
Friedrich-Schiller-Universität Jena
Lehrstuhl für Datenbanken und Informationssysteme
Ernst-Abbe-Platz 2
07743 Jena
mueller@informatik.uni-jena.de

Zusammenfassung

Die sich noch in Vorbereitung befindende nächste Version der SQL-Norm erlaubt erstmals Anfrageergebnisse, die komplette XQuery-Sequenzen enthalten. Im vorliegenden Beitrag wird ein Beispielszenario vorgestellt, bei dem es sinnvoll ist, Ausschnitte der im SQL-Anfrageergebnis enthaltenen XQuery-Sequenzen direkt im Anwendungsprogramm zu verarbeiten. Der Beitrag beschreibt den entsprechenden Verarbeitungsablauf und diskutiert grundsätzliche Möglichkeiten für die Programmiersprachenrepräsentation der Sequenzausschnitte.

1 Einführung

SQL:2003 [ISO03b], die aktuelle Version der SQL-Norm, hat „XML“ als neuen Basisdatentyp eingeführt [MKF+03, Tür03, ISO03a]. Damit wurde es auch aus Sicht der Norm möglich, Tabellen anzulegen, die in einer oder mehreren Spalten XML-Dokumente [KM03, W3C04] enthalten können. Diese Normerweiterung zielte darauf ab, einen SQL-basierten integrierten Zugriff auf XML-Daten und traditionelle (objekt)relationale Daten zu ermöglichen. Die Wichtigkeit eines solchen SQL-basierten Zugriffs wurde in [Mül04] und anderenorts erläutert.

In der gerade entstehenden, auf SQL:2003 folgenden SQL-Version — im Weiteren als *SQL:200x* bezeichnet[1] — wird die XML-Unterstützung weiter ausgebaut [EM04]. Spalten des Datentyps XML sollen dann anstelle von einfachen XML-Dokumenten auch komplette XQuery-Sequenzen [LS04, W3C05] enthalten können. Zudem wird die SQL-Funktion *XMLQuery* [EM04] eingeführt, mit der es möglich ist, XQuery-Anfragen innerhalb von SQL zu formulieren und auszuwerten.

Ein SQL:200x-Anfrageergebnis kann außer Spalten „herkömmlicher Datentypen“ auch Spalten des neuen XML-Datentyps besitzen. Die Ergebnistupel enthalten in diesem Fall als Attributwerte komplette XQuery-Sequenzen. In bestimmten Situationen ist es aus Performance- bzw. Handhabbarkeitsgründen sinnvoll, diese XQuery-Sequenzen (bzw. allgemeiner: *Teile* dieser XQuery-Sequenzen) direkt im Anwendungsprogramm zu verarbeiten. Es sollen also Ausschnitte der im Anfrageergebnis enthaltenen XQuery-Sequenzen ins Anwendungsprogramm übertragen werden, um dort für eine Verarbeitung mit Mitteln der Programmiersprache zur Verfügung zu stehen. Denkbar ist dabei einerseits ein rein lesender Zugriff auf die ins Anwendungsprogramm übertragenen Sequenzteile und andererseits die Möglichkeit, Änderungen an diesen Sequenzteilen vorzunehmen. Im letzteren Fall soll vom Anwendungsprogramm entschieden werden können, ob die vorgenommenen Änderungen in die Datenbank (wieder)eingebracht werden sollen. Dies setzt natürlich voraus, dass das zugrunde liegende SQL:200x-Anfrageergebnis änderbar ist, d. h. dass die Änderungen auf die zugrunde liegende(n) Basistabelle(n) rückabbildbar sind.

Der vorliegende Beitrag konzentriert sich auf den zuvor beschriebenen Fall, dass Teile von in SQL:200x-Anfrageergebnissen enthaltenen XQuery-Sequenzen direkt im Anwendungsprogramm

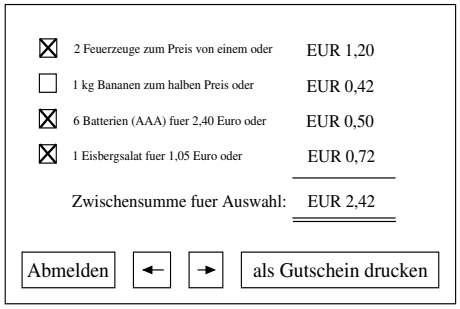
[1] Man rechnet mit einem Erscheinen etwa 2007.

verarbeitet werden sollen. Im Abschnitt 2 wird zunächst ein Beispielszenario eingeführt, anhand dessen im Abschnitt 3 der prinzipielle Verarbeitungsablauf beschrieben wird. Im Abschnitt 4 werden verschiedene grundsätzliche Varianten für die Programmiersprachenrepräsentation von Sequenzausschnitten diskutiert. Abschnitt 5 fasst den Beitrag zusammen und umreißt kurz die aktuellen und künftigen Forschungsschwerpunkte.

2 Beispielszenario Kundenkartenverwaltung

Im Mittelpunkt des Beispielszenarios „Kundenkartenverwaltung“ steht ein (fiktives) Unternehmen, welches Kundenkarten ausgibt und verwaltet [Mar05]. Die Kundenkarten können bei Einkäufen, Autoanmietungen, Hotelübernachtungen etc. eingesetzt werden, um Rabatte bzw. Sonderkonditionen zu erhalten. Die Kundenkartenverwaltung arbeitet hierfür mit verschiedenen Partnerunternehmen (Supermarktketten, Autovermietern, Hotels usw.) zusammen. Während die Kunden vor allem an den Preisnachlässen und Sonderangeboten interessiert sind, steht für die Partnerunternehmen das „Anlocken“ und Binden von Kunden im Vordergrund. Die Kundenkartenverwaltung hingegen sammelt detaillierte Informationen über das Einkaufsverhalten. Diese Informationen sind beispielsweise für die Planung gezielter Werbeaktionen von Interesse.

Als Anreiz zum Kauf von Kerzen kann ein Supermarkt seinen Kunden z. B. folgendes Bonusangebot unterbreiten: *„Wer für mindestens 6 Euro Kerzen kauft — und seine Kundenkarte vorlegt — bekommt beim nächsten Einkauf zwei Feuerzeuge zum Preis von einem.“* Solche (i. d. R. zeitlich befristeten) Bonusangebote können auch einen „einlösbaren Gegenwert“ haben, z. B. 1,20 Euro bei einem Folgeeinkauf im selben Geschäft oder 0,30 Euro bei einem Folgeeinkauf in einem anderen Geschäft. Der Kunde kann nun an einem Kundenterminal im Supermarkt auf seine bisher erworbenen Bonusangebote zugreifen und dabei gezielt ein oder mehrere Bonusangebote auswählen und deren aufsummierten Gegenwert als Gutschein ausdrucken (Abbildung 1a). Die entsprechenden Bonusangebote werden dann als „verbraucht“ markiert und verlieren ihre Gültigkeit.



a) Kundenterminal im Supermarkt

Kunde	KdNr	Jahr	Umsatz	Bonus
	4711	2003		
	4711	2004		
	4711	2005		
	4712	2002		

b) Tabelle „Kunde“

Abbildung 1: Kundenterminal und Kundentabelle

Alle gesammelten Daten werden von der Kundenkartenverwaltung mit Hilfe eines SQL:200x-fähigen relationalen Datenbanksystems verwaltet. Beispielsweise sind die Umsätze und Bonusangebote der einzelnen Kunden jahresweise in der Tabelle „Kunde“ gespeichert (Abbildung 1b). Die eigentlichen Umsatz- und Bonusdaten liegen dabei in Form von XQuery-Sequenzen in Spalten des SQL:200x-Datentyps XML. Jede XQuery-Sequenz der Spalte „Umsatz“ enthält jeweils alle vom entsprechenden Kunden im entsprechenden Jahr getätigten Kundenkartenumsätze. Jede in der Spalte „Bonus“ abgelegte XQuery-Sequenz enthält im Normalfall sowohl mehrere gültige als auch mehrere bereits als verbraucht markierte Bonusangebote, wobei für die Anzeige am Kundenterminal aber nur die Menge der *gültigen* Bonusangebote — also eine Modifikation der ursprünglichen XQuery-Sequenz — von Interesse ist.

Der Kunde soll seine am Kundenterminal vorgenommene Auswahl flexibel per Mausklick ändern können. Die angezeigte Zwischensumme soll dabei unmittelbar aktualisiert werden. Dies bedeutet, dass die Addition der einzelnen Gegenwerte direkt im Anwendungsprogramm des Kundenterminals erfolgen muss. Die Alternative — das Stellen einer separaten SQL/XMLQuery-Datenbankanfrage für jede vorgenommene Auswahländerung — kommt hier aus Performance-, Last- und Handhabbarkeitsgründen nicht in Frage.

3 Prinzipieller Verarbeitungsablauf

Zunächst wählt das Anwendungsprogramm mittels einer SQL:200x-Anfrage (möglicherweise unter Verwendung der XMLQuery-Funktion) alle relevanten Daten aus. Möchte beispielsweise der Kunde mit der Kundennummer 4711 auf seine einlösbaren Bonusangebote zugreifen, würde eine Anfrage gegen die Kunden-Tabelle gestellt. Dabei würden für diesen Kunden die Jahresangaben sowie jeweils alle *gültigen* Bonusangebote ausgewählt. Die Auswahl der gültigen Bonusangebote könnte dabei mit Hilfe der XMLQuery-Funktion realisiert werden, welche es erlaubt, eine geeignete XQuery-Anfrage auf den Werten der Bonus-Spalte auszuführen, um so die ungültigen Bonusangebote auszufiltern. Die Anfrage und das resultierende Anfrageergebnis sind in der Abbildung 2 angedeutet.

```
SELECT K.Jahr, XMLQUERY('$b...' PASSING K.Bonus AS "b") AS Bonusangebote
FROM   Kunde K
WHERE  K.KdNr = 4711
```

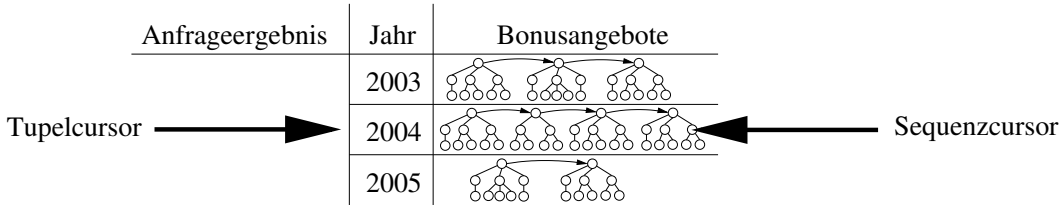


Abbildung 2: SQL:200x-Anfrage und Anfrageergebnis

Das Anwendungsprogramm kann das Anfrageergebnis nun — wie allgemein üblich — mittels (Tupel-)Cursor tupelweise durchlaufen. Zusätzlich soll es dem Anwendungsprogramm jetzt möglich sein, mittels neu eingeführter „Sequenzcursor“ durch die im aktuellen Tupel enthaltenen XQuery-Sequenzen zu navigieren [Rab05]. Da das Anfrageergebnis im konkreten Beispiel nur *eine* XML-Spalte besitzt, existiert hier nur ein Sequenzcursor. Dieser (vom Tupelcursor im Sinne einer Vater-Kind-Beziehung abhängige) Cursor erlaubt es, in die Bonusangebote-Sequenz des aktuellen Tupels „einzutauchen“ bzw. durch diese Sequenz zu navigieren.

Beim Navigieren durch eine möglicherweise sehr große Sequenz soll vom Anwendungsprogramm entschieden werden können, welche Sequenzteile ins Anwendungsprogramm übertragen werden sollen. Das Übertragungsgranulat kann dabei von einzelnen atomaren Werten bis hin zu kompletten (Teil-)Bäumen variieren. So kann zum einen beispielsweise nur der einzelne Knoten, auf dem der Sequenzcursor gerade positioniert ist, ins Anwendungsprogramm übertragen werden oder es wird der komplette Teilbaum übertragen, auf dessen Wurzel der Cursor steht.

Das Ziel besteht darin, aus Aufwands- und Handhabbarkeitsgründen nur die Teile einer XQuery-Sequenz ins Anwendungsprogramm zu übertragen, die auch aktuell zur Verarbeitung benötigt werden. Im konkreten Anwendungsszenario könnten beispielsweise zunächst (nur) so viele Bonusangebote ins Anwendungsprogramm übertragen werden, wie sich gleichzeitig am Kunden-Terminal anzeigen lassen. In Abhängigkeit von der Kundenreaktion kann die Navigation bzw. Übertragung dann bei Bedarf fortgesetzt werden.

4 Programmiersprachenrepräsentation von Sequenzteilen

Die ins Anwendungsprogramm übertragenen Sequenzausschnitte sollen im Anwendungsprogramm in einer solchen Art und Weise repräsentiert werden, dass eine effiziente und der Problemstellung angemessene Verarbeitung möglich ist. Für die Programmiersprachenrepräsentation der Sequenzausschnitte existieren die folgenden drei prinzipiellen Möglichkeiten:

1. Standard-Repräsentation
   - Es existiert eine „fest-verdrahtete“ Abbildungsvorschrift, anhand derer die Sequenzausschnitte automatisch in ihre Programmiersprachenrepräsentation überführt werden (und umgekehrt). Für jeden denkbaren Sequenzausschnitt gibt es dabei jeweils genau eine Entsprechung in der Programmierssprache.
   - Der Vorteil dieses Vorgehens liegt darin, dass sich der Anwendungsprogrammierer nicht um die Details der Umsetzung kümmern muss. Dies ist allerdings unmittelbar mit dem Nachteil verbunden, dass der Anwendungsprogrammierer keinerlei Einfluss auf die Programmiersprachenrepräsentation der Sequenzausschnitte nehmen kann. Eine Anpassung der Programmiersprachenrepräsentation an die konkrete Problemstellung ist somit nicht möglich.
2. Nutzerdefinierte Repräsentation
   - Der Anwendungsprogrammierer legt völlig selbstständig und „frei“ fest, wie die Sequenzausschnitte im Anwendungsprogramm repräsentiert werden sollen. Zudem muss er geeignete Mechanismen (beispielsweise in Form von Methoden oder Prozeduren) bereitstellen, mit denen es möglich ist, die Sequenzausschnitte in die gewünschte Programmiersprachenrepräsentation zu überführen (und umgekehrt).
   - Dieser Ansatz erlaubt eine sehr große Flexibilität und gestattet es, die Programmiersprachenrepräsentation optimal der Problemstellung anzupassen. Der gravierende Nachteil besteht jedoch darin, dass der Anwendungsprogrammierer für die Programmiersprachenumsetzung vollkommen allein verantwortlich ist. Dies hat sowohl einen enorm großen (Programmier-)Aufwand als auch eine hohe Fehleranfälligkeit zur Folge.
3. Anpassbare Repräsentation
   - Es handelt sich hierbei um einen Kompromiss zwischen den beiden zuvor beschriebenen Ansätzen. Der Anwendungsprogrammierer kann zwischen verschiedenen vorgegebenen Varianten einer Programmiersprachenrepräsentation auswählen und mit Hilfe diverser Optionen und Parameter die Details dieser Repräsentation festlegen.
   - Bei diesem Ansatz ist es für den Anwendungsprogrammierer möglich, eine auf die Problemstellung abgestimmte Programmiersprachenrepräsentation der Sequenzausschnitte zu nutzen, ohne dass er sich um die programmiertechnischen Details der Umsetzung kümmern muss.

Wegen der mangelnden Flexibilität einer *Standard-Repräsentation* und der Unhandlichkeit der (rein) *nutzerdefinierten Repräsentation* sollen in künftigen Forschungsarbeiten vor allem die Möglichkeiten einer *anpassbaren Repräsentation* von Sequenzausschnitten betrachtet werden.

5 Zusammenfassung und Ausblick

Die SQL-Normierungsgremien sind momentan dabei, die XML-Unterstützung in SQL gegenüber SQL:2003 wesentlich auszubauen. In der künftigen SQL-Version dürfen Anfrageergebnisse komplette XQuery-Sequenzen enthalten. Aus Performance- bzw. Handhabbarkeitsgründen kann es

sinnvoll sein, Ausschnitte dieser XQuery-Sequenzen direkt im Anwendungsprogramm zu verarbeiten. Das entsprechende Vorgehen wurde im Beitrag anhand des Beispielszenarios einer Kundenkartenverwaltung skizziert. Darüber hinaus wurden drei grundsätzliche Varianten für eine Programmiersprachenrepräsentation der Sequenzausschnitte diskutiert, wobei sich die *anpassbare Repräsentation* als die am viel versprechenste erwies.

Das Konzept der Sequenzcursor und weitere Möglichkeiten sollten im Rahmen künftiger Forschungsarbeiten konkretisiert und prototypisch umgesetzt werden. Dabei sind geeignete Navigationstechniken zu entwickeln und zu evaluieren. Ferner muss untersucht werden, wie die Sequenzausschnitte im Anwendungsprogramm unter Beachtung von Performance- und Handhabbarkeitsgesichtspunkten zu repräsentieren sind. Dabei sollte vor allem der Ansatz der anpassbaren Repräsentation weiter verfolgt werden.

Literatur

[EM04] A. Eisenberg und J. Melton. Advancements in SQL/XML. *SIGMOD Record*, 33(3):79–86, 2004.

[ISO03a] International Organization for Standardization, Genf. *Information technology – Database languages – SQL – Part 14: XML–Related Specifications (SQL/XML)*, Dezember 2003. ISO/IEC 9075-14:2003, International Standard (IS).

[ISO03b] International Organization for Standardization, Genf. *Information technology – Database languages – SQL – Part 2: Foundation (SQL/Foundation)*, Dezember 2003. ISO/IEC 9075-2:2003, International Standard (IS).

[KM03] M. Klettke und H. Meyer. *XML & Datenbanken : Konzepte, Sprachen und Systeme.* dpunkt-Verlag, Heidelberg, 1. Auflage, 2003.

[LS04] W. Lehner und H. Schöning. *XQuery — Grundlagen und fortgeschrittene Methoden.* dpunkt-Verlag, Heidelberg, 1. Auflage, 2004.

[Mar05] R. Marhold. Konzeption eines SQL:2007-Anwendungsszenarios. Diplomarbeit, Institut für Informatik, Friedrich-Schiller-Universität Jena. In Vorbereitung, 2005.

[MKF+03] J.-E. Michels, K. Kulkarni, C. M. Farrar, A. Eisenberg und N. Mattos. The SQL Standard. In *it – Information Technology*, 1/2003, Seiten 30–38. Oldenbourg Wissenschaftsverlag, München, Februar 2003.

[Mül04] T. Müller. SQL-basierte Datenbankzugriffe und XML: Klassifizierung von Anwendungsprogrammen. In M. Samia und S. Conrad (Hrsg.), *Tagungsband zum 16. GI-Workshop über Grundlagen von Datenbanken*, Seiten 88–92, Monheim am Rhein, Juni 2004. Institut für Informatik, Heinrich-Heine-Universität Düsseldorf.

[Rab05] G. Rabinovitch. Strategien für die Übergabe von XML-Werten zwischen Datenbanksystem und Anwendungsprogramm. Diplomarbeit, Institut für Informatik, Friedrich-Schiller-Universität Jena. In Vorbereitung, 2005.

[Tür03] C. Türker. *SQL:1999 & SQL:2003 : objektrelationales SQL, SQLJ & SQL/XML.* dpunkt-Verlag, Heidelberg, 1. Auflage, 2003.

[W3C04] World Wide Web Consortium. *Extensible Markup Language (XML) 1.0 (Third Edition)*, Februar 2004. W3C Recommendation.

[W3C05] World Wide Web Consortium. *XQuery 1.0 and XPath 2.0 Data Model*, Februar 2005. W3C Working Draft.

Modellierung und Entwicklung von Pliable Objects zur Unterstützung des Aufbaus von Informationssystemen im medizinischen Anwendungsgebiet der Anästhesie

Markus Preißner

Eichendorffstr. 46
73230 Kirchheim/Teck
Markus.Preissner@gmx.de

Abstract: Im medizinischen Anwendungsgebiet der Anästhesie ist ein hohes Informationsaufkommen zu dokumentieren. Der Einsatz moderner Informationssysteme, die Unterstützung bei der Erfassung und dem „Bereitstellen" von anästhesiespezifischen Informationen leisten, wird gefordert. Die Art und Anzahl der zu dokumentierenden anästhesiologischen Informationen, orientieren sich an der fachlichen Ausrichtung der Klinik, z.B. Unfall-, Herz- oder Kinderklinik, der die Anästhesieabteilung angehört. Ein standardisiertes Informationssystem kann die individuelle, eventuell mobile Anästhesiedatenverarbeitung nicht dokumentieren, da ein nach objektorientierten Gesichtspunkten gestaltetes Anästhesiemodell nicht dynamisch formbar ist. In dem vorliegenden Beitrag werden Pliable[1] Objects entwickelt, die über die Dynamik verfügen, an anwenderspezifische individuelle Anforderungen anpassbar zu sein.

1 Anästhesie

Im weiteren Sinne bedeutet Anästhesie die Schmerz- und Bewusstseinsausschaltung eines Patienten (vgl. [USZ03]). Das medizinische Fachgebiet der Anästhesie umfasst auch die Behandlung von akuten oder chronischen Schmerzzuständen unterschiedlichster Art. In der Anästhesie werden drei Grundformen grundsätzlich unterschieden: Bei der Lokalanästhesie (örtliche Betäubung) wird nur ein kleiner Bereich des Körpers schmerzfrei gemacht. Mittels der Regionalanästhesie wird ein größerer Körperabschnitt betäubt, und die Allgemeinanästhesie („Vollnarkose") versetzt den Patienten in einen schlafähnlichen entspannten Zustand, in dem der gesamte Körper empfindungs- und schmerzfrei gemacht wird. Lokalanästhesien werden bei kleineren Eingriffen in der Regel vom behandelnden Arzt, z.B. Chirurgen oder Zahnarzt, vorgenommen; in den Zuständigkeitsbereich von Anästhesisten gehören die Regional- und Allgemeinanästhesien.

Diese Anästhesien erfolgen in den meisten Fällen durch spezialisierte Anästhesiefachärzte und -pflegefachkräfte. Anästhesien schalten nicht nur Schmerzen aus, sondern können auch andere Auswirkungen auf verschiedene wichtige Organsysteme haben, so dass anästhesierte Patienten engmaschig überwacht werden müssen. Dadurch können bei operativen Eingriffen Auswirkungen von Anästhesien auf lebenswichtige Organfunktionen vermieden bzw. sofort behandelt werden. Wie alle ärztlichen Maßnahmen ist die Anästhesie mit Risiken verbunden. Deren Wahrscheinlichkeiten konnten durch Qualitätssteigerung der Krankenversorgung reduziert werden. Entscheidenden Anteil hatte auch die Einführung technischer und organisatorischer Sicherheitsstandards. Zur Einhaltung dieser Sicherheitsstandards wurde eine Dokumentationspflicht für die Anästhesie eingeführt. Die Dokumentation dient der Erfüllung der Nachweispflicht, der Erfassung von Leistungen, der Sicherung der Qualität, der Informationsweitergabe und der Transparenz der Verantwortung, die gegenüber dem Patienten besteht, der seine Interessen und Grundbedürfnisse in narkotisiertem Zustand nicht selbst befriedigen kann.

[1] Pliable: engl. biegsam, nachgiebig, formbar

Die Deutsche Gesellschaft für Anästhesiologie und Intensivmedizin (DGAI) hat für die Qualitätssicherung in der Anästhesie Richtlinien verabschiedet (vgl. [OW99]). Diese geben Empfehlungen ab, welche Daten im Rahmen einer Anästhesie zu erfassen bzw. zu protokollieren sind. Die Protokollierung einer Anästhesie unterteilt sich in drei Phasen: In der präoperativen Phase werden der Zustand des Patienten und präoperative Maßnahmen dokumentiert. Die interoperative Phase beginnt mit der Einleitung der Anästhesie. In dieser Phase überwacht und dokumentiert der Anästhesist permanent den Zustand des Patienten. Die postoperative Phase beginnt mit der Verlegung des Patienten in den Aufwachraum. Hier wird der Patient aus seinem narkotischen Schlaf geholt und Komplikationen medikamentiv kompensiert. Abschließend für die Anästhesie folgt gegebenenfalls noch die postanästhesiologische Visite auf der Station, auf welcher der Patient untergebracht ist.

In allen Phasen der Anästhesie ist die Protokollierung verpflichtend vorgeschrieben, um dadurch die Transparenz und Nachvollziehbarkeit ärztlicher Entscheidungen zu gewährleisten. Derzeitige Realisierung sind klinikspezifische Papierprotokolle, die vom Anästhesisten auszufüllen sind.

2 Probleme bei der Entwicklung medizinischer Informationssysteme

Von Seiten der Anästhesisten besteht der Wunsch, anstelle der Anästhesieprotokolle ein die Protokollierung unterstützendes Anästhesie-Informationssystem einzusetzen. Die Kernanforderung an das Informationssystem ist die Sicherstellung der Plausibilität der Protokolle bei deren Eingabe. Im Rahmen des Drittmittelprojektes MC-MEDIS an der TU Clausthal bzw. Universität Hamburg wurde ein Anästhesie-Dokumentationssystem entwickelt. Der Prototyp wurde entsprechend eines Datenbank-Entwurfprozesses (vgl.[V00]) entwickelt.

Während der Evaluierung des Prototypen traten neue Anforderungen auf, die ins System integriert werden mussten. Beispielsweise wünschte ein Anwender die Änderung der anfänglichen Werte {männlich, weiblich} für das Geschlecht eines Patienten auf die Wertemenge {männlich, weiblich, intersexuell}. Die Umstellung der bislang verwendeten boolschen Variablen auf den Aufzählungstyp erforderte die Anpassung sowohl auf Datenbank- als auch auf Anwendungsebene sowie auf Ebene der Bedienoberfläche. Dieses Beispiel ist exemplarisch für die Änderungswünsche, die im Laufe des Projektes auftraten. Generell waren neue oder bestehende Daten unter anderen Bedingungen zu erfassen; häufig waren ferner mit den Daten verknüpfte Plausibilitätskontrollen anzupassen. Die sich dadurch ergebene System-Evolution wurde sehr implementierungslastig.

Erforderlich wurde zusätzlich die Entwicklung und Integration einer Komponente für eine Belegleser-Lösung. Heute ist es in vielen Kliniken gängige Praxis, ausgefüllte Protokolle nach Abschluss des gesamten zu dokumentierenden Anästhesievorganges per Belegleser oder Scanner in ein Informationssystem einzulesen. Erst zu diesem Zeitpunkt können durch Plausibilitätsprüfungen Fehler erkannt werden. Nicht plausible Protokolle müssen allerdings auch in der Datenbank des Informationssystems gespeichert werden, damit sie für eine Nachbearbeitung am Bildschirm zur Verfügung stehen. Um dies zu ermöglichen, wurden Integritätsbedingungen gelockert und die dadurch verminderte Überprüfung der Plausibilität von der Datenbankseite in die Anwendungslogik verlagert.

Eine Erweiterung der Anforderungen an ein Anästhesie-Informationssystem ergab sich durch den Umstand, dass jede Klinik, die über eine eigene Anästhesie-Abteilung verfügt, sein eigenes Anästhesieprotokoll definieren und somit hochgradig individuell auf die klinikspezifischen Bedürfnisse zuschneiden kann. Die Empfehlung der DGAI ist nur die Richtlinie, an der man sich orientiert. Bundesweit verwenden schätzungsweise 50% aller Kliniken ein individuell gestaltetes Anästhesieprotokoll. Das allgemeingültige Verständnis der Anästhesie präsentiert sich damit als ein generalisiertes Anwendungsszenario, welches beliebige Freiheitsgrade für die klinikspezifische Gestaltung besitzt. Jede individuelle Ausprägung entspricht einem Anwendungsfall, für den ein eigenständiges Informationssystem entwickelt werden kann. Derzeit wird ein klinikspezifisches Anästhesie-Informationssystem unter erheblichem personellem Aufwand und unter einem hohen Anteil manueller Programmierung durch Generierung einer klinikspezifischen Version aus einer Art

Default-System gewonnen. Die Entwicklung eines einzigen Informationssystems ist gefordert, welches durch überschaubaren Aufwand an die jeweiligen klinikspezifischen Bedürfnisse angepasst werden kann. Ein objektorientiertes Modell wurde für solch ein System in Zusammenarbeit mit einem medizinischen Partner spezifiziert. Leider lagen die zu erwartenden Kosten der Realisierung weit über dem zur Verfügung stehenden Budget und führten zur Einstellung des Projektes.

3 Pliable Objects

Pliable Objects repräsentieren die Idee, individuell zugeschnittene Informationssysteme für den medizinischen Fachbereich der Anästhesie kostengünstig zu entwickeln. Grundsätzlich ist festzustellen, dass die überwiegenden Informationen, die im Anästhesiebereich zu protokollieren sind, in anwenderseitig definierten Wertemengen gegeben sind. Auf dieser Basis wird im folgenden die Kernidee der Pliable Objects präzisiert. Pliable Objects leiten sich im wesentlichen vom objektorientierten Paradigma ab, wobei eine Spezialisierung auf Methoden durch Unterteilung in Aktionen und Regeln vorgenommen wird.

3.1 Definition Pliable Object

Ein *Pliable Object* π ist eine Beschreibungsdomäne Γ, die um eine Menge von *Attributs* A, einer Menge von *Actions* X und einer Menge von *Rules* P zu einem Tupel $\pi = (\Gamma, A, X, P)$ erweitert wird.

- Die Beschreibungsdomäne setzt sich zusammen aus einem Fachausdruck zur Benennung und einer (umgangssprachlichen) Erklärung der Bedeutung und des Zweckes, die dem Pliable Object durch die Experten zugewiesen werden.

- Ein Attribut α dient der Angabe einer Zustandsinformation und sei gegeben als das Tripel $\alpha = (\nu_\alpha , \delta_\tau, \omega_\alpha)$, wobei ν_α eine Benennung, δ_τ eine Domäne und ω_α einen Wert bezeichnet. Der Wert sollte(!) ein Element der Domäne sein. Eine Domäne ist gegeben als $(\tau, \Omega_\tau) = \delta_\tau \in \Delta \subseteq \mathbf{T} \times \mathbf{W}$, mit der Typbezeichnungsmenge $\mathbf{T} \in P(\mathbf{U})$ und der *Wertemengenmenge* $\mathbf{W} \subset P(\mathbf{U})$[2]. Das Universum **U** sei die Menge aller Werte.

- Eine Action χ ist die Definition eines Zustandswechsels und sei gegeben als das Tripel $\chi = (\nu_\chi, A_\chi, \Sigma_\chi)$ mit $A_\chi \neq \emptyset$, wobei ν_χ eine Benennung, A_χ die Menge der Attribute, die eine Zustandsänderung erfahren sollen, und Σ_χ eine Liste von Anweisungen, die die Zustandsänderungen beschreiben, bezeichnet.

- Eine Rule ρ dient der Steuerung von Zustandswechseln und sei gegeben als das Tripel $\rho = (\nu_\rho, B_\rho, K_\rho)$ mit $B_\rho \neq \emptyset$ und $K_\rho \neq \emptyset$, wobei ν_ρ eine Benennung, B_ρ eine Menge von Restriktionen und K_ρ eine Menge von Aktionen bezeichnet. Die Aktionen wären als Konklusion auszuführen, wenn die Restriktionen erfüllt sind.

Beispielsweise wäre das Attribut Herzinfarkt[3] mit der entsprechenden Domäne wie folgt anzugeben:

$$\alpha_{\text{Herzinfarkt}} = (\text{Herzinfarkt}, \delta_{\text{Herzinfarktsarten}}, \text{n.bekannt})$$
$$\text{mit } \delta_{\text{Herzinfarktsarten}} = (\text{Herzinfarktsarten}, \{ < 3 \text{ Monate}, < 6 \text{ Monate}, > 6 \text{ Monate}\})$$

[2] Es sei darauf hingewiesen, dass es sich um eine echte Teilmenge handelt. Dies dient dazu, dass generell neue Domänen angelegt werden können. Bei Gleichheit zur Potenzmenge wäre dies nicht möglich.

[3] Anmerkung: Die Reinfarktrate ist innerhalb des ersten halben Jahres nach dem Infarkt besonders hoch. Liegt ein Infarkt maximal ein halbes Jahr zurück, wird empfohlen, einen neuen Operationstermin festzulegen, wenn die Operation aufschiebbar ist. Andernfalls sollten zusätzliche subtile Überwachungsmethoden zum Einsatz kommen.

3.2 Datenbankschema für Pliable Object

Ein Datenbankschema für Pliable Objects kann gemäß Abbildung 1 visualisiert werden:

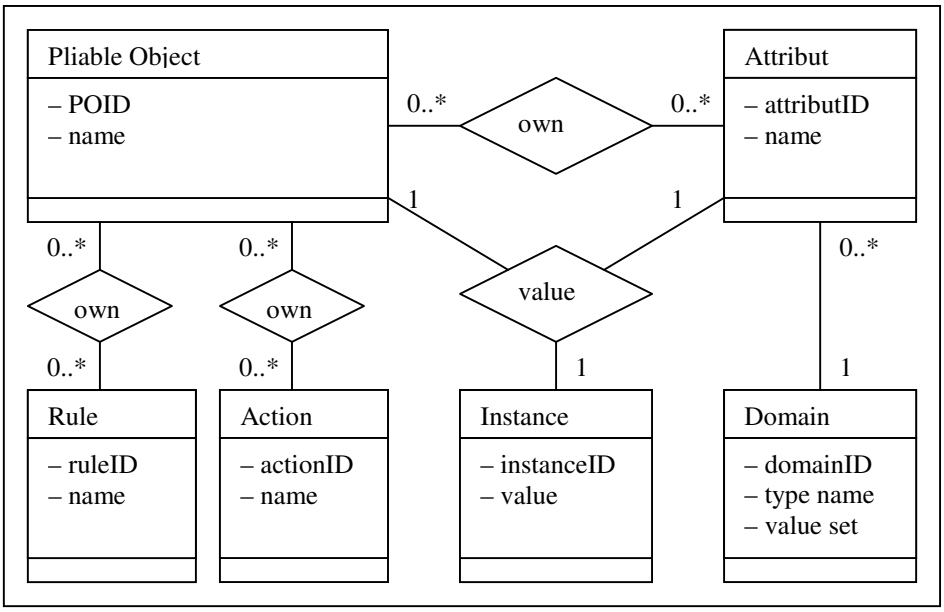


Abbildung 1: Pliable Object

Eine zentrale Bedeutung nimmt der Wert ein, der der einzugebenden Information entspricht. Der Wert *value* wird in einer Instanz gespeichert, der das zugehörige Attribut assoziiert ist. Mittels der ternären Assoziation wird die is-a-Beziehung zum Pliable Object angeben, um über dessen Regeln und Aktionen darin gespeicherte Plausibilitätskontrollen zu triggern.

4 Ausblick

Anästhesisten entwerfen das Layout ihres Anästhesieprotokolls ähnlich wie eine Vorlage in einer Textverarbeitung. Vorlagen können darin schnell verändert und beliebig oft verwendet werden. Die Datenbank eines Informationssystems, insbesondere deren Datenbankschema, ist nicht so schnell anzupassen. Die meisten Änderungswünsche der Anästhesisten beinhalten jedoch Schemaanpassungen. Zur Ermöglichung der Änderungen des Informationssystems wurden Pliable Objects vorgeschlagen. Das vorgestellte Datenbankschema für Pliable Objects erlaubt neben der Datenspeicherung nicht nur die Speicherung von zugehörigen Schemainformationen, sondern ermöglicht deren Nutzung zur Auswertung wie z.B. die Triggerung von Plausibilitätskontrollen. Schemainformationen sind nach Vossen (vgl. [V00 - Seite 493]) allgemein als „Daten über Daten“ und als zeitinvariante Informationen klassifiziert. Pliable Objects und das vorgeschlagene Datenbankschema müssen sich der Diskussion stellen, ob sie eine Grundlage für ein Data Dictionary mit zeitvarianten Informationen darstellen.

Literaturverzeichnis

[BS02] Broy, M., Siedersleben, J.: Objektorientierte Programmierung und Softwareentwicklung – Eine kritische Einschätzung, Informatik Spektrum, 25.2.2002

[B97] Bruder, M., Rehfeld, W., Seeger, T., Strauch, D. (Hrsg.): Grundlagen der praktischen Information und Dokumentation, 4. völlig neu gefasste Ausgabe, Band 1 und 2, K.G.Saur Verlag 1997

[C96] Casanave, Cory (ed.): OMG Common Facilities RFP-4 Common Business Objects and Business Object Facility, OMG TC Document CF/96-01-04, www.omg.org

[CYA94] Coad, P., Yourdon, E.: OOA – Objektorientierte Analyse, Prentice Hall Verlag, 1994

[CYD94] Coad, P., Yourdon, E.: OOD – Objektorientiertes Design, Prentice Hall Verlag, 1994

[CC01] Cuhls, M., Cuhls, H.: Patienteninformation – Ablauf einer Narkose in Bildern, 15.06.2001, http://www.meb.uni-bonn.de/institute/kliansint/patient/inhalt.htm

[GHJV95] Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Entwurfsmuster, Addison-Wesley, 1995

[H92] Heuer, A.: Objektorientierte Datenbanken, Konzepte, Modelle, Systeme, Addison-Wesley, 1992

[KS01] Kretz, F.J., Schäfer, J.: Anästhesie, Intensivmedizin, Notfallmedizin, Schmerztherapie, Springer-Verlag, 2001

[OW99] Opderbecke, H.W., Weissauer, W.(Hrsg.): Entschließungen, Empfehlungen, Vereinbarungen, Ein Beitrag zur Qualitätssicherung in der Anästhesie, Kerndatensatz Anästhesie – Version 2.0/1999, http://www.dgai.de/06_1_00tabelle.htm#zu_viii

[P01] Pfeifer, T.: Qualitätsmanagement, Hanser Verlag, 2001

[PGPB96] Puppe, F., Gappa, U., Poeck, K., Bamberger, S.: Wissensbasierte Diagnose- und Informationssysteme, Springer Verlag 1996

[RT04] Roewer, T., Thiel, H.: Taschenatlas der Anästhesie, 2. aktualisierte Auflage, Thieme Verlag, 2004

[S94] Sims, O.: Business Objects, Delivering cooperative objects for client-server, McGraw – Hill, 1994

[SPCH97] Sutherland, J., Patel, D., Casanave, C., Hallowell G., Miller, J.(eds): Business Object Design and Implementation, OOPSLA'95 Workshop Proceedings, Springer Verlag, 1997

[USZ03] UniversitätsSpital Zürich, Institut für Anästhesiologie, Anästhesie Allgemein http://www.anaesthesie.unispital.ch/german/PatientenUndBesucher/AnaesthesieAllgemein/default.htm

[V00] Vossen, G.: Datenmodelle, Datenbanksprachen und Datenbankmanagementsysteme, Oldenbourg, 2000

[Z99] Zwally, B.: Pflegestandards für die Anästhesieabteilung im OP 04, 1998/99, http://pflege.klinikum-grosshadern.de/campus/anaesthe/standard/zwally.html

Pathfinder/MonetDB: A High-Performance Relational Runtime for XQuery

Jan Rittinger*
University of Konstanz, Department of Computer & Information Science,
P.O.Box D188, 78457 Konstanz, Germany
Jan.Rittinger@uni-konstanz.de

Pathfinder/MonetDB is a collaborative effort of the University of Konstanz, the University of Twente, and the Centrum voor Wiskunde en Informatica (CWI) in Amsterdam to develop an XQuery compiler that targets an RDBMS back-end. The author of this abstract is student at the University of Konstanz and spent six months as an intern at the CWI, designing and implementing a translation of XQuery Core to (a variant of) relational algebra. His work continues in the research group at the University of Konstanz.

Pathfinder/MonetDB

Pathfinder/MonetDB can be divided into two parts: Pathfinder and MonetDB. Pathfinder is an XQuery compiler (see Figure 1) that translates XQuery expressions into a variant of relational algebra which is executable by the back-end database MonetDB. MonetDB is an extensible main-memory database system kernel which adapts database architecture concepts to the characteristics of modern hardware in order to improve the CPU and memory cache utilization [2].

The MonetDB kernel is equipped with a low-level interface language MIL (MonetDB Interpreter Language), which forms the target language for a number of different front-end query languages (*e.g.*, SQL, OQL, or XQuery). The table manipulation operations in MIL form a closed algebra on the binary table model. MIL can be extended with new primitives, data types, and associated search accelerator structures and contains a computationally complete procedural language. Pathfinder uses these features to extend MonetDB with an XQuery specific runtime module implementing a small number of additional operations (*e.g.*, XML serialization or staircase join [8]).

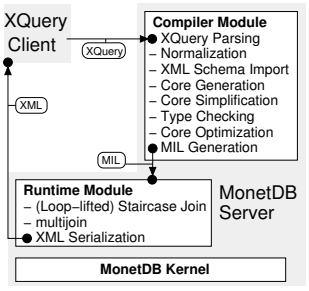


Figure 1: System architecture.

Compiling XQuery to Relational Algebra

The W3C proposes XQuery [1] as the standard query language for XML data. XQuery contains constructs to explore the *tree-structured* XML data model (plus constructors to create new XML fragments) as well as iteration primitives—notably the FLWOR block—to process *sequences* of data items. Earlier work in the context of the Pathfinder project provides the means to close the apparent gap between the set-oriented relational data model and the two principle data types which form the backbone of the XQuery data model, namely *ordered, unranked trees of nodes* and *ordered, finite sequences of items*.

We translate trees of nodes into a relational encoding—the XPath accelerator—applying the ideas presented in [5]. Item sequences, on the other hand, may be transformed using techniques originally developed for a mapping from XQuery to SQL [6, 7]. The inference rules described in these papers form a mapping from XQuery Core to an (almost) standard relational algebra. MIL perfectly supports these operators and also allows for a tight integration of extensions (*e.g.*, staircase join [8]) which are geared to support the embedded XPath sub-language. In this work, we significantly extended the compiler described in [6] to support a wide variety of XQuery constructs.

Optimizing Relational XQuery Evaluation — XQuery Join Recognition

The compilation procedure originally described in [6] opens several opportunities to improve, among them the recognition and translation of joins. Because the XQuery language lacks an explicit join oper-

*Advisors: Torsten Grust, University of Konstanz, Department of Computer & Information Science, P.O.Box D188, 78457 Konstanz, Germany (Torsten.Grust@uni-konstanz.de); Peter Boncz, CWI Amsterdam, Department INS1: Database Architectures and Information Access, P.O.Box 94079, 1090 GB Amsterdam, The Netherlands (p.boncz@cwi.nl)

ator, the only implicit way to perform a join is to use nested loops with an embedded filter expression. Unfortunately, the compiler described in [6] always emits algebraic Cartesian products for such nested `for`–loops and the implicit XQuery join remains undetected. Query Q_1 below shows a typical XQuery scenario which implicitly joins the two sequences `(30,20,10)` and `(1,2,3)`. The original translation strategy forms the Cartesian product of the two sequence representations (yielding an intermediate result of 9 rows), despite the fact that the final query result is linear in the size of the input sequences.

```
for $u in (30, 20, 10),
    $v in (1, 2, 3)
where $u eq ($v * 10)
return "hit"
```
(Q_1)

This inefficiency may be avoided by an XQuery compilation procedure that recognizes XQuery joins. The compiler matches a given general XQuery Core pattern and, in case of a successful match, translates the XQuery expression into an relational equivalent which makes use of an algebraic join. In the current compiler, join recognition solely works at the level of XQuery Core and matches the following query pattern:

$$\texttt{for \$}v\texttt{ in }e_{in}\texttt{ return if (}p(e_1,e_2)\texttt{) then }e_{\text{return}}\texttt{ else () .} \qquad (P_1)$$

There are some preconditions which have to be met to guarantee that the above syntactic pattern actually describes a join (in a nutshell, the conditions below guarantee that the join inputs are indeed *independent* of each other). To be more concrete, the pattern P_1 above qualifies for an XQuery join, only if

(i) variable `$v` appears free in e_2 only[1],
(ii) variables occurring free in e_2 and e_{in} are bound in any enclosing scope, except for the scope that *directly* encloses P_1, and
(iii) predicate p is supported by the theta-join implementation of the relational back-end (*i.e.*, typically, p will be `eq`, `=`, `lt`, `<`, ...).

```
for $u in (30, 20, 10)
return $u
        and
for $v in (1, 2, 3)
return $v * 10
```
(Q_2)

Since all restrictions hold for Query Q_1, this query can be split into two independent sub-queries (Q_2) which compute the join inputs: The final overall translation does not involve Cartesian products at all.

The gory details and a formal description of the join compilation can be captured by means of an inference rule similar to the ones gives in [6]. This work is currently under submission.

Performance Results

To show that our relational approach may indeed yield a high-performance XQuery processor, we conducted experiments[2] in which we focused on XMark [9], as the most frequently used benchmark for evaluating XQuery efficiency and scalability. We performed measurements at scaling factors 0.1, 1 and 10 (which yield documents of respectively 10 MB, 100 MB, and 1 GB), using Pathfinder/MonetDB as well as the latest versions of Galax (0.4.0) [4] and X-Hive (6.0) [10].

XQuery Join Recognition. In our first runs, Pathfinder/MonetDB was not able to evaluate the XMark join queries Q8–Q12 on the 100 MB and 1 GB sizes, due to excessive running time and resource consumption.

The reason was the generation of huge intermediate Cartesian products. Figure 2 contrasts the results for the 10 MB document with the performance we obtained with join recognition enabled in our MIL generation. It is obvious that the execution of XQuery statements with join predicates simply *requires* join recognition when the query is run on significant XML document sizes.

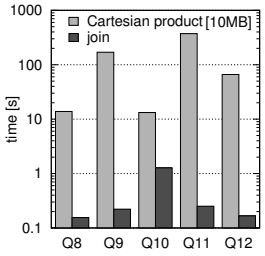


Figure 2: Benefits of XQuery join recognition.

Scalability. In Figure 3 all numbers are normalized to the elapsed time on the 100 MB document. The figure clearly shows that Pathfinder/MonetDB scales linearly with document size. The only outliers are query Q11 and Q12. The culprit in this queries is a theta-join predicate ($>$) which generates an intermediate result almost the size of the Cartesian product of its inputs. Note that this concerns the query

[1] The roles of e_1, e_2 may be arbitrarily swapped.

[2] The platform was a 1.6 GHz AMD Opteron 242 (1 MB L2 cache) processor with 8 GB RAM and a RAID-5 disk subsystem (3ware 7810, configured with eight 250 GB IDE disks of 7200 RPM). The operating system was Linux 2.6.9, using a 64-bit address space.

result, whose computation cannot be avoided (though the end result becomes small, due to subsequent aggregation) and thus *any* XQuery processor will face this problem.

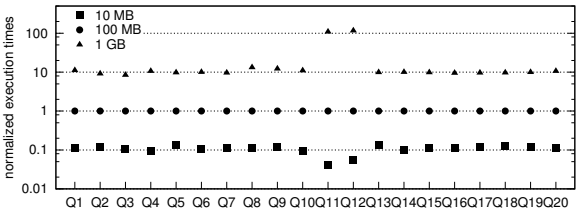


Figure 3: Scaling with respect to document size.

System Comparison. For our own system, we generated MIL query plans[3], which range from 318 lines (XMark Query Q6) to 11,509 lines (Q10) with an average of about 2,000 lines. The next step, the document loading[4], is followed by the query evaluation and the serialization[5]. Our two reference systems are Galax, which is the most popular "native" XQuery engine available in open-source, and X-Hive, which is one of the faster native XML database systems (shown in [3]). The performance results of X-Hive and Pathfinder/MonetDB contain only query evaluation times, while Galax still[6] includes serialization times.

The table on the right side shows our full experimental results (elapsed time in seconds). Galax failed to process the queries once the XMark documents were of size 100 MB or larger. Compared to our system Galax is marginally faster on queries Q2, Q5, Q13 (obvious, as we use it as base reference), Q16, and Q19. X-Hive also finishes the execution of non-join queries in reasonable time. For the join queries (Q8–Q12) both Galax and X-Hive are dominated by the Cartesian products. X-Hive avoids quadratic complexity in Q8 due to the value indices we created. However if the queries join *intermediate* query results, indices cannot be used and performance degrades strongly. With the help of join recognition, Pathfinder/MonetDB clearly outperforms the other two systems on these queries.

	10 MB			100 MB		1 GB	
Q	Galax	X-Hive	PF/M	X-Hive	PF/M	X-Hive	PF/M
1	0.13	0.37	0.04	1.29	0.37	9.9	4.1
2	0.08	0.45	0.09	1.75	0.72	33.0	6.5
3	0.16	0.65	0.29	5.66	2.63	25.0	22.2
4	0.36	0.10	0.10	0.99	1.00	18.1	10.6
5	0.02	0.13	0.06	1.17	0.43	20.7	4.2
6	1.27	1.07	0.06	10.17	0.54	178.0	5.4
7	2.87	1.57	0.12	24.84	1.09	278.4	10.4
8	127.28	0.85	0.16	3.51	1.38	49.1	18.2
9	142.74	32.25	0.22	2280.66	1.81	DNF	22.2
10	18.16	5.28	1.27	442.37	13.68	DNF	150.6
11	218.23	98.91	0.25	9927.29	6.29	DNF	683.5
12	63.66	23.39	0.17	5100.19	2.98	DNF	347.7
13	—	0.09	0.07	1.03	0.53	12.9	5.2
14	2.13	0.72	0.17	11.17	1.69	110.2	16.9
15	0.08	0.03	0.09	0.49	0.80	10.6	7.8
16	0.06	0.03	0.10	0.51	0.87	10.9	8.2
17	0.14	0.09	0.10	0.85	0.80	11.8	7.6
18	0.03	0.08	0.05	0.64	0.43	14.8	4.2
19	1.32	0.67	0.21	12.15	1.70	254.5	16.9
20	0.52	0.11	0.21	1.40	1.79	24.6	19.0

Conclusion

This present work builds on both, an XPath-aware relational encoding of XML trees and a relational XQuery compiler, to turn a relational database back-end into an XQuery processor. The outcome is a prototype implementation backed by the extensible MonetDB RDBMS which exhibits the efficiency and scalability provided by a relational database. XML input documents of 1 GB size and beyond can be queried in interactive time.

References

[1] S. Boag, D. Chamberlin, M. Fernández, D. Florescu, J. Robie, and J. Siméon. XQuery 1.0: An XML Query Language. World Wide Web Consortium, Oct. 2004. http://www.w3.org/TR/xquery/.

[2] P. Boncz and M. Kersten. MIL Primitives For Querying a Fragmented World. *The VLDB Journal*, 8(2):101–119, Mar. 1999.

[3] D. DeHaan, D. Toman, M. Consens, and M. Öszu. A Comprehensive XQuery to SQL Translation Using Dynamic Interval Encoding. In *Proc. SIGMOD Conf.*, pages 623–634, San Diego, CA, USA, June 2003.

[4] M. Fernández, J. Siméon, B. Choi, A. Marian, and G. Sur. Implementing XQuery 1.0: The Galax Experience. In *Proc. VLDB Conf.*, pages 1077–1080, Berlin, Germany, Sept. 2003.

[5] T. Grust. Accelerating XPath Location Steps. In *Proc. SIGMOD Conf.*, pages 109–120, Madison, WI, USA, June 2002.

[6] T. Grust, S. Sakr, and J. Teubner. XQuery on SQL Hosts. In *Proc. VLDB Conf.*, pages 252–263, Toronto, Canada, Sept. 2004.

[7] T. Grust and J. Teubner. Relational Algebra: Mother Tongue—XQuery: Fluent. In *Twente Data Management Workshop on XML Databases and Information Retrieval (TDM)*, pages 7–14, Enschede, The Netherlands, June 2004.

[8] T. Grust, M. van Keulen, and J. Teubner. Staircase Join: Teach a Relational DBMS to Watch its (Axis) Steps. In *Proc. VLDB Conf.*, pages 524–535, Berlin, Germany, Sept. 2003.

[9] A. Schmidt, F. Waas, M. Kersten, M. Carey, I. Manolescu, and R. Busse. XMark: A Benchmark for XML Data Management. In *Proc. VLDB Conf.*, pages 974–985, Hong Kong, China, Aug. 2002.

[10] X-Hive/DB. http://www.x-hive.com/.

[3] The times for running the XQuery compiler varies between 60 and 100 ms for all XMark queries. The compiler timings are excluded from our performance results.

[4] The XML loader of Pathfinder/MonetDB imports the 10 MB XMark document in 1.15 seconds.

[5] Serialization times are excluded from the performance numbers. For the 10 MB instance they are all below 50 ms (except Q10: 690 ms).

[6] Galax is a file-oriented system that parses the XML file on each query. To (over-)compensate for XML parsing time, we subtracted 8.25 seconds (which is the time of the fastest query Q13) from all Galax performance figures.

Content Sharing – Probleme und Lösungen bei der Föderation von Lernmoduldatenbanken

Matthias Rust, Guntram Flach, Ralph von Petersdorff-Campen

eGovernment & Multimedia Information Management
Zentrum für Graphische Datenverarbeitung e. V., Rostock
Joachim-Jungius-Str. 11
18059 Rostock
{matthias.rust, guntram.flach, ralph.v.petersdorff-campen}@rostock.zgdv.de

Abstract. Die Nachnutzung von einmal erstellten E-Learning-Modulen ist aus wirtschaftlichen Gesichtspunkten eine unumgängliche Notwendigkeit, die durch aktuelle Systeme und Ansätze noch nicht ausreichend unterstützt wird. Dieser Beitrag stellt Methoden und Algorithmen vor, die im Rahmen des Projektes WIESEL entwickelt wurden und eine dynamische Zusammenstellung von E-Learning-Kursen aus einzelnen Lernmodulen erlauben. Diese Lernmodule können dabei aus verteilten Lernmoduldatenbanken (Repositories) stammen. Der Aufbau der Lernmodulkette wird anhand eines anzugebenden Lernziels und entsprechender Metadaten für die einzelnen Module, die auf einer Erweiterung des LOM-Standards beruhen, mit ontologiegestützten Methoden durch Komponenten von WIESELretrieval durchgeführt.

1. Einleitung

Durch die Entwicklung und Verbreitung von Computern und hochkapazitären Internetanbindungen hat sich E-Learning in den letzen Jahren in Wirtschaft und Forschung als Kernthema etabliert. Viele Firmen integrieren E-Learning-Aspekte in die Weiterbildung. Universitäten bereiten ihren Lernstoff multimedial für das Internet auf. Für die Sicherstellung der Qualität und die Ermöglichung interoperabler Systeme wurde auf nationaler und internationaler Ebene bereits eine Reihe von Standards und Referenzsystemen für E-Learning entwickelt, die sich in ständiger Fortentwicklung befinden.

SCORM ist ein Standard, der international etabliert ist und bereits von vielen Systemen umgesetzt wird [ADL01]. SCORM definiert E-Learning-Kurse als Pakete, die aus einzelnen SCOs (sharable content objects) zusammengesetzt sind. Integraler Bestandteil von SCORM ist der Metadatenstandard LOM (learning object metadata), der für Kurse, für einzelne SCOs, aber auch für einzelne Dateien innerhalb von SCOs die Angabe verschiedenartiger Metadaten erlaubt. Eine Vision von SCORM besteht in der Nachnutzung und Wiederverwendung von SCOs zu neuen Kursen, die allerdings von heutigen Systemen noch nicht hinreichend umgesetzt ist [Aeh05].

Im Rahmen des Projektes WIESEL[1] hat das ZGDV Rostock in Zusammenarbeit mit der ANOVA Multimedia Studios GmbH ein Konzept entwickelt, das auf der Basis von SCORM die automatische und personalisierte Zusammenstellung existierender Lernmodule (SCOs) zu neuen Kursen ermöglicht (siehe Abbildung 1). Allein die Eingabe eines Lernziels soll genügen, eine Lernmodulkette zu erzeugen, die dann innerhalb eines Lernmanagementsystems als SCORM-kompatibler Kurs verwendet werden kann.

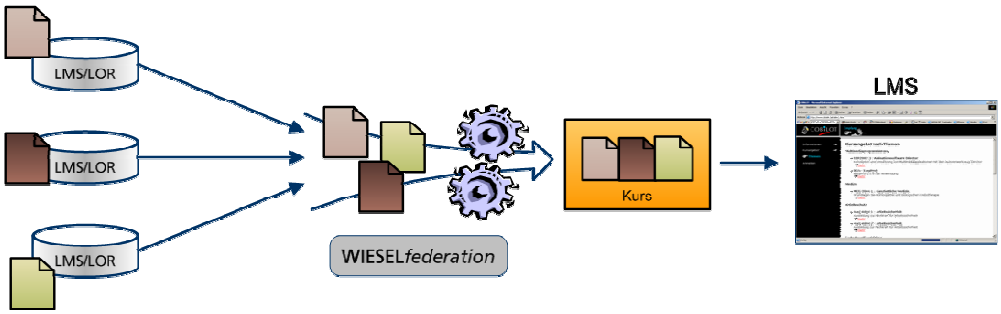


Abbildung 1: Föderation von Kursen mit WIESELfederation

[1] WIESEL wurde vom BMWA im Rahmen von PRO INNO im Zeitraum 05/2003-10/2004 gefördert.

Durch den vorgestellten Ansatz wird es möglich, personalisierte Kurse, d. h. an den Lerner angepasste Kurse automatisch erzeugen zu lassen. Die folgenden Kapitel widmen sich dem entwickelten Lösungsansatz und Komponenten sowie dem umgesetzten Experimentalsystem. Zum Abschluss werden eine kurze Zusammenfassung und ein Ausblick auf weitere Arbeiten und offene Probleme gegeben.

2. Lösungsansatz

Das angestrebte Ziel besteht in dem Ermöglichen eines automatischen wissensgestützten Aufbaus von Lernmodulketten aus verteilten Lernobjekt-Repositories (LORs) [Se02]. Als softwarearchitektonische Grundlage des entwickelten Lösungsvorschlages wurde ein serviceorientierter Ansatz gewählt, der auf einer losen Kopplung der beteiligten Komponenten über WebServices basiert. Einzelne LORs fungieren als Service-Provider und erlauben die Recherche und den Zugriff auf die verwalteten E-Learning-Module. Da keine standardisierte und etablierte Schnittstelle für E-Learning LORs existiert [IMS03] [ND02], erwächst daraus als eine notwendige Teilaufgabe die Anbindung heterogener LORs, die durch eine Anpassung des am ZGDV entwickelten Gateway-Ansatzes verwirklicht wurde [AS03].

Allein die Eingabe eines Lernzieles soll genügen, einen an den Nutzer angepassten Kurs vorzuschlagen, der aus unterschiedlichen und verteilten Modulen föderiert wird. In unserem Lösungsvorschlag kann durch die explizite Angabe von notwendigem Vorwissen für einzelne Lernmodule und der jeweils erreichten Lernziele mit intelligenten Algorithmen eine Kette von passenden Lernmodulen zusammengestellt werden.

Die folgenden Abschnitte dieses Beitrags beschreiben zuerst die vorgeschlagene Architektur und die relevanten Komponenten. Für die Umsetzung dieses Konzepts müssen Erweiterungen an den Metadatenspezifikationen für E-Learningmodule zur Unterstützung der automatischen Föderation vorgenommen werden. In entsprechender Abhängigkeit wird der Algorithmus für den Aufbau der Lernmodulkette vorgestellt. Zur Unterstützung einer flexiblen Auswertung dieser Metadaten werden fachspezifische Wissensnetze (Domainontologien) eingesetzt, um auch bei einer unmöglichen direkten Zuordnung von Metadaten passende Kursmodule zu identifizieren. Durch die durchgängigeVerwendung des SCORM-Standards wird Kompatibilität zu existierenden E-Learning Werkzeugen und Umgebungen gewährleistet.

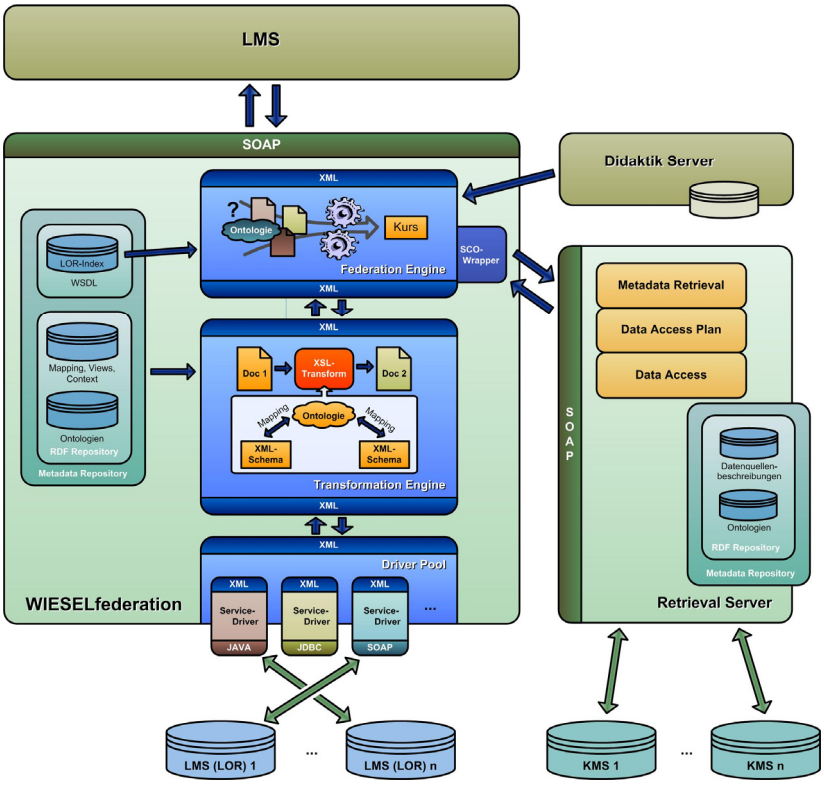


Abbildung 2: Architektur von WIESELfederation

Architektur

In der Abbildung 2 sind die Komponenten von WIESELfederation, welche für die Föderation eingesetzt werden, dargestellt. Die FederationEngine generiert einen Kurs, welcher die Anforderungen des Lernenden erfüllt. Hierzu generiert diese Komponente Suchanfragen, um das benötigte Wissen durch Kursmodule aus verschiedenen LORs bereitzustellen. Die TransformationEngine übernimmt die Aufgabe der Transformation der Suchanfragen an die heterogenen teilnehmenden LORs basierend auf dem am ZGDV entwickelten Gateway-Ansatz [AS03]. Das von dieser Komponente erzeugte Ergebnis ist eine Übersetzung der Antwort des entsprechenden LORs, so dass die Federation Engine diese verarbeiten kann. Weiterhin werden die Begriffe, welche durch die Kursteile erlernt werden, an eine CourseEnrichmentEngine weitergegeben, um den Kurs durch weitere Inhalte aus Wissensquellen zu ergänzen (SCO-Wrapper) [FR04]. Das Ergebnis dieser Komponente ist ein Kurs, welcher in einem übergeordneten Lernmanagementsystem bereitgestellt werden kann.

Anforderungen an Kursmodule/Metadaten

Wie bereits erwähnt, ist es notwendig, Erweiterungen an existierenden Metadatenstrukturen von SCOs vorzunehmen, um eine Föderation nach dem vorgestellten Prinzip der Vor- und Nachwissensbeschreibungen zu ermöglichen. Der breit akzeptierte Standard LOM (Learning Object Metadaten) für die Definition von Metadaten stammt vom IMS und ist Bestandteil der SCORM-Spezifikation.

Die vorgestellten Erweiterungen ergänzen den LOM-Standard um zwei zentrale Konzepte: Die Angaben zum benötigten Vorwissen werden innerhalb des zusätzlichen Knotens „prerequisite“ angegeben. Im Knoten „learningsresults“ werden die Angaben über das geschulte Wissen gesichert. Beide Knoten wurden innerhalb der LOM-Beschreibungsgruppe „classification“ zugefügt. Die erweiterten Strukturen gestatten sowohl den Verweis auf Konzepte existierender Ontologien (Kindelement „id“) als auch die Angabe von freitextlichen Werten. Im Falle des Fehlens der Konzept-ID wird über einen einfachen Textvergleich versucht, passende Ontologiekonzepte zu dem angegebenen Keyword zu identifizieren und diese in den wissensbasierten Föderationsalgorithmus einfließen zu lassen.

Komponente WIESELfederation

In der Regel wird man die Anforderungen, welche ein Nutzer an das LMS stellt, nicht mit einem vorgefertigten Kurs erfüllen können. Dies hängt damit zusammen, dass jeder Lernende ein anderes Vorwissen besitzen kann und damit für einen fertigen Kurs entweder zu viel oder zu wenig Wissen hätte. Dieses Problem soll mit WIESELfederation gelöst werden. Es ist somit notwendig, ausgehend vom aktuellen Wissen des Nutzers, die Ziele, die der Nutzer von dem zu generierenden Kurs erwartet, zu erfüllen. Es ist in den meisten Fällen nicht möglich, nur einen Weg von den Lernvoraussetzungen zu den gewünschten Zielen zu beschreiten, daher bietet sich ein Suchbaumansatz für die Föderation an.

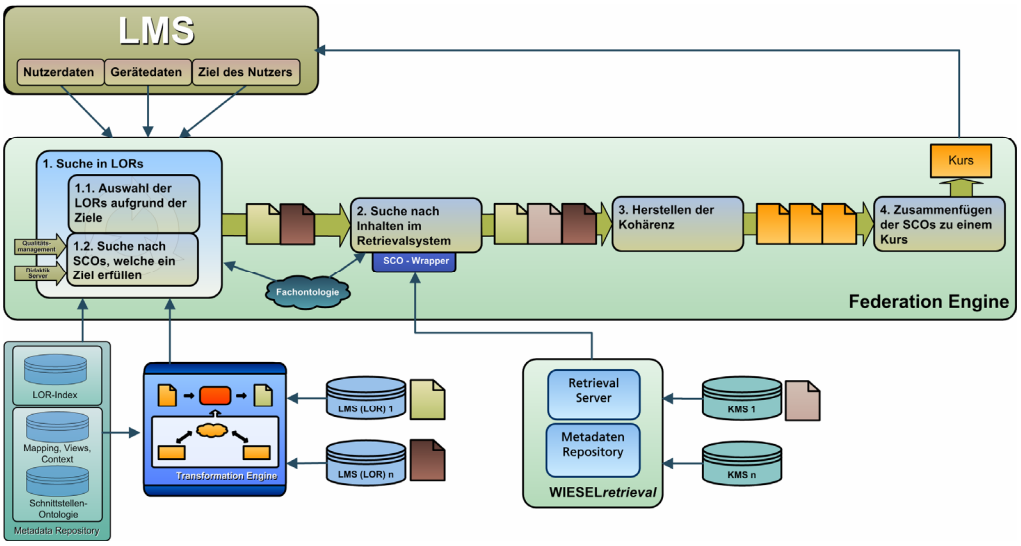


Abbildung 3: Kernalgorithmus von WIESELfederation

In der Abbildung 3 sind die Bearbeitungsschritte während des Föderationsprozesses innerhalb der zentralen Komponente FederationEngine dargestellt. Der erste Schritt beinhaltet die Suche nach Kursmodulen innerhalb der teilnehmenden LORs. Im ersten Teilschritt wird zuerst die LOR-Registry im Metadatenrepository von WIESELfederation nach passenden LORs angefragt. Durch die semantisch angereicherten LOR-Beschreibungen können für registrierte Repositories thematische Schwerpunkte angegeben werden. Z. B. würde man bei der Suche nach biologischen Lernmodulen kein LOR mit ausschließlich juristischen Inhalten in den Algorithmus einbeziehen.

Der zweite Teilschritt widmet sich dem Finden einer Modulstruktur in den relevanten LORs. Aufgrund der Anforderungen an den Prozess bieten sich für die Föderation sowohl die Algorithmen der Heuristisch Informierten Suche als auch die der Optimalen Netzsuche an. Die Unterschiede liegen bei der Güte des föderierten Kurses. Die Algorithmen der Heuristisch Informierten Suche finden auch nicht-optimale Lösungen für das Problem und sind daher in der Regel schneller als die Algorithmen der Optimalen Netzsuche, welche den besten Weg durch den Suchbaum finden. In WIESELfederation wird für die Suche ein Best-First Search-Algorithmus eingesetzt. Als Heuristik wird eine Funktion gewählt, die die Anzahl der gefundenen SCOs und erreichten bzw. unerreichten Lernziele in Betracht zieht.

Nachdem durch den ersten Schritt eine erste Kursstruktur entstanden ist, welche aus Lernmodulen aus verschiedenen LORs besteht, wird im zweiten Schritt eine Anreicherung dieser Kette durch die Einbindung von Daten aus weiteren Wissensquellen vorgenommen (CourseEnrichtmentEngine). Der so entstandene erweiterte Kurs wird anschließend mittels spezieller Techniken auf ein einheitliches Layout gebracht (CoherenceEngine). Beschreibungen der CourseEnrichmentEngine und der CoherenceEngine sind nicht Bestandteil dieser Veröffentlichung. Im vierten und letzten Schritt wird dieser in adäquate SCORM-Strukturen gepackt, welches vom Lernmanagementsystem angezeigt werden kann.

Verwendung von Fachontologien

Die Anbindung an Ontologien basiert auf einer Erweiterung des am ZGDV entwickelten RetrievalSystems. Mit OntoResmo, dem Ontology Retrieval Server Module, wurde der Retrievalserver um Funktionen zum Zugriff auf ontologische Wissensstrukturen erweitert. OntoResmo ist ein HighLevel-Wrapper über der RDF-Datenbank Sesame, die einen effizienten Zugriff auf große Ontologien gewährleistet [Ru04]. Fachontologien, die die Föderation unterstützen sollen, können im RetrievalSystem als Retrievalquellen definiert werden und stehen über eine einheitliche API in WIESELfederation zur Verfügung.

3. Experimentalsystem

Der vorgestellte Lösungsansatz wurde in Kooperation mit dem Institut für Zellbiologie und Biosystemtechnik (Universität Rostock, Fachbereich Biowissenschaften, Prof. Weiss) für das Anwendungsfeld „Einführung in die Zellbiologie“ umgesetzt. Parallel zu den Arbeiten wurden in Zusammenarbeit mit der ANOVA Multimedia Studios GmbH mehrere SCORM-Kursmodule für die ergänzende Ausbildung in einer thematisch passenden Vorlesung umgesetzt. Das Drehbuch wurde von Prof. Weiss entworfen und von ANOVA in einzelne SCOs umgesetzt.

Als Benutzerschnittstelle wurde eine Erweiterung für das von der ANOVA Multimedia Studios GmbH entwickelte LMS COBilOT[2] umgesetzt. Nach der erfolgreichen Authentifizierung am LMS kann der Nutzer über den Menüpunkt „Kurs föderieren…“ die Dienste von WIESELfederation verwenden. Nach Definition des gewünschten Lernzieles und weiterer Randbedingungen für die Föderation wird über eine Webservice-Schnittstelle der Föderationsalgorithmus von WIESELfederation gestartet. Die entsprechenden LORs werden angefragt und ein Kurs entsprechend der Heuristik generiert. Die jeweiligen Ergebnisse werden in SCORM-konformen Modulen und die Module zu einem SCORM-konformen Kurs verpackt. Das Kurspackage wird als Resultat der Anfrage an das LMS ge-

[2] siehe auch http://www.cobilot.de

sendet, welches für die Verwaltung der Kursstrukturen und die Generierung der Navigationselemente verantwortlich ist.

4. Zusammenfassung und Ausblick

Der vorgestellte Ansatz erlaubt die automatische Zusammenstellung von E-Learning-Modulen aus verteilten Lernobjekt-Repositories zu SCORM-Kursen. Basierend auf der Definition des gewünschten Lernergebnisses wird mit wissensverarbeitenden Methoden, die zur Auswertung semantischer Beziehungen auf fachspezifische Ontologien zurückgreifen, eine Modulkette vorgeschlagen. Durch den Einsatz der Komponenten von WIESELfederation ebnet sich also der Weg für Lernmodulersteller, ihre erstellten Inhalte für weitere Nutzergruppen recherchierbar und einfach nutzbar zu machen. Auf der anderen Seite wird den Lernenden mit WIESELfederation und dessen Einbindung in ein bestehendes Lernmanagementsystem ein flexibles System geboten, mit dem E-Learning-Kurse, die an eigene Erfahrung und Vorlieben angepasst sind, automatisch erstellt werden können. Weitere Arbeiten müssen über die technische Ebene hinaus auch organisatorische und wirtschaftliche Aspekte von Content Sharing-Anwendungen betrachten.

Die Umsetzung in einem Experimentalsystem für Kursmodule aus dem biologischen Bereich demonstriert die Möglichkeiten der Föderation. Es ist geplant, das Experimentalsystem parallel zu einer stattfindenden Vorlesung Studenten zur Verfügung zu stellen und damit den Föderationsansatz mit einer größeren Nutzerzahl zu evaluieren. In Zusammenarbeit mit der ANOVA Multimedia Studios GmbH werden weitere Nutzungsszenarien und Themenbereiche aktiv untersucht.

Zukünftige Arbeiten befassen sich mit zahlreichen Problemen, die bei der Föderation von Lernobjekten auftreten. Zum einen existieren für ein Fachgebiet durchaus verschiedene Ontologien, die von verschiedenen Kompetenzträgern entwickelt wurden. Da häufig eine semantische Deckung oder zumindest semantische Überlappung gegeben ist, kann durch Algorithmen zur Ontologieharmonisierung (z. B. Aligning oder Merging) eine verschränkte Nutzung heterogener Ontologien erfolgen. Darüber hinaus werden weitere Fachontologien, aber auch generische Ontologien wie z. B. der WORDNET-Thesaurus in das System eingebunden.

Literatur

[ADL01] Advanced Distributed Learning Initiative: Sharable Content Object Reference Model (SCORM), http://www.adlnet.org, 2001

[Aeh05] Aehnelt, M.: Content Sharing als technologische Herausforderung, In: Tagungsband 13th European Conference and Specialist Trade Fair for Educational and Information Technology, LEARNTEC, 2005

[AS03] Audersch, S.; Schulz, M.: eFormsConnect - Gateway-Ansatz auf der Basis Semantic Web enabled Web Services,.In (Tolksdorf, R. (Hrsg.) u.a.): Proceedings Berliner XML-Tage 2003. Bonn, 2003, S. 257-268

[FR04] Flach, G., Rust, M.: WIESEL – semantikbasierter Ansatz zur Integration von Wissensmanagement und E-Learning, Knowtech, 2004

[IMS03] IMS Digital Repositories Specification, Version 1.0, IMS Global Learning Consortium, 2003, http://www.imsglobal.org/digitalrepositories/

[ND02] Neven, Filip, Duval Erik: Reusable Learning Objects - a Survey of LOM based Repositories, Multimedia ’02, 2002.

[Ru04] Rust, M.: BioVid - ontologiegestütztes, kontextsensitives Retrieval biologischer Videosequenzen. In (Dadam, Peter (Hrsg.) u.a.): Informatik verbindet - 34. Jahrestagung der Gesellschaft für Informatik e.v. (GI), Bonn, 2004, S. 54-58

[Se02] Seeberg, C.: Life Long Learning – modulare Wissensbasen für elektronische Lernumgebungen, Springer, Berlin, 2002

Towards Content Aggregation on Knowledge Bases through Graph Clustering

Christoph Schmitz

Knowledge and Data Engineering Group, Universität Kassel
http://www.kde.cs.uni-kassel.de/schmitz

Abstract

Recently, several research projects such as PADLR and SWAP have developed tools like Edutella or Bibster, which have been targeted at establishing peer-to-peer knowledge management (P2PKM) systems. In such a system, it is necessary for participants to provide brief descriptions of themselves, so that routing algorithms or matchmaking processes can make decisions about which communities peers should belong to, or to which peer a given query should be forwarded. In this talk, I propose the use of graph clustering techniques on knowledge bases for that purpose. After a brief round-trip over an ontology-based P2P knowledge management scenario, I will demonstrate the automatic generation of self-descriptions of peers' knowledge bases through the use of graph clustering. Viewing the knowledge base of a peer as a graph consisting of concepts and instances, one can employ clustering techniques to partition it into clusters of similar entities. From each cluster, the centroid can then be selected as a representative. This yields a list of entities giving an aggregated self description of the peer.

1 Ontology-Based P2P Knowledge Management

Recently, a lot of effort has been spent at integrating the upcoming research areas of peer-to-peer systems and the semantic web vision [12, 3, 1, 8], based on a notion of peer-to-peer, personal knowledge management (P2PKM for short). In such a scenario, users will model their knowledge – e.g., metadata on research papers they store on their computers – in personal knowledge bases, which can then be shared with other users via a peer-to-peer network.

One crucial point in such a P2P network is that in order to find relevant material to match a user's query, query messages need to be *routed* to peers which will be able to answer the query without flooding the network with unnecessary traffic.

Several proposals have been made recently as to how the network can self-organize into a topology beneficial for routing, and how messages can be routed in a P2PKM network based on the abovementioned scenario [10, 11, 6, 13]. All of these are based on the idea of routing indices as proposed in [2], adapted to the P2PKM scenario.

1.1 P2P Network Model

Following [10], we thus make the following assumptions about peers in a P2PKM network:

- Each peer stores a set of *content items*. On these content items, there exists a *similarity function* called *sim*. We assume $sim(i, j) \in [0, 1]$ for all items i, j, and the corresponding *distance function* $d := 1 - sim$ shall be a metric. For the purpose of this paper, we assume *content items* to be entities from a knowledge base (cf. Section 2.1), and the metric to be defined in terms of the ontology as described in section 2.2.
- Each peer provides a self-description of what it contains, in the following referred to as *expertise*. Expertises need to be much smaller than the knowledge bases they describe, as they are transmitted over the network and used in other peers' routing indices. A method of obtaining this expertise is outlined in Section 3.

- There is a relation *knows* on the set of peers. Each peer knows about a certain set of other peers, i. e., it knows their expertises and network address (IP, JXTA ID). This corresponds to the routing index as proposed in [2]. In order to account for the limited amount of memory and processing power, the size of the routing index at each peer is limited.
- Peers query for content items on other peers by sending query messages to some or all of their neighbors; these queries are forwarded by peers according to some *query routing strategy*. Using the *sim* function mentioned above, queries can thus be compared to content items and to peers' expertises.

1.2 Use Cases

Several use cases for P2PKM as sketched above have been implemented recently. In the PADLR and ELENA projects[1], a P2P infrastructure is established for the exchange of learning material among teachers and students; Bibster[2] is a tool for sharing BibTeX entries between researchers; the SCAM tool[3] for knowledge repositories can act as a peer in a P2P network.

These tools assume that peers agree beforehand on an ontology for describing their contents (e. g. the LOM standard for learning objects or the ACM Computing Classification System), and each peer builds a knowledge base on top of this ontology.

2 Ontologies and Metrics on Knowledge Base Entities

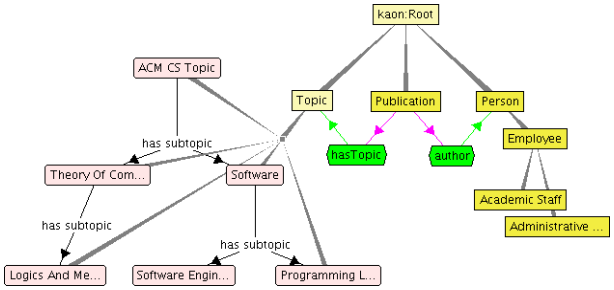


Figure 1: Example Ontology

2.1 Ontology Model

In short, an ontology is a formal conceptualization a group of stake-holders has agreed upon [5]. For the purpose of this paper, we use the view on ontologies proposed by the KAON[4] framework. Basically, an *ontology* consists of concepts with an is-a partial order, and relations between concepts[5]. A *knowledge base* or *OIModel* consists of an ontology and instantiations of concepts and relations. Concepts and instances are both called *entities* (for details cf. [4]).

Two additional important features of KAON OIModels are the attachment of lexical information (labels, descriptions, synonyms, etc.) to entities, and the possibility of nesting OIModels, so that the including model can refer to entities of the included one.

2.2 A Family of Ontology-Based Metrics

An ontology of the kind described above can be viewed as a graph: the set of node comprises the entities, and the relations, relation instances, the is-a and instance-of relationships make up the set of edges. An edge between entities in this graph expresses relatedness in some sense: the instance `phdstud1` may be related to the concepts `PhDStudent` (by an instance-of

[1] `http://www.l3s.de` [2] `http://bibster.semanticweb.org` [3] `http://scam.sourceforge.net/`
[4] `http://kaon.semanticweb.org` [5] We are following the simplified nomenclature of [4].

edge), `PhDStudent` and `Professor` would be connected by an edge due to the `supervises` relation, etc.

On this kind of semantic structure, [9] has proposed to use the distance in the graph-theoretic sense (lengths of shortest paths) as a semantic distance measure. We follow this suggestion and apply it to the abovementioned graph as follows:

- To each edge, a length is assigned; in order to account for the different kinds of relationships, taxonomic edges (is-a, instance-of) get smaller lengths than non-taxonomic edges. This reflects the fact that `is-a(PhDStudent,Person)` would be considered a closer link between these concepts than, say, `rides(Person,Bicycle)`.
- Edge lengths are divided by the average distance from the root concept of the incident nodes. This reflects the intuition that top-level concepts such as `Person` and `Project` would be considered less similar than, e.g., `Graduate Student` and `Undergraduate` farther from the root.
- The lengths are normalized such that the longest distance in the graph equals 1, so that $1 - d = sim$ holds.

2.3 Caveats and Pitfalls on Real-World Ontologies

While these strategies of deriving metrics from semantic structures seem straightforward, applying them to ontologies used in real-world applications can turn out to be non-trivial:

Noise and Technical Artifacts Not all of the content of a knowledge base may be genuinely taking part in the user's view of a certain domain; e. g., in KAON lexical information is represented as first-class entities in the knowledge base. This leads to a large number of entities which are not relevant for the semantic distance computation. Similarly, there may a root class which every entity is an instance of, which would render our approach to calculating distances useless.

Modeling Idiosyncrasies Engineering an ontology implies making design decisions, e. g. whether to model something as an instance or as a concept [14]. These decisions carry implications for the weighting of edges, e. g. if a taxonomic relationship is expressed by a special relation which is not one of `instance-of, is-a`.

To overcome these problems, we have implemented extensive entity filtering and weighting customization strategies which are applied prior to the metric computation itself.

3 Graph Clustering for Content Aggregation

As mentioned above, a peer needs to provide an expertise in order to be found as an information provider in a P2PKM network. From the discussion above, the following requirements for an expertise can be derived:

- The expertise should be provide an aggregated account of what is contained in the knowledge base of the peer, meaning that using the similarity function, a routing algorithm can make good a-priori guesses of what can or cannot be found in the knowledge base.
- The expertise should be orders of magnitude smaller than the knowledge base itself, because it will be used in routing indices.

We propose the use of a version of *k-modes clustering* [7] for this purpose.

3.1 k-Modes Clustering

In short, k-modes clustering works for partitioning a set S of items into k clusters works as follows:

1. Given k, choose k elements of the S as *centroids*
2. Assign each $s \in S$ to the centroid C_i minimizing $d(C_i, s)$

3. For $i = 1 \ldots k$, recompute C_i as such that $\sum_{s \text{ assigned to } C_i} d(C_i, s)$ is minimized.
4. Repeat steps 2 and 3 until centroids converge.

This algorithms yields (locally) optimal centroids which minimize the average distance of each centroid to its cluster members. A variation we will use is *bi-section k-modes clustering*, which produces k clusters by starting from an initial cluster containing all elements, and then recursively splitting the cluster with the largest variance with k-modes until k clusters have been reached.

In order to apply this algorithm in our scenario, some changes need to be made:

- The set S to be clustered consists only of those parts of the knowledge base which are not shared between peers; otherwise, the structure of the shared part (which may be comparatively large) will shadow the interesting structures of the private part.
- The centroids will not be chosen from S, but only from the shared part of the ontology. Otherwise, other peers could not interpret the expertise of the peer.

3.2 Example of Knowledge Base Aggregation

As an example, consider a P2PKM network of researchers with knowledge bases about publications according to the ontology shown in Figure 1. Every publication would be related to its corresponding topics from the ACM Computing Classification System[6].

We consider a researcher from DBLP[7] with a sufficient number of papers available on-line, in this case Gruia-Catalin Roman from Washington University. 13 of his papers available with classification at the ACM web site were modeled in a knowledge base, and the clustering algorithm described was run. Table 1 shows some examples of the centroids which were extracted. Note that the k-modes algorithm is non-deterministic because of the random initialization.

k	*Centroids*
2	Network Architecture And Design, Software/Program Verification
2	Requirements/Specifications, Computer-Communication Networks
3	Network Architecture And Design, Requirements/Specifications, Operating Systems
3	Network Architecture And Design, Programming Techniques, Software/Program Verification

Table 1: Centroids for different values of k

A brief examination of the papers and Prof. Roman's home page[8] shows that these centroids indeed reflect his interests of software engineering on the one hand and mobile, distributed computing on the other.

4 Discussion and Work in Progress

In the previous sections, a way of extracting expertises from knowledge bases in a P2PKM setting based on graph clustering was proposed. These expertises consist of entities from the shared part of the ontology which are computed as the centroids in a k-modes clustering procedure. This clustering provides a (locally) optimal set of centroids with respect to the average semantic distance of centroids to knowledge base entities.

While this talk provides anecdotical hints of how the clustering procedure extracts suitable expertises, we are currently conducting a thorough evaluation of this method in conjunction with self-organization techniques for P2PKM networks as described in [10]. Note that usually (e. g. in the text summarization community), the value of aggregations or summaries is measured by evaluating it against human judgment. In our case, however, the aggregations

[6] `http://www.acm.org/class/1998/` [7] `http://dblp.uni-trier.de`

[8] `http://www.cs.wustl.edu/~roman/`

will be evaluated with regard to their contribution to improving the performance of the P2P network.

Other ongoing research questions include the treatment of literals (e. g. looking for an instance of `PhDStudent` with a last name "Schmitz") in this metric and the self-organization scheme relying on it, and the formation and labeling of topical communities in the P2PKM network.

References

[1] M. Bonifacio, R. Cuel, G. Mameli, et al. A peer-to-peer architecture for distributed knowledge management. In *Proceedings of the 3rd International Symposium on Multi-Agent Systems, Large Complex Systems, and E-Businesses MALCEB'2002*. Erfurt, Germany, 2002.

[2] A. Crespo and H. Garcia-Molina. Routing indices for peer-to-peer systems. In *Proceedings of the International Conference on Distributed Computing Systems (ICDCS)*. Vienna, Austria, 2002.

[3] M. Ehrig, P. Haase, F. van Harmelen, et al. The SWAP data and metadata model for semantics-based peer-to-peer systems. In M. Schillo, M. Klusch, J. P. Müller, et al. (eds.), *Proceedings of MATES-2003. First German Conference on Multiagent Technologies*, vol. 2831 of *LNAI*, pp. 144–155. Springer, Erfurt, Germany, 2003.

[4] M. Ehrig, S. Handschuh, A. Hotho, et al. KAON - towards a large scale Semantic Web. In K. Bauknecht, A. M. Tjoa, and G. Quirchmayr (eds.), *Proc. E-Commerce and Web Technologies, Third International Conference, EC-Web 2002*, no. 2455 in LNCS. Springer, Aix-en-Provence

[5] T. R. Gruber. Toward principles for the design of ontologies used for knowledge sharing. In *Proceedings of the International Workshop on Formal Ontology*. Padova, Italy, 1993.

[6] P. Haase and R. Siebes. Peer selection in peer-to-peer networks with semantic topologies. In *Proceedings of the 13th International World Wide Web Conference*. New York City, NY, USA, 2004.

[7] Z. Huang. Extensions to the k-means algorithm for clustering large data sets with categorical values. *Data Min. Knowl. Discov.*, 2(3):283, 1998.

[8] W. Nejdl, B. Wolf, C. Qu, et al. Edutella: A p2p networking infrastructure based on rdf. In *Proceedings of the 11th International World Wide Web Conference (WWW 2002)*. Honolulu, Hawaii, 2002.

[9] R. Rada, H. Mili, E. Bicknell, et al. Development and application of a metric on semantic nets. *IEEE Transactions on Systems, Man and Cybernetics*, 19(1):17, 1989.

[10] C. Schmitz. Self-organization of a small world by topic. In *Proc. 1st International Workshop on Peer-to-Peer Knowledge Management*. Boston, MA, 2004.

[11] C. Schmitz, S. Staab, and C. Tempich. Socialisation in peer-to-peer knowledge management. In *Proc. International Conference on Knowledge Management (I-Know 2004)*. Graz, Austria, 2004.

[12] J. Tane, C. Schmitz, and G. Stumme. Semantic resource management for the web: An elearning application. Proc. 13th International World Wide Web Conference (WWW 2004), 2004.

[13] C. Tempich, S. Staab, and A. Wranik. Remindin': Semantic query routing in peer-to-peer networks based on social metaphors. In W3C (ed.), *Proceedings of the 13th International World Wide Web Conference (WWW 2004)*, pp. 640–649. ACM, New York, USA, 2004.

[14] C. A. Welty and D. A. Ferrucci. What's in an instance? Tech. Rep. #94-18, RPI Computer Science Dept., 1994.

Dimensionsreduktion als Konzept der interaktiven Suche in Bilddatenbanken

Anke Schneidewind
e-mail: a.schneidewind@iti.cs.uni-magdeburg.de
Institut für Technische und Betriebliche Informationssysteme (ITI)
OvG-Universität Magdeburg
Universitätsplatz 2, 39106 Magdeburg, Deutschland

Zusammenfassung

Ähnlichkeitsanfragen an Bilddatenbanken sind bis heute problematisch geblieben. Eines der Hauptprobleme ist die semantische Lücke zwischen digitaler und visueller Ähnlichkeit. Ein weiteres Problem ist der „Fluch der hohen Dimensionen", der aufgrund der vielen Dimensionen der Bild-Feature-Vektoren auftritt. Auswirkungen sind hohe Kosten für die Ähnlichkeitssuche und eine langwierige Nutzer-Computer-Interaktion, die oft in einer kleinen unbefriedigenden Bildmenge stagniert.

Das vorliegende Konzept zum Durchsuchen einer Bilddatenbank soll Vorzüge bekannter Strategien nutzen und neben einem umfassenden Einblick des Nutzers in die Datenbank, möglichst kurze Antwortzeiten während der Interaktionen unterstützen. Bei der Suche bestimmt der Nutzer zwei Bilder die für ihn ähnlich und relevant sind. Durch Projektion werden die Bilder im Vektorraum zueinander gebracht. Wird die Projektion auf sämtliche Bilder der Datenbank angewendet wird der Vektorraum um eine Dimension reduziert. Dadurch kommen sehr weit entfernte Bilder in den Betrachtungskreis des Nutzers. Der Nutzer kann so schnell einen Überblick über den Datenbestand erhalten.

Ziel ist ein Retrievalsystem mit kurzen Antwortzeiten, dass ein browsendes Navigieren erlaubt, und dabei die Vielfalt möglicher Ähnlichkeiten nicht einschränkt.

1 Einleitung und Motivation

Es gibt zwei Fälle in denen Anfragen an Bilddatenbanken zwingend in Form eines Beispielbildes gestellt werden. Zum einen wenn in der Bilddatenbank Bilder ohne zusätzliche Annotationen gespeichert werden und zum anderen wenn der Nutzer keine textuelle Information hat. Im ersten Fall kann durch aufwändige manuelle oder semi-automatische Annotation [4] der Datenbestand so erweitert werden, dass die Ähnlichkeitssuche vom effektiveren und effizienteren, Text-Retrieval unterstützt werden kann.

Der zweite Fall tritt z.B. bei der LostArt-Datenbank[1] [7] auf. Anhand eines digitalen Bildes eines unbekannten Kunstobjektes werden ähnliche Bilder in der Datenbank gesucht. Wird das Kunstobjekt in der Datenbank gefunden erfährt der Nutzer alles Wissenswerte darüber. Ein anderes Szenario ist eine Datenbank als Fremdenführer für Reisende[2]. Digitale Reisefotos werden an eine zentrale Datenbank gesendet, um z.B. Informationen über unbekannte Gebäude und deren Umgebung zu erhalten. In diesen Szenarien ist der Sucherfolg allein von der Qualität des Retrievals auf extrahierten Merkmalen, den so genannten Feature, der Bilder abhängig.

Die Ähnlichkeit von Bildern ist subjektiv, sie ist abhängig von gesuchten Bildinhalten und abhängig von der beurteilenden Person. Je nach dem sind sehr unterschiedliche visuelle Merkmale der Bilder bei der Ähnlichkeitsbestimmung ausschlaggebend. Eines der Ziele im Bild-Retrieval

[1] Entwicklung an der Universität Magdeburg: http://mmdb.cs.uni-magdeburg.de/lostart.shtml.de
[2] Entwicklung an der Universität Bonn: http://www.ipb.uni-bonn.de/FotoNav/index.html

ist daher, möglichst viele visuelle Merkmale durch automatisch extrahierbare Feature abzubilden. Auch wenn nicht alle visuellen Merkmale durch Feature abgebildet werden, geht man davon aus, dass ähnliche Bilder durch eine Feature-Kombination mit ähnlichen Werten gefunden werden können. Als Maß der Ähnlichkeit zwischen zwei Bildern wird eine errechnete Distanz zwischen den zugehörigen Features genommen.

In der Regel werden in einer iterativen Suche mittels Relevance Feedback Nutzerpräferenzen immer wieder neu abgefragt und in die Berechnung der nächsten Anfrage mit einbezogen. Das Feedback des Nutzers wird dazu genutzt die für die Ähnlichkeit ausschlaggebenden Feature(-Kombinationen) zu ermitteln.

Grundsätzliches Problem ist die große Anzahl der Feature und der sich daraus ergebenden hohen Kosten für den Ähnlichkeitsvergleich und die Suche. Weiteres Problem ist die unzureichende Abbildung von visuellen Merkmalen auf extrahierte Feature und das die von Feature unabhängige Semantik eines Bildes keinen Einfluss auf die Ergebnismenge hat.

Das in dem Papier vorgestellte Konzept bietet eine, in den Antwortzeiten angemessene, interaktive Suche an. Dabei kann der Nutzer mit wenigen Klicks unterschiedliche Dimensionen aus dem Feature-Raum entfernen und danach die veränderte Ergebnismenge auswerten.

2 Grundlagen und relevante Arbeiten

Eine der oft aufgegriffenen Möglichkeiten die rechenintensive Suche zu beschleunigen ist die Reduktion der Feature [3]. Dazu gehören beispielsweise FastMap [2] und Karhunen-Loeve-Transformation oder die Dimensionsreduktion als Minimierungsproblem [6]. Durch eine hohe Anzahl von extrahierten Feature-Werten pro Bild, entsteht ein hochdimensionaler Raum in dem Bilder anhand ihres Feature-Vektors als Punkt dargestellt werden können. Die Reduktionsverfahren reduzieren den hochdimensionalen Raum in dem sie eine kleine(re) Anzahl von (statistisch) aussagekräftigen Dimensionen ermitteln. Vorteil dieser Verfahren ist, dass sie nicht nur eine schnellere Suche aufgrund weniger Feature-Werte ermöglichen, sondern so auch das Problem des „Fluches der hohen Dimensionen“ umgehen.

Eine Besonderheit von FastMap ist, dass der Algorithmus auf Distanzen arbeitet. Die extrahierten Feature werden nur einmalig zur Berechnung einer Distanzmatrix benutzt. Das Ermitteln von Distanzen in einem neu gewählten Koordinatensystem erfolgt durch Projektion auf die neuen Dimensionen mittels der bekannten Dreiecksgleichungen wie Pythagoras und Kosinussatz. Die Berechnung der Ähnlichkeit bzw. Distanz zwischen Bildern zur Laufzeit entfällt.

Nachteil der Reduktionsverfahren ist, dass mit weniger Dimensionen auch weniger Variationen der Ähnlichkeit möglich sind. So wird ein gesuchtes Bild nicht mehr gefunden, wenn die visuell vorhandene Ähnlichkeit zum Anfragebild nicht mehr von den noch zur Verfügung stehenden Dimensionen abgebildet werden kann. Die vorgestellten Reduktionsverfahren eignen sich auch nicht direkt für eine ähnlichkeitsorientierte Dimensionsreduktion zur Laufzeit, da die Kosten meist über $O(N^2)$ liegen.

Die Anpassung der Anfrageergebnisse an Nutzerpräferenzen erfolgt in der Regel durch Verschieben des Anfragepunktes [5] und durch variables Gewichten der Feature[4]. Während die Verschiebung im ansonsten starren Feature-Raum erfolgt, beeinflusst die Gewichtung, die Distanzen zwischen den Bildern und sogt so für eine variable Ähnlichkeit. Nachteile liegen wiederum in den Kosten der Neuberechnung der Ergebnismenge. Sie ist abhängig von der Komplexität der Feature-Analyse für Gewichtung und Verschiebung. Ein Problem der einfachen Dimensionsgewichtung von Feature liegt in der Raumverzerrung. Liegen die visuell ähnlichen Bilder relativ weit auseinander und ungünstig im Raum müssen dementsprechend viele Dimensionen in ihrer Ausdehnung durch Gewichtung gestaucht werden, was letztlich nur zu einer Verkleinerung des gesamten Raumes führt und nicht unbedingt einen wesentlichen Einfluss auf die Ergebnismenge hat, sieheAbbildung 1a. Auch ist die Erreichbarkeit von weit außerhalb liegenden Bildern weder durch die Gewichtung noch durch die Anfragepunktsverschiebung innerhalb einer angemessenen

Zeit gewährleistet. Außerdem wird die Relevanzbewertung von Bildern oder gar Feature vom Nutzer oft als lästig oder schwer interpretierbar empfunden.

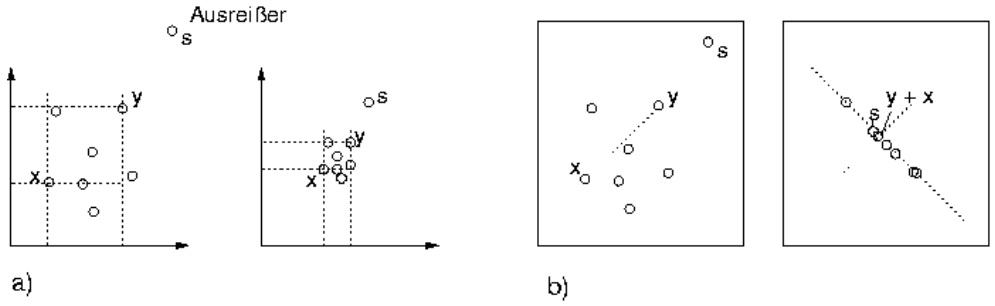


Abbildung 1: Unterschied zwischen a) Gewichtung im Feature-Raum und b) Projektion im Distanzraum

3 System Architektur

Die Umgebung für die vorgestellte Bildsuche lässt sich in vier Bereiche einteilen, siehe Abbildung 2. Eine Nutzerschnittstelle für die Ähnlichkeitsanfrage und die Initialisierung der Suche, einen Indexierungsbereich in dem für den effizienten Zugriff auf die Bilder gesorgt wird, die Datenhaltung welche die Bilder, Feature und Indices verwaltet und einen Bereich in dem Anfragen visualisiert werden und in dem eine Feedback orientierte Interaktion mit dem Nutzer ermöglicht wird.

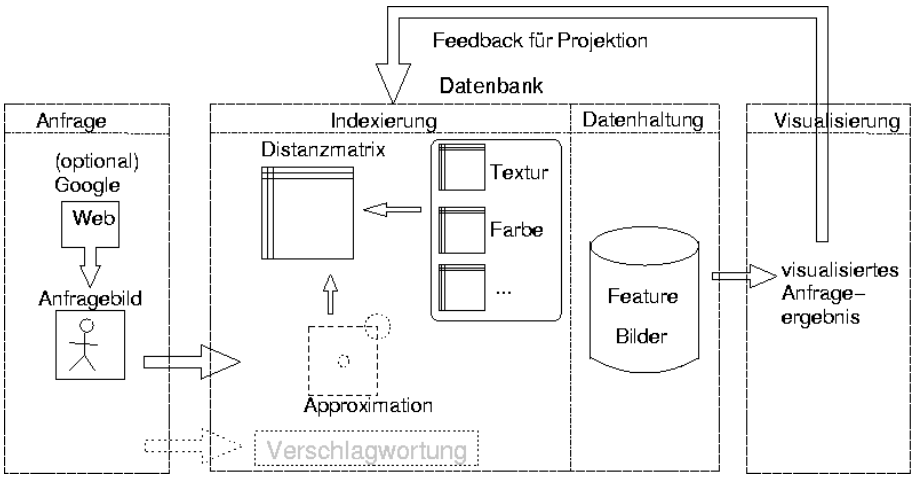


Abbildung 2: Systemarchitektur

Der Bereich für die Anfrage erlaubt das Hochladen eines Anfragebildes. Fehlen dem Nutzer Beispielbilder kann er über einen integrierten Aufruf von GoogleImage[3] geeignete Bilder per Texteingabe suchen. Bei dieser Vorgehensweise bietet es sich natürlich an, Texte zur Annotation für die Datenbank automatisch aus den relevanten Internetseiten zu extrahieren und abzuspeichern [4]. Soll aber an dieser Stelle nicht weiter betrachtet werden. Für die initiale Anfrage wird über einen, an der Universität Magdeburg entwickelten, Approximationsindex [1] auf die nächsten Nachbarn des Anfragebildes zugegriffen. Die so ermittelte Ergebnisbildmenge wird im Visualisierungsbereich auf unterschiedlichen Abstraktionsebenen visualisiert. Dadurch soll ein Einblick in das Verhältnis von visueller und feature-basierter Ähnlichkeit ermöglicht werden. Dies kann wiederum als Entscheidungshilfe für die nächste Bildwahl dienen, worauf in [8] näher eingegangen

[3] http://images.google.de

wird. Im Bereich der Visualisierung wird auch die gesamte Interaktion zwischen Nutzer und Computer abgewickelt. Dazu gehört vor allem die Wahl zweier ähnlicher Bilder für die darauf folgende Projektion im Feature-Raum.

Der Feature-Raum ist durch Distanzmatrizen im Indexierungsbereich abgebildet. Jede Zeile bzw. Spalte jeder Matrix enthält sämtliche Distanzen zu je einem Bild der Datenbank. Für jeden extrahierten Feature-Vektor (Farbe, Textur, Kontur...) wird eine Distanzmatrix über alle Bilder der Datenbank abgelegt. Diese werden normalisierter und in einer übergeordneten Distanzmatrix zusammengefasst. Die weitere Suche, Visualisierung und Anpassung von Anfrageergebnissen erfolgt auf den Distanzen dieser Maritx.

4 Projektion im hochdimensionalen Raum

Die eingesetzte Distanzmatrix D enthält alle möglichen $\frac{N^2}{2} - N$ Distanzen $d(B_i, B_j)\, kurz\, d_{ij}$ zwischen den N Bildern $B1 \ldots Bn$ der Datenbank, siehe Abbildung 3a). In dem Beispiel Abbil-

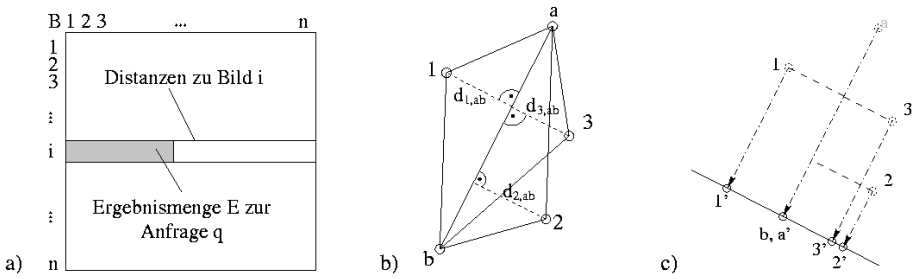


Abbildung 3: Distanzmatrix.

dung 3 erhält der Nutzer die visualisierte Ergebnismenge E zur Anfrage q aus der Zeile i der Distanzmatrix. Hat der Nutzer zwei relevante Datenbankbilder a und b in der Visualisierung entsprechend dem Beispiel in Abbildung 3b) ausgewählt, wird die Dimension, die durch die zwei Bilder im Feature-Raum, definiert ist entfernt, Abbildung 3c). Dazu wird mittels der abgelegten Distanzen (angedeutet durch Linien) die Distanzen zur Dimension ab für alle Bilder der Zeile i außerhalb der Ergebnismenge berechnet:

$$d_{i,ab} = \sqrt{d_{ia}^2 - \left(\frac{d_{ia}^2 + d_{ab}^2 - d_{ib}}{2d_{ab}}\right)}$$

Da die Projektion eine ganze Dimensionen entfernt, können in einem Projektionsschritt weit außerhalb liegende Bilder in den Betrachtungskreis des Nutzers kommen, siehe Abbildung 1b. Die Analyse der Feature in relevanten Bildern für eine Gewichtung der Feature, wie in Abbildung 1, fällt weg, da durch die Projektion der Bilder aufeinander alle Varianzen zwischen den Bildern in einem Schritt aus dem Feature-Raum entfernt werden. Das Verschieben des Anfragepunktes erfolgt durch die Wahl eines neuen Anfragepunktes aus der neuen Ergebnismenge. Bei der Wahl eines, im Feature-Raum, weit entfernten Bildes wird die, aus der Distanzmatrix erzeugte, Ergebnismenge stark von der Startmenge verschieden sein. Da sich das Entfernen der Dimension nur auf die Visualisierung auswirkt wird die Variabilität der Ähnlichkeit für die nächsten Suchschritte nicht eingeschränkt.

Durch Speichern der Bilder a und b kann an jeder beliebigen Stelle der Suche erneut auf diese Dimensionsreduktion zurückgegriffen und auf die aktuell visualisierte Matrixzeile angewendet werden. Dabei müssen die Bilder a und b nicht in der Ergebnismenge erscheinen.

5 Zusammenfassung und Ausblick

Das vorgestellte Konzept vereint unter einer einfachen Dimensionsreduktion eine schnelle, benutzerfreundliche Ähnlichkeitsanpassung. Die schnelle Projektion ersetzt die zeitaufwändige Analyse von Feature-Vektoren und Neuberechnung von Distanzen. Die Ähnlichkeit ist durch viele Feature-Werte sehr variabel, was sich aufgrund der Distanzmatrix trotzdem nicht auf die Geschwindigkeit der Berechnungen auswirkt. Durch das Entfernen ganzer Dimensionen werden auch weit außerhalb liegende Bilder mit in den Anfrageraum gezogen, wodurch der Nutzer einen Überblick über die gesamte Datenbank erhalten kann und die Suche nicht in einem kleinen Bereich stagniert.

Logische Weiterführung der Arbeit wäre die Speicherung von Informationen aus den einzelnen Suchprozessen. Hat der Nutzer eine bestimmte Bildkombination zur Dimensionsreduktion mehrfach angewendet bzw. wurde die Reduktion als positiv bewertet, kann dies auf die Feature $v_1 \dots v_m$ der Datenbank abgebildet werden und zu Zeiten mit geringer Last, die Distanzmatrix neu berechnen. Die Neuberechnung der Feature erfolgt einfach mit Hilfe der Distanzvektoren der Feature-Vektoren $v_{i,dist} = v_{i,a} - v_{i,b}$ der zwei ausgewähltenBilder.

Auf dieser Basis werden dann Anfragen und Ähnlichkeitscharakteristika erlernt- und reproduziert.

Literatur

[1] S. Balko. *Grundlagen, Entwicklung und Evaluierung einer effizienten Approximationstechnik für Nearest-Neighbor-Anfragen im hochdimensionalen Vektorraum.* PhD thesis, Otto-von-Guericke-Universität Magdeburg, 2004. in German.

[2] Christos Faloutsos and King-Ip Lin. Fastmap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. In Michael J. Carey and Donovan A. Schneider, editors, *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data, San Jose, California, May 22-25, 1995*, pages 163–174. ACM Press, 1995.

[3] I. K. Fodor. A survey of dimension reduction techniques. Technical Report UCRL-ID-148494, Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, 2002.

[4] Ye Lu, Chunhui Hu, Xingquan Zhu, HongJiang Zhang, and Qiang Yang. A unified framework for semantics and feature based relevance feedback in image retrieval systems. In *MULTIMEDIA '00: Proceedings of the eighth ACM international conference on Multimedia*, pages 31–37, 2000.

[5] Henning Müller, Patrick Ruch, and Antoine Geissbuhler. Enriching content-based image retrieval with multi-lingual search terms. volume 54, pages 6–11, 2005.

[6] Yossi Rubner, Leonidas J. Guibas, and Carlo Tomasi. The earth mover"s distance, multi-dimensional scaling, and color-based image retrieval. In *Proceedings of the ARPA Image Understanding Workshop*, pages 661–668, 1997.

[7] E. Schallehn, I. Schmitt, and N. Schulz. Visual Retrieval for Searching in a LostArt Metasearch Engine System. In *Int. Workshop on Electronic Imaging & Visual Arts (EVA'02), November 6-8, Berlin*, 2002.

[8] Anke Schneidewind, Petra Neumann, and Ingo Schmitt. An approach to visualize image retrieval results. In *Proceedings MDDE '04, 4th International Workshop on Multimedia Data and Document Engineering, Washington, DC, USA, July 2nd 2004*, 2004.

Persistenz- und Transaktionskonzepte in $\mathbf{XOBE}_{\mathrm{DBPL}}$

Henrike Schuhart
Institut für Informationssysteme
Universität zu Lübeck
schuhart@ifis.uni-luebeck.de

Zusammenfassung

XML als Datenaustauschformat zwischen verschiedenen Anwendungen gewinnt mehr und mehr an Bedeutung. Heutige Applikationen manipulieren und operieren im Allgemeinen mit Hilfe von Objekten. Ein Austausch erfordert also einen Mapping-Prozess sowohl zwischen Objekten und XML als auch umgekehrt. Falls eine Applikation darüber hinaus dieselben Daten auch persistent speichern will, kommen häufig aus Performance-Gründen relationale Datenbanken zum Einsatz. Das heißt, es findet ein weiterer Mapping-Prozess zwischen den Objekten der Programmiersprache und den Tabellen der relationalen Datenbank statt, ebenfalls in beide Richtungen. Programmierer einer solchen Applikation müssen daher viele verschiedene Schnittstellen verwenden bzw. komplexe Frameworks benutzen. In dem $\mathbf{XOBE}_{\mathrm{DBPL}}$ (**X**ML **OBjE**cts **D**ata**B**ase **P**rogramming **L**anguage) Projekt wird eine Sprache basierend auf Java entwickelt, die diese Mapping-Prozesse überflüssig macht. Java wird syntaktisch und semantisch um XML, XPath und Update-Ausdrücke erweitert. Desweiteren wird eine persistente Umgebung eingeführt, die transparente und typunabhängige Persistenz integriert. Datenkonsistenz vor allem im Zusammenhang mit Mehrbenutzerbetrieb wird durch Transaktionsintegration in Java erreicht.

1 Einleitung

XML gewinnt als Datenaustauschformat zwischen verschiedenen Applikationen immer mehr an Bedeutung. Die Anwendungslogik manipuliert und bearbeitet Daten allerdings in Form von Objekten. Ein Mapping-Prozess in beide Richtungen ist erforderlich. Zumeist werden Daten aber auch persistent in objekt-relationalen Datenbanken abgelegt. Ein zweiter Mapping-Prozess wird zwischen den Objekten und Tupeln der relationalen Tabellen benötigt. Es existieren zwar viele verschiedene APIs und komplexe Frameworks, um derartige Applikationen zu entwickeln, aber kein transparenter und einheitlicher Ansatz. Im $\mathbf{XOBE}_{\mathrm{DBPL}}$ Projekt soll dieses gelöst werden. Java wird als Basis verwendet, um eine neue Datenbankprogrammiersprache für XML-Anwendungen zu entwickeln, die transparente und typunabhängige Persistenz und Konsistenz in sich vereint.

In Kapitel 2 werden verwandte Arbeiten kurz vorgestellt. Kapitel 3 gibt einen Überblick über die Integration von XML in Java. Das Kapitel 4 führt das transparente und typunabhängige Persistenzkonzept, das in $\mathbf{XOBE}_{\mathrm{DBPL}}$ zum Einsatz kommt, ein. Im Zusammenhang mit persistenten Daten und mehreren Benutzern stellt sich die Frage der Konsistenz, hierzu stellt Kapitel 5 das grundlegende Konzept in $\mathbf{XOBE}_{\mathrm{DBPL}}$ vor. Schließlich wird in Kapitel 6 die derzeitige Implementierung des Präprozessors und der Laufzeitumgebung erläutert. Ein Ausblick schließt dieses Papier. Weitere Details lassen sich den Arbeiten [5],[10] und [11] entnehmen.

2 Verwandte Arbeiten

Persistentes XML. In Programmiersprachen wird persistentes XML bis jetzt noch nicht direkt unterstützt. Daher ist expliziter Code erforderlich, um mit einer Datenbank zu kommunizieren und XML

persistent ablegen zu können. Ein Ansatz besteht darin, XML auf relationale Tupel oder Blobs abzubilden und in objekt-relationalen Datenbanken wie Oracle [7] oder DB2 [3] zu speichern. Trotz einiger XML-Erweiterungen operieren Anwendungen entweder mittels konventionellem JDBC oder exportierten DOM- oder SAX-Schnittstellen. All diese Schnittstellen unterstützen keine statische Typüberprüfung. Mit denselben Problemen sind auch native XML-Datenbanken wie Xindice [1], Infonyte DB [4] oder Tamino [9] behaftet.

Datenbankprogrammiersprachen. Nach unserem besten Wissen existiert bis jetzt noch keine Datenbankprogrammiersprache, die das XML-Datenmodell in eine objekt-orientierte Programmiersprache integriert. Unter den Ansätzen, die das relationale Modell integrieren, ist insbesondere DBPL [8] zu nennen und dessen Nachfolgerprojekt Tycoon [6].

3 XML Integration

In diesem Abschnitt soll die Syntax und Semantik der XML Integration in **XOBE**$_{\text{DBPL}}$ auf informelle Art kurz vorgestellt werden. **XOBE**$_{\text{DBPL}}$ erweitert die objekt-orientierte Programmiersprache Java um Sprachkonstrukte, die XML-Fragmente und insbesondere XML-Dokumente verarbeiten. XPath, um bestehende XML-Objekte zu traversieren, und Update-Ausdrücke, um diese zu manipulieren, werden eingeführt. In **XOBE**$_{\text{DBPL}}$ werden XML-Fragmente, die einem gegebenen Schema entsprechen, mittels XML-Objekten repräsentiert. XML-Objekte sind also Objekte erster Klasse. Die deklarierten Schemata, die die Typen der XML-Objekte angeben, können entweder XML Schemata oder DTDs sein. Im Folgenden soll eine Beispiel-Anwendung erläutert werden, die einen Buchladen modelliert. Bücher sind XML Elemente, deren Schema-Beschreibung in Listing 2 zu sehen ist. Eine erste Version der Klasse **Buchladen** ist in Listing 1 gegeben.

Listing 1: Klassen-Version des Buchladens

```
1  ximport Buecher.xsd;
2  public class Buchladen implements Shop{
3   //Deklaration einer XML Membervariablen
4   xml<buecher> shop_buecher;
5   ...
6   public xml<buch> registriereBuch(String titel, List autoren, String isbn, int preis){
7    //ein XML Objekt-Konstruktor, der ein Buch erzeugt
8    xml<buch> b = <buch isbn={isbn}>
9                    <titel>{titel}</titel>
10                   <preis>{preis}</preis>
11                 </buch>;
12   registriereAutoren(b,autoren);
13
14   if(!buchRegistriert(isbn)){
15    //ein XML Update, das ein Buch einfuegt
16    $UPDATE shop_buecher INSERT {b}$;
17   }
18   return b;
19  }
20  private void registriereAutoren(xml<buch> buch, List autoren){
21   for(int i=0; i<autoren.size(); i++){
22    //ein XML Update, das den iten Autor einfuegt
23    $UPDATE buch INSERT {(String)autoren.get(i)}$;
24   }
25  }
26  public boolean buchRegistriert(String isbn){
27   //ein XPath-Ausdruck, der nach einem Buch mit gegebener ISBN sucht
28       xml<buch*> list = $shop_buecher/buch[@isbn={isbn}]$;
29       if(list.getLength()>0)return true;
30       return false;
31  }
32  public void deregistriereBuch(String isbn){
33   $UPDATE shop_buecher DELETE /buch[@isbn={isbn}]$;
34  }
35 }
```

Die Methode `registriereBuch` in den Zeilen 6-19 erzeugt und registriert ein Buch im Buchladen. In den Zeilen 8-11 wird ein XML-Objekt-Konstruktor verwendet, um ein XML-Buch-Element zu erzeugen. Im Allgemeinen beginnt eine XML-Typ Deklaration mit dem Schlüsselwort `xml` gefolgt von spitzen Klammern. Innerhalb der Klammern steht entweder ein einzelner Schematypbezeichner gefolgt von einem optionalen Stern oder ein Choice-Typ. Die verwendeten XML-Schematypen werden wie in Zeile 1 zu sehen mittels einer `ximport`-Deklaration bekanntgegeben. Auf XML-Objekt-Bestandteile kann mittels eines XPath-Ausdrucks zugegriffen werden. In Zeile 28 der Methode `buchRegistriert` wird die XML Membervariable `shop_buecher` nach einem Buch mit einer gegebenen ISBN durchsucht. Nur wenn das Ergebnis dieser Anfrage eine leere XML Objektliste ist, liefert die Methode ein `true` zurück. Update-Ausdrücke werden in den Zeilen 16,23, und 33 verwendet. Der erste Ausdruck in Zeile 16 fügt ein neues Buchelement als Kind in das XML-Element `shop_buecher` ein und der zweite, eingebettet in eine Java for-Schleife, in Zeile 23 fügt eine Liste von Autorennamen ein. Zeile 33 zeigt eine Lösch-Operation eines Buchs anhand dessen ISBN. In **XOBE**$_{\text{DBPL}}$ stehen darüberhinaus noch eine Umbenennungs- und eine Ersetzungs-Operation zur Verfügung, sowie eine Kombination aller dieser Basisoperationen.

Listing 2: XML Schema für Bücher

```
<schema>
        <element name="buecher">
          <complexType>
           <sequence>
            <element name="buch" minOccurs="0" maxOccurs="unbounded" type="buchTyp"/>
           </sequence>
          </complexType>
        </element>
        <complexType name="buchTyp">
         <sequence>
          <element name="titel" type="string"/>
          <element name="autor" minOccurs="0" maxOccurs="unbounded" type="string"/>
          <element name="preis" type="int"/>
          <attribute name="isbn" use="required"/>
         </sequence>
        </complexType>
</schema>
```

Die statische Typüberprüfung von XML-Objekten und Operationen in **XOBE**$_{\text{DBPL}}$ besteht aus drei Hauptbestandteilen, einer XML-Sprachformalisierung, der Typinferenz und dem Subtypchecking. Im folgenden Abschnitt wird diese erste Version des Buchladens erweitert, um eine einheitliche Persistenz für XML-Objekte und reine Java-Objekte vorzustellen.

4 Persistenzkonzept

Bis jetzt sind XML-Objekte und Java-Objekte lediglich transient, das bedeutet, daß diese Objekte und ihre Daten nach Beendigung ihrer Anwendung verloren gehen. Im Gegensatz zu vielen anderen Ansätzen und Frameworks, die typabhängige und intransparente Persistenz anbieten, ermöglicht **XOBE**$_{\text{DBPL}}$ typunabhängige und transparente Persistenz mittels einer persistenten Umgebung, die `database` genannt wird. Zusätzlich zu Java-Klassendeklarationen, gibt es in **XOBE**$_{\text{DBPL}}$ Datenbankdeklarationen, um Typen innerhalb einer persistenten Umgebung zu definieren. In einer Datenbankdeklaration wird das Schlüsselwort **class** durch das neue Schlüsselwort **database** ersetzt. Daraufhin werden alle Membervariablen eines solchen Datenbankobjekts persistent durch Erreichbarkeit, und zwar **unabhängig vom Typ**. In Listing 1 wird der Buchladen als Klasse deklariert, deren Objekte transient sind. Indem der Buchladen nun innerhalb einer persistenten Umgebung definiert wird, werden seine Objekte persistent abgelegt. Aus Sicht des Programmierers genügt es also, den Buchladen als `database` zu deklarieren, alles weitere geschieht automatisch. Konstruktoren generieren neue persistente Objekte und Methodenaufrufe auf diesen Objekten verändern sie persistent. Listing 3 zeigt die persistente Version des Buchladens.

Listing 3: Persistente Version des Buchladens

```
1   ximport Buecher.xsd;
2   public database Buchladen implements Shop{
3    //Deklaration einer XML Membervariablen
4    xml<buecher> shop_buecher;
5    //Deklaration von Java Membervariablen
6    Person besitzer;
7    List bestellungen;
8
9    public Buchladen(Person besitzer){
10    this.besitzer = besitzer;
11    this.bestellungen = new XobeList();
12    ...
13   }
14   ...
15   //alle Methodendeklarationen
16   //bleiben unveraendert
17  }
```

Listing 4: Suche und Zugriff auf bereits existierende persistente Buchladen-Objekte

```
1   public class MainSuche{
2    ...
3    private List sucheBuchlaeden(
4       Person besitzer){
5     List shops =
6       $Buchladen[besitzer={besitzer}]$;
7     return shops;
8    }
9   }
```

Um die typunabhängige Persistenz zu demonstrieren, sind noch zwei weitere Membervariablen aufgeführt. Besitzer und Bestellungen sind mittels Java Klassen definiert. Es können zur Erzeugung persistenter Objekte beliebige Konstruktoren definiert werden. In Listing 3 ist ein Konstruktor definiert, dem ein Besitzer übergeben wird und der die Bestellungen mit einer leeren Liste initialisiert. In einer Anwendungsklasse macht es für den Programmierer keinen Unterschied, ob er transiente Objekte (z.B. `new Person(...)`) oder persistente Objekte (z.B. `new Buchladen(...)`) erzeugt. Ein weiterer wichtiger Punkt ist, daß im Laufe eines Programms ein anfangs transient erzeugtes Objekt mittels Zuweisung oder Parameterübergabe an eine Variable in einer persistenten Umgebung übergeben werden kann. Hier wird das transiente Objekt automatisch in eine persistente Objekt-Variante umgewandelt. In **XOBE**$_{\text{DBPL}}$ werden XPath-Ausdrücke zum Suchen verwendet, auch im Zusammenhang mit der Suche nach bereits existierenden persistenten Objekten. In diesem Fall ist der Kontext des XPath-Ausdrucks ein Datenbankname. Listing 4 zeigt die Suche nach Buchladen-Objekten anhand eines gegebenen Besitzers.

Das grundlegende Konzept, um diese typunabhängige und transparente Persistenz in **XOBE**$_{\text{DBPL}}$ zu verwirklichen, zeigt Abbildung 1. Jede Klassendeklaration in **XOBE**$_{\text{DBPL}}$ wird automatisch in eine tran-

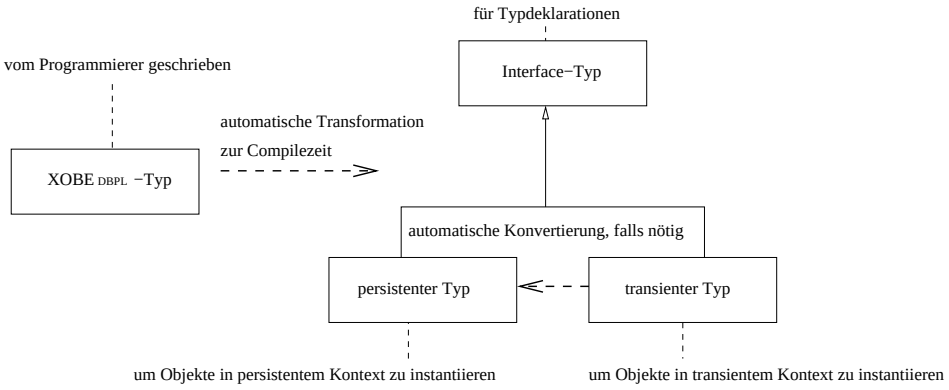


Abbildung 1: Grundlegendes Konzept zur Realisierung der transparenten typunabhängigen Persistenz

siente und eine persistente Variante transformiert, die beide ein gemeinsames Interface implementieren. Die Interface-Variante wird im gesamten Code für jegliche Typdeklarationen verwendet, die transiente Variante, um Objekte in transienten Umgebungen zu instantiieren und die persistente, um analog Objekte in persistenten Umgebungen zu instantiieren. Wie bereits erwähnt, kann es nötig sein, zu einem späteren Zeitpunkt ein transientes in ein persistentes Objekt umzuwandeln. All das detektiert der Precompiler automatisch. **XOBE**$_{\text{DBPL}}$ Laufzeitumgebungen müssen natürlich wissen, *wo* persistente Daten abgelegt werden sollen. Dieses *wo* hängt von der Konfiguration des **XOBE**$_{\text{DBPL}}$ lokalen Servers ab, bzw. von einer

optionalen Konfigurationsdatei.

5 Transaktionskonzept

In **XOBE**$_{\text{DBPL}}$ können persistente Daten zwischen verschiedenen Applikationen ausgetauscht werden. Um einen konsistenten Zugriff zu gewährleisten, ist ein Transaktionskonzept erforderlich. Javas Syntax wird daher um ein Transaktions-Statement analog zum bereits existierenden Synchronisations-Statement erweitert. Das Transaktions-Statement beginnt mit dem neuen Schlüsselwort **transaction** und erwartet eine Liste von Variablennamen. Diese Variablen müssen als Datenbanktypen deklariert worden sein. Der Block des Transaktions-Statement wird dann als `eine` Transaktion für die aufgelisteten Datenbank-Objekte ausgeführt. Einzelne Methodenaufrufe auf Datenbank-Objekten ausserhalb einer solchen Umgebung werden automatisch als eine Transaktion pro Methodenaufruf ausgeführt. Transaktions-Statements werden zur Compilezeit automatisch in Kommunikationsanweisungen mit dem **XOBE**$_{\text{DBPL}}$ lokalen Server übersetzt, dieser ist für die korrekte Durchführung z.B. das Locking verantwortlich.

Listing 5: Eine Transaktion auf einem persistenten Buchladen

```
...
Buchladen shop;
...
transaction(shop){
 if(shop.buchRegistriert(isbn)){
  shop.deregistriereBuch(isbn);
 }
 shop.registriereBuch(titel,autoren,isbn,preis);
}
...
```

Deadlocks können in **XOBE**$_{\text{DBPL}}$ ausgeschlossen werden, da alle zu sperrenden Datenbank-Objekte im Vorhinein durch die anzugebende Parameterliste bekannt sind. Listing 5 zeigt ein Transaktions-Beispiel in **XOBE**$_{\text{DBPL}}$. Nur eine Transaktionsumgebung kann hier garantieren, daß ein Buch in einer persistenten Buchhandlung nicht doppelt registriert wird.

6 Implementierung

Der Precompiler. Abbildung 2 zeigt den Precompiler, der **XOBE**$_{\text{DBPL}}$ Programme in reinen Java-Code transformiert. Der Programm-Parser ist mittels des Java Compiler Compilers (JavaCC) [14] gebaut. Um XML Schemata und DTDs zu parsen, verwenden wir den Xerces-Parser [2]. Intern werden Programme mit Hilfe des Java Tree Builders (JTB) [12] repräsentiert. Innerhalb des Compilers und damit zur Compilezeit findet der XML-Typechecking-Prozess statt. Die anschliessende Transformation besteht aus zwei Schritten. Im ersten werden die XML spezifischen Erweiterungen transformiert und im zweiten findet die Transformation der Typstruktur statt, deren Grundkonzept bereits in Abbildung 1 erläutert worden ist. Nach dem Transformations-Prozess werden XML-Objekte mittels eines DOM-ähnlichen Modell dargestellt.

Die Laufzeitumgebung. Abbildung 3 zeigt die Laufzeitumgebung für **XOBE**$_{\text{DBPL}}$ Programme. Transformierte Programme kommunizieren mit einem **XOBE**$_{\text{DBPL}}$ lokalen Server. Dieser ist insbesondere für die Datenkonsistenz zuständig. Lokale Daten werden in einer lokalen Datenbank gespeichert, während geteilte Daten mittels SOAP [13] an entfernte **XOBE**$_{\text{DBPL}}$ Server weitergeleitet bzw. angefragt werden.

7 Ausblick

XOBE$_{\text{DBPL}}$ bietet Datenbankprogrammierung auf einem sehr abstrakten Level. Java ist durch XML-Objekte, XPath-Anfragen und Update-Ausdrücke sowohl syntaktisch als auch semantisch erweitert worden. Desweiteren realisiert **XOBE**$_{\text{DBPL}}$ typunabhängige und transparente Persistenz, sowie ein Transaktionskonzept für den Datenaustausch mit anderen Applikationen.

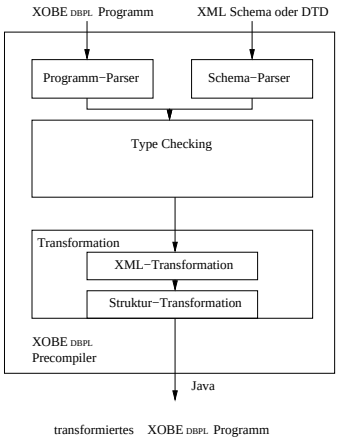


Abbildung 2: Der **XOBE**$_{\text{DBPL}}$ Precompiler

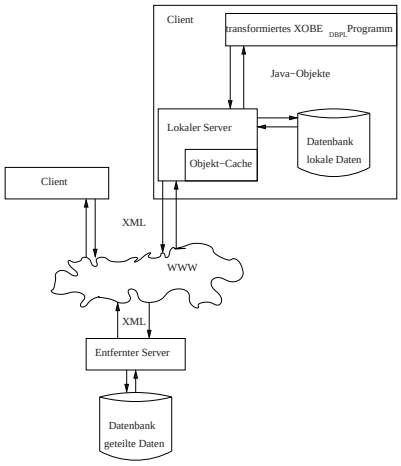


Abbildung 3: Die Laufzeitumgebung für **XOBE**$_{\text{DBPL}}$ Programme

In Zukunft soll die Laufzeitumgebung von **XOBE**$_{\text{DBPL}}$ weiter verbessert werden. Darüberhinaus planen wir, zur Überprüfung Wert-basierter Bedingungen dynamisches Typechecking für XML-Objekte und Operationen anzubieten.

Literatur

[1] The Apache Xindice Project. Xindice. `http://xml.apache.org/xindice/index.html`, Januar 2004. Version 2.0.

[2] The Apache XML Project. Xerces Java Parser. `http://xml.apache.org/xerces-j/index.html`, 15. November 2001. Version 1.4.4.

[3] Josephine Cheng and Jane Xu. Xml and db2. In *Proceedings of the 16th IEEE International Conference on Data Engineering (ICDE)*, pages 569–576. IEEE, 2000.

[4] Infonyte GmbH. Infonyte DB. URL: `http://www.infonyte.com`, 2003.

[5] Martin Kempa and Volker Linnemann. Type Checking in XOBE. In Gerhard Weikum, Harald Schöning, and Erhard Rahm, editors, *Proceedings of Datenbanksysteme für Business, Technologie und Web (BTW), 10. GI-Fachtagung,*, volume P-26 of *Lecture Notes in Informatics*, pages 227–246. Gesellschaft für Informatik, 26.-28. Februar 2003.

[6] F. Matthes, G. Schröder, and J.W. Schmidt. Tycoon: A Scalable and Interoperable Persistent Environment. Fully Integrated Data Environments, ESPRIT Basic Research Series, pages 365–381, Heidelberg, 2000. Springer-Verlag.

[7] Ravi Murthy and Sandeepan Banerjee. XML Schemas in Oracle XML DB. In *Proceedings of the 29th VLDB Conference, Berlin, Germany*, pages 1009–1018, 2003.

[8] J.W. Schmidt and F. Matthes. The DBPL Project: Advances in Modular Database Programming. volume 19 of *Information Systems*, pages 121–140, 1994.

[9] Harald Schöning. Tamino - A DBMS designed for XML. In *Proceedings of the 17th International Conference on Data Engineering*, pages 149–154, Heidelberg, Germany, April 2-6 2001. IEEE Computer Society.

[10] Henrike Schuhart and Volker Linnemann. Valid Updates for Persistent XML Objects. In *Proceedings of Datenbanksysteme für Business, Technologie und Web (BTW), 11. GI-Fachtagung,*, Lecture Notes in Informatics. Gesellschaft für Informatik, 2.-4. March 2005.

[11] Henrike Schuhart, Dominik Pietzsch, and Volker Linnemann. Framework of the XOBE Database Programming Language. In *International Conference Applied Computing(IADIS)*, 21.-25. Februar 2005.

[12] Kevin Tao, Wanjun Wang, and Dr. Jens Palsberg. Java Tree Builder JTB. `http://www.cs.purdue.edu/jtb/`, 15. May 2000. Version 1.2.2.

[13] W3Consortium. SOAP Version 1.2 Part 1: Messaging Framework. W3C Recommendation, `http://www.w3.org/TR/soap-12-part1/`, 24. June 2003.

[14] WebGain. Java Compiler Compiler (JavaCC) – The Java Parser Generator. `http://www.webgain.com/products/java_cc/`, 2002. Version 2.1.

Antwortmengenprogrammierung in der Praxis - eine Fallstudie

Petra Schwaiger

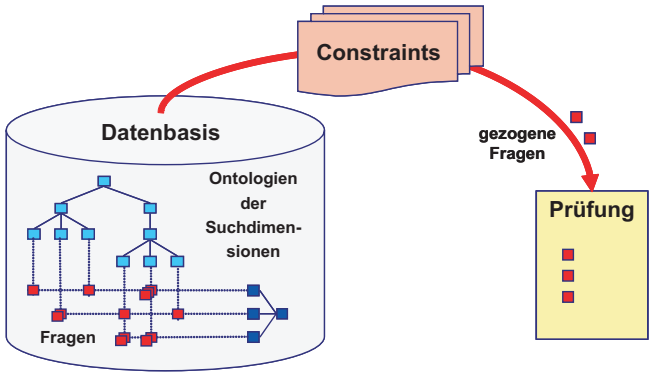
Universität Passau, Fakultät für Mathematik und Informatik
D-94030 Passau, Germany
schwaige@im.uni-passau.de

Zusammenfassung

Die Bedeutung des E-learning zeigt sich unter anderem in der zunehmenden Verwendung von Prüfungsmanagementsystemen, welche unter anderem zum Zwecke der Zusammenstellung von Tests eingesetzt werden. Das Problem der Auswahl von Fragen unter vorgegebenen Randbedingungen ist in der Praxis oftmals nur unzureichend gelöst. In dieser Arbeit stellen wir hiervon eine umfassende Lösung in Smodels vor.

1 Einleitung

In der industriellen Praxis gewinnen verschiedene Aspekte des E-learning an Bedeutung. So erfahren Aus- und Fortbildungsabteilungen von Unternehmen durch Prüfungsmanagementsysteme - wie zum Beispiel der IFIS Assessment.Suite, welche vom Berufsbildungswerk der Bausparkassen und von der Deutschen Bahn AG eingesetzt wird - zunehmende Unterstützung in der Zusammenstellung von Tests.



Bei der Prüfungsgenerierung wird aus einem vorhandenen Datenpool eine Auswahl von Fragen getroffen. In der Regel werden die Fragen durch Metadaten aus verschiedenen Suchdimensionen, wie z. B. Thema, Schwierigkeitsgrad, Frageart und Bearbeitungsdauer, beschrieben. Möglicherweise liegen einzelne Dimensionen als Ontologie, z. B. in Form einer Themenhierarchie, vor. Denkbar sind auch n:m-Assoziationen, was der Fall wäre, wenn eine Frage mehreren Themen zugeordnet ist. Typischerweise sind an die Zusammensetzung der Fragen Bedingungen (Constraints) geknüpft. Diese können in Form von Zahlenbeschränkungen repräsentiert sein. So könnte eine maximale Anzahl von Fragen zu einem bestimmten Thema oder eine Mindestbearbeitungsdauer gefordert sein.

In den industriellen Lösungen ist das Problem der dynamischen Generierung von Prüfungen aus einem vorhandenen Fragenpool unter vorgegebenen Randbedingungen meist nur mit Einschränkungen - und

damit nicht zufriedenstellend - modelliert bzw. gelöst. Je nach Lösungsansatz bereiten die Zahlenrestriktionen, die hierarchischen Strukturen und die damit verbundenen transitiven Anforderungen, die Mehrdimensionalität des Lösungsraums und n:m-Beziehungen - bzw. deren Kombination - Schwierigkeiten.

Smodels ([SN01], [NS00]) stellt gegenüber den normalen logischen Programmen unter anderem eine Erweiterung um Kardinalitäts- und Gewichtsconstraints dar. Mit dem Zugewinn an Semantik lässt sich gerade eben beschriebenes Problem relativ einfach und umfassend lösen.

Ziel unserer Arbeit ist die Präsentation der Lösung des in der Praxis interessanten und relevanten Problems der "Prüfungsfragengenerierung unter Constraints" unter Zuhilfenahme von Smodels.

2 Erläuterungen zur verwendeten Sprache

Die in Smodels verwendete deklarative Sprache ist dem Answer Set Programming ([Lif]) zuzurechnen.

2.1 Grundlagen des Answer Set Programming (ASP)

Ausgangspunkt für ASP-Systeme ist die Implementierung der stabilen Modellsemantik von normalen (logischen) Programmen ([GL88]) mit Regeln der Form $H \leftarrow A_1, ..., A_n, not\, B_1, ..., not\, B_m$ mit atomaren Formeln (Atomen) H, A_1, ..., A_n, B_1, ... B_m ($n, m \geq 0$). Als Literal bezeichnet man ein Atom A (positives Literal) oder dessen Negation $not\, A$ (negatives Literal). Unter not wird hier die Defaultnegation verstanden. In manchen Implementierungen ist zusätzlich zu dieser auch die klassische Negation ($\neg A$) erlaubt.

Typischerweise haben ASP-Programme mehrere stabile Modelle (Antwortmengen), welche die verschiedenen Lösungen des durch das Programm repräsentierten Problems darstellen.

2.2 Erweiterungen

Durch zusätzlich erlaubte Anzahlconstraint-, Gewichtsconstraint- und Bedingungsliterale lassen sich in Smodels Erweiterungen der oben beschriebenen Basisregeln - sogenannte allgemeine Gewichtsconstraint-Regeln - formulieren ([NS00, Syr03]). Diese haben die Gestalt $C_0 \leftarrow C_1, ..., C_n$ mit $n \geq 0$ und Gewichtsconstraints (oder als Sonderfall Basisliterale, Anzahlconstraints) $C_i, 0 \leq i \leq n$. Die zugehörige Semantik stellt eine Erweiterung der stabilen Modellsemantik dar.

Möchte man ausdrücken, dass aus den vorhandenen 4 Fragen "`frage(fi).`" ($1 \leq i \leq 4$) mindestens 1 und höchstens 3 Fragen auszuwählen sind (Prädikat `gezFrage`), so kann dies durch

```
1 {gezFrage(f1), gezFrage(f2), gezFrage(f3), gezFrage(f4)} 3.
```

geschehen. Unter Zuhilfenahme eines Bedingungsliterals (Bedingung: `frage(X)`) lässt sich dies noch wie folgt kürzer ausdrücken:

```
1 {gezFrage(X) : frage(X)} 3.
```

In analoger Weise können Bedingungen an die Summe der Gewichte der Literale (Default: 1) in einem Gewichtsconstraintliteral formuliert werden. Im Folgenden erfolgt nach der Zuordnung des Gewichts die "Ziehung" von Literalen mit dem Prädikat `gezFrageDauer` derart, dass die Summe der Gewichte der vorkommenden Literale einen Wert zwischen 5 und 13 einnimmt:

```
#weight gezFrageDauer(X,Y) = Y.
5 [gezFrageDauer(X,Y) : frageDauer(X,Y)] 13.
```

Um Variablen zulassen zu können und trotzdem die Entscheidbarkeit zu gewährleisten, sind in Smodels nur sogenannte ω-beschränkte Regeln erlaubt, was bedeutet, dass jede Variable der Regel in einem positiven Literal - genauer einem sogenannten positiven Domänenliteral (zur Einschränkung der möglichen Werte der Variablen) - des Rumpfes vorkommen muss.

3 Lösungsansätze in Smodels

3.1 Problemausprägung 1: Eine hierarchische Suchdimension

Die in dem Beispiel-Datenpool vorhandenen 1000 Fragen sind nach Lernzielen (und indirekt Themen) klassifiziert. Genauer gilt, dass jede Frage genau einem Lernziel und jedes Lernziel genau einem Thema zugeordnet ist. An eine Prüfung mit einer vorgegebenen Anzahl von n ($1 \leq n \leq 1000$) Fragen ist die Bedingung geknüpft, dass sie möglichst viele verschiedene Themen und Lernziele abdecken soll.

Durch Fakten wie

```
thema(thA). thema(thB). ...
lernziel(lza). lernziel(lzb). ...
frage(f1). frage(f2). ...
themaLernziel(thA,lza). themaLernziel(thA,lzb). ...
lernzielFrage(lza,f1). lernzielFrage(lza,f2). ...
```

lässt sich die Datenbasis zusammen mit der hierarchischen Struktur von Thema - Lernziel - Frage modellieren. Die indirekte Zuordnung der Fragen zu den Themen erhält man mit der Regel

```
themaFrage(X,Y) :- themaLernziel(X,Z), lernzielFrage(Z,Y).
```

Die Anzahl der zu ziehenden Fragen ist dem Programm als Konstante `n` zu übergeben und kann dort durch das Fakt "`anzGezFragen(n).`" "verfügbar" gemacht werden. Durch die Regel

```
Y {gezFrage(X) : frage(X)} Y :- anzGezFragen(Y).
```

wird das Ziehen der Fragen repräsentiert. Die gezogenen Lernziele und Themen können anhand von

```
gezLernziel(X) :- lernzielFrage(X,Y), gezFrage(Y).
gezThema(X) :- themaLernziel(X,Y), gezLernziel(Y).
```

bestimmt werden. Schließlich ist die Bedingung, dass möglichst viele verschiedene Themen in der Ziehung vertreten sein - also "gezogen" werden - müssen, auszudrücken. Sind mehr Themen als zu ziehende Fragen vorhanden, so stimmt die Anzahl der zu ziehenden Themen mit der Anzahl der zu ziehenden Fragen überein. Im anderen Fall müssen aufgrund des Mangels an Auswahl alle Themen gezogen werden. In einem ersten Schritt wird die Anzahl der vorhandenen Themen gezählt:

```
anz(0..1000).
anzThemen(Y) :- Y {thema(X) : thema(X)} Y, anz(Y).
```

Hierbei wird das Prädikat `anz` aufgrund der geforderten ω-Beschränktheit benötigt; "`anz(0..1000).`" ist eine Abkürzung für die 1001 Regeln "`anz(i).`" ($0 \leq i \leq 1000$). In einem zweiten Schritt kann durch

```
anzGezThemen(X) :- anzGezFragen(X),anzThemen(Y),anz(Y), X <= Y.
anzGezThemen(Y) :- anzGezFragen(X),anzThemen(Y),anz(Y), X > Y.
```

die Anzahl der zu ziehenden Themen bestimmt werden. Im letzten Schritt erfolgt die Formulierung der Zahleneinschränkung:

```
:- {gezThema(X) : thema(X)} Y-1, anzGezThemen(Y).
:- Y+1 {gezThema(X) : thema(X)}, anzGezThemen(Y).
```

Der leere Regelkopf deutet $\perp$ oder $false$ an, was heißt, dass in einem Modell der Regelrumpf nicht erfüllt sein darf. Analog ist die entsprechende Bedingung an die zu ziehenden Lernziele zu modellieren.

3.2 Problemausprägung 2: Zwei Suchdimensionen

Das Beispiel 3.1 wird dahingehend erweitert, dass die Fragen auch in ihrer Fragenart (Single Choice oder Pick and Place) unterschieden werden. Als Randbedingung ist eine Mindestanzahl von Single-Choice-Fragen gefordert.

Die Datenbasis ist durch Fakten wie "`frageArt(f1,sc). frageArt(f2,pp). ...`" erweitert. Analog zu `n` wird auch die Anzahl der zu ziehenden Single Choice Fragen `n_sc` dem System übergeben und durch "`anzGezFragenSc(n_sc).`" repräsentiert. Schließlich sind nur noch die Regeln

```
gezFrageArt(X,Y) :- gezFrage(X), frageArt(X,Y).
:- {gezFrageArt(X,sc) : frage(X)} Y-1, anzGezFragenSc(Y).
```

anzufügen. Die Mehrdimensionalität des Suchraumes stellt insofern keine Schwierigkeit an die Modellierung dar, da die Bedingungen an die zweite Dimension (Frageart) analog zu denen der ersten und vollig unabhängig von diesen formuliert werden können. Dies verdient insofern eine Erwähnung, da manche "Ziehungsalgorithmen" diese Unabhängigkeit nicht gewährleisten und somit den Lösungsraum ungewollt einschränken.

3.3 Problemausprägung 3: n:m-Assoziation

Nehmen wir an, dass die Fragen in unserem Beispiel-Datenpool ausschließlich und direkt nach drei verschiedenen Themen (thA, thB, thC) klassifiziert sind. Die Beziehung Frage - Thema ist vom Typ n:m, so dass eine Frage mehreren Themen zugeordnet sein kann. Eine Prüfung mit einer vorgegebenen Anzahl von n ($1 \leq n \leq 1000$) Fragen soll $n\_thA \leq n$ Fragen zu Thema A und $n\_thB \leq n$ Fragen zu Thema B enthalten.

Die Datenbasis und die Ziehung kann analog zu obigen Beispielen modelliert werden. Problem hierbei ist, dass eine gezogene Frage nicht mehrfach gezählt werden darf. Wird beispielsweise die Frage $f1$, welche den Themen thA und thB zugeordnet ist, gezogen, so darf sie bei der Zählung der gezogenen Fragen in den einzelnen Themenbereichen nur für eines der beiden Themen gewertet werden. Dies erreicht man dadurch, dass "während" des Fragen-Ziehens eine Entscheidung für ein mögliches zulässiges Thema vorgenommen wird. Dies kann durch

```
gezA(X) :- gezFrage(X),frageThema(X,thA),not gezB(X),not gezC(X).
gezB(X) :- gezFrage(X),frageThema(X,thB),not gezA(X),not gezC(X).
gezC(X) :- gezFrage(X),frageThema(X,thC),not gezA(X),not gezB(X).
```

oder alternativ durch

```
1{gezFrageGezThema(X,Y):frageThema(X,Y)}1 :- gezFrage(X),frage(X).
```

nachgebildet werden. Die Constraints bezogen auf die Anzahl der Fragen in den einzelnen Themen sind in entsprechender Weise zu formulieren.

3.4 Problemausprägung 4: Bedingung nicht über Fragenanzahl, sondern einer anderen numerischen Eigenschaft

Dieser Fall wird hier nur kurz skizziert: Im Gegensatz zu obigen Beispielen sind die Fragen mit einer Bearbeitungsdauer ("`frageDauer(X,Y).`") versehen und die Prüfung durch eine vorgegebene Gesamtbearbeitungsdauer ("`dauer_ges(t).`") bestimmt. Problem hierbei ist, dass nicht von vornherein die Anzahl der zu ziehenden Fragen klar ist. Eine Lösung lässt sich dadurch realisieren, dass die entsprechenden Fragen-Prädikate mit der entsprechenden Fragedauer gewichtet werden und die Ziehung über die Summe der Gewichte erfolgt, wie im Folgenden angedeutet ist:

```
#weight gezFrageDauer(X,Y) = Y.
Z [gezFrageDauer(X,Y) : frageDauer(X,Y)] Z :- dauer_ges(Z).
```

4 Diskussion

Obige kurze Liste von Problemausprägungen lässt sich sinnvoll und problemlos erweitern. So kann man zum Beispiel durch eine geeignete Modellierung erreichen, dass, falls inkonsistente Anforderungen an die zu generierende Prüfung gestellt werden, ein entsprechender Hinweis mit möglicher Ursache vom System ausgegeben wird. Smodels scheint ein geeignetes Mittel zu sein, vielerlei Anforderungen, welche im Rahmen der "Generierung von Prüfungen unter Constraints" auftreten, zu modellieren. Im Sinne der Ziele der Working Group on Answer Set Programming (WASP) ist hiermit also ein in der Industrie auftretender Anwendungsfall gefunden, welcher den Nutzen der ASP-Technologie zeigt.

Allerdings erinnern natürlicherweise manche gelöste (Teil-)Probleme an Aufgaben, die auch im Constraint-Programming (vgl. auch [Nie99]) oder im Operations Research gelöst werden. Eine genauere Abgrenzung zu diesen Bereichen gehört zu den nächsten Aufgaben. Desweiteren soll eine Verbindung von ontologischem Reasoning - die einzelnen Suchdimensionen sind oftmals als Ontologien gegeben - und Smodels untersucht werden.

Testläufe haben ergeben, dass der gezeigte Ansatz mit realen Daten sinnvolle Ergebnisse liefert. Eine genauere Untersuchung bzgl. Effizienz und Skalierbarkeit ist jedoch noch erforderlich.

5 Danksagung

Besonderer Dank gilt Prof. Dr. Burkhard Freitag für die wertvollen Anregungen und Ratschläge im Rahmen der Betreuung der Forschungstätigkeit.

Für die hilfreichen Erläuterungen und Diskussionen zu den Anforderungen eines Prüfungsmanagementsystems und die Bereitstellung einer realen Datenbasis möchte ich meinen Kollegen im Institut für Informationssysteme und Softwaretechnik (IFIS, http://www.ifis.uni-passau.de/), insbesondere Dr. Ulrich Zukowski, danken.

Literatur

[GL88] Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In Robert A. Kowalski and Kenneth Bowen, editors, *Proceedings of the Fifth International Conference on Logic Programming*, pages 1070–1080, Cambridge, Massachusetts, 1988. The MIT Press.

[Lif] Vladimir Lifschitz. Introduction to answer set programming, unpublished draft.

[Nie99] Ilkka Niemelä. Logic programming with stable model semantics as a constraint programming paradigm. *Annals of Mathematics and Artificial Intelligence*, 25(3,4):241–273, 1999.

[NS00] Ilkka Niemelä and Patrik Simons. Extending the Smodels system with cardinality and weight constraints. In Jack Minker, editor, *Logic-Based Artificial Intelligence*, chapter 21, pages 491–521. Kluwer Academic Publishers, 2000.

[SN01] Tommi Syrjänen and Ilkka Niemelä. The smodels system. In *Proceedings of the 6th International Conference on Logic Programming and Nonmonotonic Reasoning*, Vienna, Austria, September 2001. Springer-Verlag.

[Syr03] Tommi Syrjänen. Logic programming with cardinality constraints. Research Report A86, Helsinki University of Technology, Laboratory for Theoretical Computer Science, Espoo, Finland, December 2003.

Robuste Systemevolution durch automatische Unterstützung

Boris Stumm
stumm@informatik.uni-kl.de
Technische Universität Kaiserslautern

Zusammenfassung

Unternehmen integrieren ihre Systeme immer stärker miteinander, um den größtmöglichen Nutzen aus den Informationen zu ziehen. Resultat sind hochkomplexe, verteilte Informationssysteme mit einer Vielzahl von Abhängigkeiten zwischen den beteiligten Einzelsystemen.

Informationslandschaften unterliegen ständig verschiedenen Formen von Veränderung durch Umstrukturierungen, Neuanschaffungen usw. Während das bei einzelnen Systemen unproblematisch ist, können sich lokale Änderungen in einer integrierten Informationslandschaft auf andere Systeme auswirken. Das reicht vom Totalausfall über Datenkorruption bis hin zu subtilen Fehlern, die möglicherweise erst viel später entdeckt werden.

Diese negativen Auswirkungen müssen daher auf ein Minimum (sowohl an Zahl als auch an Dauer) reduziert werden. Das ist auf organisatorischer Ebene alleine (z. B. neue Geschäftsprozesse zur Koordination der Verantwortlichen) nicht zu bewältigen. In dieser Arbeit stellen wir einen Ansatz vor, der Entwickler und Systemverwalter einerseits bei Planung und Durchführung von Änderungen unterstützt, und andererseits das Gesamtsystem robust macht gegen ungeplante lokale Systemänderungen.

1 Einleitung

Die heutige Systemlandschaft ist geprägt von zunehmender Komplexität und Heterogenität. Durch Neuanschaffungen und Firmenzusammenschlüsse kommen immer neue Systeme hinzu. Verschiedene Systeme werden integriert, um die vorhandenen Informationen besser verarbeiten zu können. In einer solchen heterogenen Informationslandschaft werden einzelne Systeme immer wieder verändert und weiterentwickelt, sei es, um Fehler zu beheben, oder um neuen Bedürfnissen gerecht zu werden.

Diesen nicht zentral gesteuerten Prozess nennen wir *Systemevolution*. In der Regel sind nur ein einzelnes oder einige wenige Systeme von einem Evolutionsschritt betroffen; aufgrund von Interdependenzen zwischen den Systemen kann sich eine Änderung jedoch auf weitere Systeme auswirken. Bei einer fehlenden Organisation dieses Prozesses werden die Auswirkungen möglicherweise erst viel später erkannt, was die Problemursachen schwer nachvollziehbar macht. Mit einer *Änderung* bezeichnen wir prinzipiell alle Vorgänge in einem System, die sich auf andere Systeme funktional auswirken. Das betrifft Schemaänderungen (Schemaevolution), Konfigurationseinstellungen, Netzwerkverbindung, Qualitätszusagen usw.

Diese Problematik verschärft sich mit der zunehmenden Komplexität und Heterogenität von integrierten Informationssystemen, deshalb wird eine automatisierte Unterstützung des Evolutionsprozesses erforderlich. Wir möchten in dieser Arbeit einen Überblick über unseren Ansatz geben. Er soll ein Integrationssystem robust machen gegen Probleme, die durch lokale Änderungen hervorgerufen werden. Dazu definieren wir einen Prozess und stellen automatisierte Analysewerkzeuge bereit, die auch bei nicht optimalem Prozessablauf Probleme erkennen können.

Die Arbeit ist wie folgt aufgebaut. In Abschnitt 2 grenzen wir die Probleme der Systemevolution in einer heterogenen Umgebung ein und stellen Soll- und Ist-Zustand in verteilten Informationsystemen gegenüber. Unser Ansatz, mit dem wir die Probleme angehen wollen, wird in Abschnitt 3 näher ausgeführt. Eine Abgrenzung zu verwandten und komplementären Arbeiten sowie ein Fazit geben wir in Abschnitt 4.

2 Problematik

Zu den Änderungen, die ein Informationssystem während seiner Lebensdauer erfährt, gehören alle Formen der Schemaevolution und Änderungen der technischen Konfiguration des Systems (Hardware und Software). Die Auswirkungen sind unterschiedlicher Art:

- Bei der „klassischen Schemaevolution“ werden Schemaelemente hinzugefügt, geändert oder gelöscht.
- Reparaturen und Verbesserungen an Systemen sind eigentlich auch der Schemaevolution zuzurechnen. Wir führen sie hier gesondert auf, weil es normalerweise nur „kleine“ Änderungen sind (z. B. Erweiterung eines Datentyps von char(20) auf char(30) oder Änderung des Verhaltens einer Funktion bei Eingaben außerhalb des Definitionsbereichs). Während bei größeren Schema-Umbauten davon ausgegangen werden kann, dass alle betroffenen Systeme mit einbezogen werden, ist das hier nicht unbedingt der Fall.
- Auch Änderungen an der Konfiguration von Systemen sind von Bedeutung. Dies umfasst die Vergabe von Benutzerrechten, periodisch wiederkehrende Abläufe (z.B. Data-Warehouse-Updates), Softwareupdates, Bandbreite der Netzwerkanbindung usw.

Bei eng gekoppelten Systemen, z.B. Anwendung und zugehöriges DBVS oder Data Warehouse und OLAP-System, kann man in der Regel davon ausgehen, dass bei Änderungen alle betroffenen Systeme gleichzeitig betrachtet werden und die Abhängigkeiten klar sind. Im größeren Rahmen einer verteilten Informationslandschaft sind Systeme aber nur noch lose gekoppelt, und nicht jeder Systemverantwortliche ist sich aller Abhängigkeiten bewusst. In einer solchen Konstellation wird eine automatische Verwaltung der Abhängigkeiten und Beziehungen der Informationsysteme zueinander nötig, um Probleme, die in abhängigen Systemen durch lokale Änderungen an einzelnen Systemen oder Systemgruppen entstehen, zu vermeiden oder zumindest frühzeitig zu erkennen. Diese Probleme nennen wir allgemein *Änderungsprobleme*.

Um Änderungsprobleme zu vermeiden, muss dafür gesorgt werden, dass vor der Durchführung von *lokalen* Änderungen die Auswirkungen auf *globaler* Ebene analysiert werden und eventuell auftretende Konflikte zuerst aufgelöst werden.

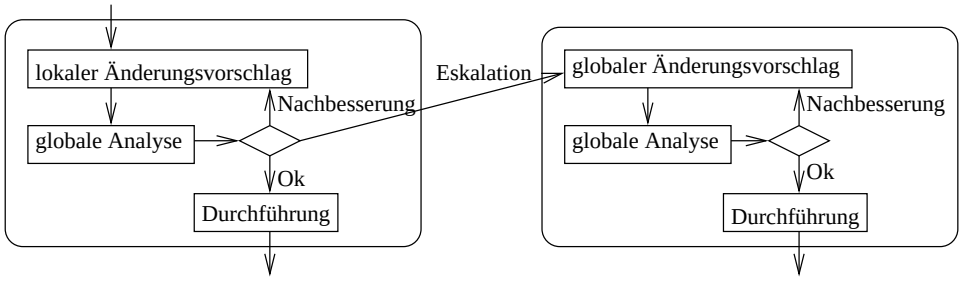


Abbildung 1: Prozess der Systemevolution

Abbildung 1 veranschaulicht diesen Prozess. Er lässt sich in einen lokalen und einen globalen Teil aufteilen, die beide ähnlich ablaufen. Bevor Änderungen an einem System vorgenommen werden wird der Änderungsvorschlag analysiert. Bei Problemen kann nachgebessert oder der Prozess auf alle betroffenen Systeme ausgeweitet werden. Der Kreislauf von Änderungsvorschlag, Analyse und Nachbesserung wiederholt sich dann auf der globalen Ebene, bis alle Konflikte beseitigt sind. Am Ende steht die Durchführung der Änderungen. Die Abfolge soll an einem kleinen Beispielszenario verdeutlicht werden.

Wir betrachten ein föderiertes DBVS (FDBVS), das eine Personendatenbank verwaltet. Eines der Quellsysteme speichert Name und Vorname von Personen konkateniert in der Spalte NAME der Tabelle PERSON. Im Quellsystem soll diese Spalte aufgeteilt werden in VORNAME und NACHNAME. Dieser Vorschlag wird analysiert, und es wird festgestellt, dass ein Problem auftritt, da das FDBVS auf die

Namensspalte zugreift. Es bietet sich jetzt die Möglichkeit, im Nachbesserungsschritt die Spalte NAME redundant im Schema zu belassen, oder durch die Verwendung von Sichten abwärtskompatibel zu bleiben. Damit sind die Konflikte aufgelöst, und die Änderung kann vollzogen werden. Wenn nicht nachgebessert wird, muss ein globaler Änderungsvorschlag eingebracht werden, der auch das FDBVS in die Änderungen mit einbezieht. Dadurch können weitere Systeme beeinflusst werden, weshalb eine iterative Ausweitung der in die Planung einbezogenen Systeme möglich ist.

Dieser Prozess funktioniert nur dann zufriedenstellend, wenn *alle* Änderungen wie beschrieben analysiert werden. Aufgrund der normalerweise relativ autonomen Einzelsysteme und verteilten Verantwortlichkeiten kann das im allgemeinen nicht garantiert werden. Deshalb ist es uns wichtig, in unserem Ansatz einen geordneten Evolutionsprozess zwar zu unterstützen, aber keinesfalls vorauszusetzen. Damit ist die Robustheit des Gesamtsystems auch unter widrigen Bedingungen gewährleistet (siehe Abschnitt 3).

Je größer die Informationslandschaft wird, desto unwahrscheinlicher ist es auch, das alle Informationen zur automatischen Analyse vollständig vorliegen. Das hat verschiedene Gründe:

- Änderungsvorschläge sind nicht immer genau und exakt beschrieben, es können z.B. semantische Informationen fehlen. Beispiel: In einer Spalte NAME, die vorher Vor- und Nachnamen einer Person enthielt, ist nach dem Einfügen der Spalte VORNAME nur noch der Nachname gespeichert. In diesem trivialen Fall vermutlich unproblematisch, kann dies in größerem Rahmen jedoch zu Problemen führen.
- Die Abhängigkeiten zwischen verschiedenen Systemen sind nicht immer vollständig beschrieben, weil semantische Informationen fehlen oder die Abhängigkeiten lediglich prozedural beschrieben sind (z.B. ETL-Skripte).
- Um eine globale Analyse mit vertretbarem Aufwand durchführen zu können, müssen die Metadaten in einem homogenen Format vorliegen, wodurch ein Informationsverlust im allgemeinen nicht vermeidbar ist.

Unser Ziel ist es, auch mit unvollständiger Information noch nützliche Ergebnisse zu liefern. Prinzipiell lässt sich das mit pessimistischen Abschätzungen bei fehlender Information erreichen.

3 Ansatz

Die für uns relevanten Aufgaben im Systemevolutionsprozess bestehen in der Überwachung der Informationssysteme, der Analyse von Änderungsvorschlägen (bzw. vollzogenen Änderungen) und der Ergreifung von Maßnahmen im Konfliktfall. Dafür stellt unser Ansatz drei Komponenten bereit: Änderungsmanager, Metadaten-Repository (MDR) und Metadaten-Agenten (MDAs). Im Metadaten-Repository sind die zur Analyse benötigten Metadaten gespeichert. Der Änderungsmanager nutzt die dort gespeicherte Information, um Änderungsanalysen durchzuführen. Für jedes Informationssystem gibt es einen MDA, welcher die Vermittlerrolle zwischen Änderungsmanager, MDR und Systemen übernimmt. Die MDAs sind zuständig für die Synchronisierung der Daten im MDR und den betreuten Systemen, das Erkennen von Änderungen, und zur Maßnahmenergreifung bei Problemen. Abbildung 2 gibt einen Überblick über die Zusammenhänge zwischen den Komponenten.

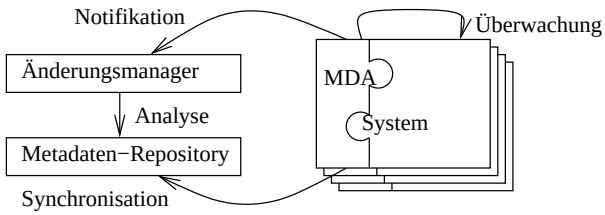


Abbildung 2: Architektur

Bevor eine Analyse stattfinden kann, müssen alle relevanten Metadaten in das MDR geladen werden (Synchronisation). In unserem Fall sind das unter anderem das föderierte Schema und die Quellschemas. Das geschieht teilautomatisiert mit Unterstützung durch die Metadatenagenten, welche proprietäres und MDR-Datenformat gleichermaßen verstehen. Bei einer *geplanten* Änderung wird zuerst beim Änderungsmanager ein Vorschlag eingereicht, welcher in der Regel gleich der neuen Systembeschreibung[1] ist. Der Änderungsmanager analysiert die Auswirkungen des Vorschlags auf andere Systeme (Analyse). Da es (vor allem, wenn mehrere Systeme beteiligt sind) im allgemeinen keine atomare Durchführung gibt, sind während der Durchführungsphase die betroffenen Systeme im MDR entsprechend markiert, und die Metadaten-Agenten können den Zugriff auf Systeme von außen einschränken[2]. Im Falle einer *ungeplanten* Änderung, wenn also im Quell-DBVS die Tabelle PERSON ohne Abstimmung mit dem FDBVS geändert wird, erkennt der zuständige Metadaten-Agent das, alarmiert die zuständigen Personen und den Änderungsmanager (Notifikation) und führt zusammen mit diesem ähnliche Aktionen aus wie auch bei einem geplanten Evolutionsschritt.

Ein wichtiger Punkt unseres Ansatzes ist das Metadatenmodell. Es muss prinzipiell beliebige Datenmodelle ausdrücken können, und auch mit unvollständigen Informationen zurechtkommen. Da wir uns auf die Analyse von möglichen Problemen beschränken ohne tatsächlich Schemaintegration zu betreiben, ist ein solches „universelles“ Datenmodell in unserem Fall durchaus realistisch. Prinzipiell gibt es in unserem Metamodell drei Ebenen. Auf Ebene 1 werden verschiedene *Aussagen* gemacht, die z.B. ein SQL-Schema modellieren. In Ebene 2 machen die Systeme einerseits *Zusagen*, z.B. zur Veröffentlichung ihres Schemas, und haben andererseits *Erwartungen* an Aussagen anderer Systeme. Auf der dritten und obersten Ebene schließlich werden Abhängigkeiten zwischen den Erwartungen und Zusagen modelliert sowie das Verhalten im Konfliktfall festgelegt. Wir prüfen derzeit die Verwendbarkeit von RDF [1] und das Konzept der Reifikation zur Modellierung und Verbindung der Ebenen. Aus Platzgründen kann hier jedoch nicht näher darauf eingegangen werden. Als Grundlage des Datenmodells kann evtl. CWM [2] Verwendung finden, welches in ein RDF-Format umgesetzt wird (z.B. [3]).

4 Abgrenzung und Fazit

Im Bereich der Informationsintegration gibt es viele Ansätze, die sich mit unterschiedlichen Facetten beschäftigen. Wir gehen bei uns davon aus, dass Dinge wie Schema-Matching [4, 5] und Schemaintegration [6] unserem Ansatz vorgeschaltet sind. Einige Arbeiten befassen sich mit ähnlichen Problemstellungen, wie z.B. View Maintenance [7] und View Synchronization [8, 9]. Das Kantana-System [10] definiert den Begriff der Mapping Adoption [11], und ist unserem Ansatz in der Zielsetzung ähnlich. Alle diese Arbeiten konzentrieren sich jedoch hauptsächlich auf die tatsächliche Anpassung der Systeme nach einer Änderung, was ihre Anwendbarkeit einschränkt. Unser Ansatz hingegen verfolgt das Ziel, ein integriertes Informationssystem ganzheitlich zu erfassen, Möglichkeiten zur prozessunterstützten Verwaltung zu geben und auftretende Probleme automatisch zu erkennen und zu analysieren. Auch beschränken wir uns nicht nur auf Schemainformationen, sondern lassen beliebige Metadaten zu. Der RADES-Ansatz [12] verfolgt ähnliche Ziele, setzt jedoch einen Schwerpunkt auf die dort zwingend erforderlichen Prozesse. Ein weiterer ähnlicher, aber beschränkterer Ansatz findet sich in [13]. Auch wenn sich teilweise Überlappungen zeigen, ist unser Ansatz im wesentlichen komplementär zu den vorgenannten. Wir sehen in unserem Ansatz einen Schritt in Richtung des Autonomic Computing [14] auf einer verteilten Ebene.

Unser Ansatz bietet die Infrastruktur für eine ganzheitliche Verwaltung eines integrierten Informationssystems. Geordnete Prozesse zur Systemevolution werden unterstützt, aber auch in der Abwesenheit dieser kann das System arbeiten. Kernpunkt ist die globale Analyse von lokalen Änderungen und die Fähigkeit, selbst bei unvollständigen Informationen nach dem Prinzip der „graceful degradation“ weiter zu funktionieren. Ein weiterer Schritt ist, sich nicht auf das automatische Erkennen von Änderungen

[1] Wir verwenden hier absichtlich nicht das Wort „Schemabeschreibung“, da die Schemainformationen nur einen Teil der gesamten Metadaten ausmachen.

[2] Hierzu ist möglicherweise nötig, Wrapper einzusetzen, durch die die Zugriffe geleitet werden.

und Änderungsproblemen sowie einigen defensiven Aktionen zu beschränken, sondern aktiv und automatisch Probleme aufzulösen. Es gibt hier im Moment noch viele offene Fragen, auch wenn die Basis der Infrastruktur und des Metamodells klar ist. Im Verbund mit anderen Werkzeugen und Systemen zur (automatischen) Informationsintegration nähert man sich jedoch weiter dem Idealbild eines sich selbst verwaltenden Systems, welches nur in „Notfällen“ noch menschliche Unterstützung benötigt.

Literatur

[1] World Wide Web Consortium (W3C). *Resource Description Framework (RDF): Concepts and Abstract Syntax*, February 2004. http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/.

[2] Object Management Group (OMG). *Common Warehouse Metamodel (CWM) Specification Version 1.1*, March 2003. http://www.omg.org/cwm/.

[3] Sergey Melnik. Representing UML in RDF, 2000. http://www-db.stanford.edu/ melnik/rdf/uml/.

[4] Sergey Melnik, Erhard Rahm, and Phillip A. Bernstein. A survey of approaches to automatic schema matching. *Journal of Web Semantics*, 1/1, 2004.

[5] Erhard Rahm and Phillip A. Bernstein. Developing Metadata-Intensive Appliacitons with Rondo. *VLDB Journal*, 10, 2001.

[6] C. Parent and S. Spaccapietra. Issues and approaches of database integration. *CACM 41/5*, 1998.

[7] S. Ceri and J. Widom. Deriving Production Rules for Incremental View Maintenance. *Proceedings of the 29th VLDB Conference, Barcelona, Spain*, 1991.

[8] X. Zhang and E. A. Rundensteiner. Data Warehouse Maintenance Under Concurrent Schema and Data Updates. Technical report, 1998.

[9] E. A. Rundensteiner, A. Koeller, X. Zhang, A. J. Lee, and A. Nica. Evolvable View Environment EVE: A Data Warehous System Handling Schema and Data Changes of Distributed Sources. *International Database Engineering and Application Symposium (IDEAS)*, April 1999.

[10] Periklis Andritsos, Ariel Fuxman, Anastasios Kementsietsidis, Renee J. Miller, and Yannis Velegrakis. Kanata: Adaptation and Evolution in Data Sharing Systems. *SIGMOD Record*, 33/4, December 2004.

[11] Yannis Velegrakis, Renee J. Miller, and Lucian Popa. Mapping Adoption under Evolving Schemas. *Proceedings of the 29th VLDB Conference, Berlin, Germany*, 2003.

[12] Clemens Dorda, Hans-Peter Steiert, and Jürgen Sellentin. Modellbasierter Ansatz zur Anwendungsintegration. *Information Technology*, 4/2004, August 2004.

[13] L. Deruelle, M. Bouneffa, G. Goncalves, and J. C. Nicolas. Local and Federated Database Schemas Evolution: An Impact Propagation Model. *LNCS: Proc. of the 10th International Conference on Database and Expert Systems Applications (DEXA'99), Florence, Italy*, September 1999.

[14] John J. Ritsko and Alfred G. Davis, editors. *IBM Systems Journal: Autonomic Computing*, volume 42/1. IBM, 2003. http://www.research.ibm.com/journal/sj42-1.html.

Anfragen an Ontologien in relationalen Datenbanken

Silke Trißl
Humboldt-Universität zu Berlin
Unter den Linden 6, 10099 Berlin
trissl@informatik.hu-berlin.de

Abstract

Ontologien und Taxonomien sind kontrollierte, strukturierte Vokabulare für eine Domäne um Objekte darin zu beschreiben. Ontologien in der Biologie, wie beispielsweise die NCBI Taxonomie, die Klassifizierung von Lebewesen, oder die Gene Ontology sind als Bäume bzw. gerichtete, azyklische Graphen aufgebaut. Taxonomien werden häufig zusammen mit den annotierten Objekten in relationalen Datenbanken gespeichert.

Durch die Struktur der Ontologien bzw. Taxonomien können Beziehungen zwischen Objekten angefragt und erkannt werden. Diese Anfragen innerhalb eines relationalen Datenbanksystems auszuführen setzt entweder die Verwendung von rekursiven Funktionen oder die Indizierung des Graphen voraus. Ich stelle 2 verschiedene Indizierungsmöglichkeiten vor, die transitive Hülle und Pre- und Postorder Ranking, und vergleiche die Anfragezeiten für beide Indexstrukturen mit Anfragezeiten für die rekursive Funktion.

1 Einleitung und Motivation

Ontologien und Taxonomien spielen eine wichtige Rolle in der Biologie und Medizin. Die vielleicht älteste Taxonomie in der Biologie ist die Klassifizierung von Flora und Fauna. In ihr spiegelt sich die evolutionäre Abstammung von verschiedenen Organismen wieder. Die NCBI Taxonomy ist als Baum mit gerichteten Kanten aufgebaut [1]. In den letzten Jahren wurden weitere Ontologien entwickelt um biologische Objekte und Vorgänge zu beschreiben, beispielsweise die Gene Ontology [2]. Sie stellt ein strukturiertes Vokabular zur Verfügung, um Proteine in ihren Funktionen und Wirkorten zu beschreiben. Die Gene Ontology (GO) ist ein gerichteter, azyklischer Graph (directed acyclic graph, DAG) mit einem Wurzelknoten.

Betrachtet man folgende Fragestellung 'Zeige mir alle biologischen Proben, die laut NCBI Taxonomy Eukaryoten sind?', so erwartet ein Biologe nicht nur die Proben, die direkt mit dem Konzept 'Eukaryot' beschrieben sind, sondern auch Proben von Menschen, Mäusen, Pantoffeltierchen und weiteren Eukaryoten.

Diese Frage in einer relationalen Datenbank zu beantworten ist mit Standard-SQL nicht möglich, da nicht nur nach Kindern eines Knotens gesucht wird, sondern auch nach deren Nachfahren. Da diese beliebig weit vom Startknoten entfernt sein können, ist es nur möglich die Frage mit Hilfe einer rekursiven Funktion zu beantworten.

Ein großer Nachteil der rekursiven Funktion ist die Abhängigkeit der Antwortzeit von der Anzahl der traversierten Knoten. Je nach Größe des Ergebnissets kann dies einige Zeit dauern. Eine Alternative dazu ist einen vorberechneten Index in nahezu konstanter Zeit anzufragen. Indizes benötigen allerdings Speicherplatz und Zeit für die Vorberechnung, was viele Indexstrukturen für große Ontologien ungeeignet macht.

In dieser Arbeit präsentiere ich eine Indexstruktur für Bäume und gerichtete azyklische Graphen, die Anfragen in konstanter Zeit beantwortet, aber wesentlich weniger Speicherplatz für baumähnliche Graphen benötigt als die transitive Hülle.

2 Speichern und Anfragen von Ontologien

Datenmodell. Wir betrachten Ontologien die die Form von gerichteten Bäumen oder azyklischen Graphen haben. Ein Pfad in einer solchen Struktur ist eine Folge von Knoten, die durch gerichtete Kanten verbunden sind, wobei die Länge des Pfades die Anzahl an Knoten in dem Pfad ist. In einem Baum kann jeder Knoten auf genau einem Pfad von der Wurzel aus erreicht werden. In DAGs können Knoten mehr als einen Elternknoten haben, daher kann mehr als ein Pfad zwischen zwei Knoten existieren. In einem Baum bzw. gerichteten Graphen ohne Zyklen (DAG) sind alle Knoten w Nachfahren von v wenn ein Pfad zwischen v und w existiert. Alle Knoten w, die von v aus zu erreichen sind, bilden die Menge an Nachfahren.

Graphen werden häufig als Sammlung von Knoten und Kanten gespeichert. Die Tabelle `node` enthält als eindeutige Kennung das Attribut `node_name`, in der Tabelle `EDGE` werden die gerichteten Kanten des Graphen in Form von Paaren von Knoten gespeichert, `from_node` und `to_node`.

Rekursive Datenbankfunktion. Alle Nachfahren eines gegebenen Knotens v können durch eine Tiefensuche über einem Baum oder DAG gefunden werden. Algorithmus 1 sucht zuerst alle Kinder des Startknotens v und für jedes der Kinder ruft sich die Funktion mit dem Kindknoten als neuen Startknoten selbst wieder auf. Können keine weiteren Kinder in dem Subgraphen mehr gefunden werden, gibt die Funktion alle traversierten Knoten zurück. Der Aufruf der Funktion erfolgt mit `SELECT * FROM successorSet(`v`);`.

Algorithmus 1 Rekursiver Algorithmus um alle Nachfahren von v zu erhalten.

```
FUNCTION successorSet(v) RETURNS nachfahren
  FOR EACH kind IN SELECT to_node FROM EDGE WHERE from_node = v DO
    FOR EACH enkel IN SELECT nachfahre FROM successorSet(kind) DO
      nachfahren := nachfahren + enkel;
    nachfahren := nachfahren + kind;
  RETURN nachfahren;
```

3 Indizierung von Baum- und DAG-Strukturen

In diesem Abschnitt erkläre ich Möglichkeiten die Beziehungen zwischen Knoten in Bäumen und DAGs vorzuberechnen. Die erste Möglichkeit ist die Berechnung der transitiven Hülle, während die zweite die Indizierung der Struktur mit Pre- und Postorder Ranks ist.

3.1 Berechnung der transitiven Hülle

Die transitive Hülle eines Graphen ist eine Menge von Knotenpaaren. Jedes Paar (v, w) wird in die transitive Hülle aufgenommen, wenn entweder eine Kante oder ein Pfad zwischen den Knoten v und w existiert.

In der Vergangenheit wurden verschiedene Algorithmen entwickelt um die transitive Hülle in relationalen Datenbanksystemen zu berechnen. Wir fanden heraus, dass der so genannte 'Logarithmic algorithm' [3] für Bäume und DAGs die geringste Zeit benötigt.

Dieser Algorithmus fügt zuerst alle Tupel der ursprünglichen Tabelle `EDGE` in die Tabelle für die transitive Hülle, `TC` mit der Distanz 1 zwischen zwei Knoten ein. In Schritt 2 werden alle Tupel mit maximaler Distanz mit `TC` gejoint. Die Bedingung für diese Operation ist, dass der Nachfolger der ersten Relation, $\texttt{TC}_1$, gleich dem Vorgänger der zweiten Relation, $\texttt{TC}_2$ sein muß. In der Tabelle `TC` wird $\texttt{TC}_1$`.VORG`, $\texttt{TC}_2$`.NACHF`, sowie `min(`$\texttt{TC}_1$`.DIST+`$\texttt{TC}_2$`.DIST)` eingefügt. Schritt 2 wird so lange wiederholt, bis keine weiteren Tupel mehr in `TC` eingefügt werden können.

Die transitive Hülle hat einen theoretischen Speicherplatzverbrauch von $O(|V|^2)$, doch der in Kapitel 4 gemessene Bedarf ist wesentlich geringer. Alle Nachfahren eines Knotens erhält man durch folgenden Ausdruck: `SELECT nachfahre FROM TC WHERE vorgaenger =` v`;`

3.2 Pre- und Postorder Ranking Modell

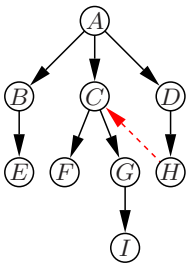
Das Pre- und Postorder Ranking Modell für Bäume ist gut untersucht. Verschiedene Gruppen haben dieses Modell vorgeschlagen, um XML Dokumente in relationalen Datenbanken zu indizieren [4].

Algorithmus 2 zeigt die Funktion, um Knoten in Bäumen als auch in DAGs Pre- und Postorder Ranks zuzuordnen. Die Knoten werden während einer Tiefensuche, angefangen am Wurzelknoten, markiert. Den Preorder Rank bekommt ein Knoten sobald er während der Tiefensuche gefunden wurde. Der Postorder Rank wird einem Knoten erst zugeteilt, wenn alle Nachfolger einen Postorder Rank haben, aber noch bevor ein Elternknoten in dem Pfad einen solchen besitzt.

Algorithmus 2 Pre- und Postorder Rank-Zurordnung, angefangen beim Wurzelknoten, r.

```
var pr:=0; var post:=0;
FUNCTION prePostOrder(r) RETURNS void
  FOR EACH kind IN SELECT to_node FROM EDGE WHERE from_node = r DO
    pr:=pr+1; pre:=pr;
    prePostOrder(kind);
    post:=post+1;
    s:= pr-pre;
    INSERT kind, pre, post, s INTO prePostOrder;
```

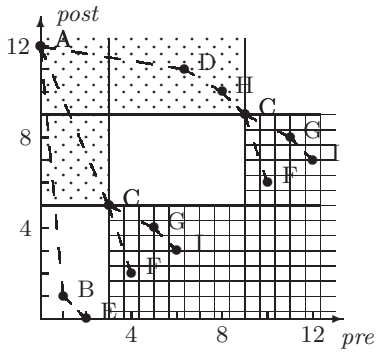
Wie in Abbildung 1(b) dargestellt, werden die Pre- und Postorder Ranks zusammen mit der eindeutigen Kennung des Knotens und der Anzahl an Kindern, `s`, in einer separaten Tabelle gespeichert. Für Bäume ist der Platzbedarf gleich der Anzahl an Knoten, in DAGs können jedem Knoten mehrere Ranks zugeordnet werden, abhängig von der Zahl an Wegen, auf denen dieser Knoten von der Wurzel aus erreichbar ist.



(a) DAG

ID	*pre*	*post*	*s*
A	0	12	12
B	1	1	1
C	3	5	3
D	7	11	5
E	2	0	0
F	4	2	1
G	5	4	1
H	8	10	4
I	6	3	0
C	9	9	3
F	10	6	0
G	11	8	1
I	12	7	0

(b) Tabelle mit den Pre-/ Postorder Ranks.



(c) Pre- /Postorder Ranks im Koordinatensystem. ⊞: Nachfolger von C, ∴: Vorgänger.

Abbildung 1: Pre- und Postorder Ranks in einem gerichteten azyklischen Graphen.

Der Nutzen von Pre- und Postorder Ranks wird deutlich, wenn wie in Abbildung 1(c) die Knoten mit den beiden Ranks in einem zweidimensionalen Koordinatensystem aufgetragen werden. Wie für Knoten C dargestellt, kann die Fläche in vier Regionen eingeteilt werden. Für Ontologien sind nur die Region oben links mit allen Vorfahren von C und die Region unten rechts mit allen Nachfolgern des Knotens C interessant.

Für alle Nachfolger eines Knotens v gilt, dass der Preorder Rank jedes Knotens w größer sein muß als der von v, während der Postorder Rank kleiner sein muß. Die Position der Nachfolger kann noch weiter eingeschränkt werden, denn alle Nachfahren von v besitzen einen Preorder Rank zwischen pre_v und $pre_v + s$. Beide Möglichkeiten sind in folgenden Anfragen dargestellt:

Option 1:

```
SELECT DISTINCT p1.node_name AS w
FROM prePostOrder p1, prePostOrder p2
WHERE p2.node_name = v
  AND p1.pre > p2.pre
  AND p1.post < p2.post;
```

Option 2:

```
SELECT DISTINCT p1.node_name AS w
FROM prePostOrder p1, prePostOrder p2
WHERE p2.node_name = v
  AND p1.pre > p2.pre
  AND p1.pre ≤ p2.pre + p2.s;
```

In Bäumen kommt jeder Knoten nur einmal vor, in DAGs kann ein Knoten jedoch mehrmals vorkommen. Wie man aber in Abbildung 1(c) sehen kann, ist die Menge an Nachfahren von jeder Instanz des Knotens C gleich, da Teilbäume von Knoten mit mehr als einem Elternknoten vervielfacht werden. Um nun jeden Knoten nur einmal in der Ergebnismenge zu erhalten ist `DISTINCT` in den Anfragen notwendig.

4 Ergebnisse

In diesem Abschnitt vergleiche ich die Indizierungsmethoden mit dem rekursiven Algorithmus. Ich stelle sowohl Ergebnisse von generierten Bäumen und DAGs als auch von existierenden Ontologien vor.

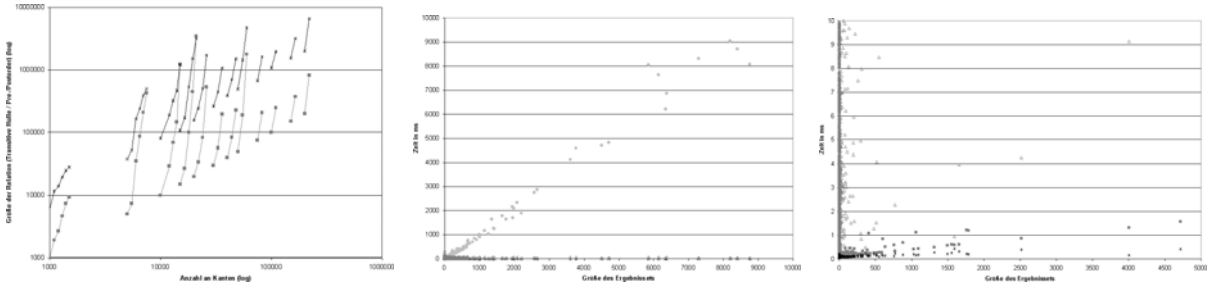
Speicherplatzbedarf. Um systematisch den Speicherplatzbedarf zu messen, habe ich Bäume und DAGs mit gegebener Zahl an Knoten erstellt. Abbildung 2(a) zeigt den Speicherplatzbedarf für die beiden Indexstrukturen. Der Startpunkt jeder Kurve stellt den Baum mit der entsprechenden Zahl an Kanten dar, während jeder weitere Punkt für einen DAG mit jeweils 10 % mehr Kanten steht.

Für einen Baum ist die Anzahl an Tupel, die in `prePostOrder` eingefügt werden, gleich der Anzahl an Knoten. Mit steigender Anzahl an Kanten im Graphen nähert sich die Anzahl an Tupeln in beiden Indextabellen an. Allerdings enthält für DAGs mit bis zu 30 % zusätzlichen Kanten die Tabelle `TC` noch über 50 % mehr Tupel als `prePostOrder` in den untersuchten Graphen. Werden 40 % oder mehr Kanten zusätzlich eingefügt, so kehrt sich die Situation um. Der Grund für dieses Verhalten ist die mehrfache Traversierung von Teilbäumen bei der Pre- /Postorder Ranking-Methode mit jeder zusätzlich eingefügten Kante. Auch die transitive Hülle wächst, doch je mehr Kanten bereits existieren, desto seltener wird eine neue Verbindung zwischen zwei Knoten erstellt, sondern nur alternative Pfade eingeführt.

Anfragezeiten. Die Anfragezeiten an Ontologien habe ich auf real existierenden Ontologien, dem Baum der NCBI Taxonomy und dem DAG der Gene Ontology, gemessen. Der Index für Pre-/ Postorder enthält in beiden Fällen weniger Tupel als die transitive Hülle, bei der NCBI Taxonomy 230 550 im Gegensatz zu 3 583 760. Auch die Gene Ontology (16 859 Knoten, 23 526 Kanten) hat nur 76 734 Tupel in dem Pre-/ Postorder Index im Gegensatz zu 178 033 für die transitive Hülle, obwohl bereits etwa 40 % zusätzliche Kanten eingefügt worden sind.

Die Suche nach allen Nachfahren für eine Menge zufällig ausgewählter Knoten zeigt in Abbildung 2(b) deutlich, dass die Anfragezeit der rekursiven Funktion linear von der Größe des Ergebnissets abhängt, während die Zeiten für die Indexstrukturen nahezu konstant bleiben. Abbildung 2(c) zeigt, dass für die Anfrage an die Pre-/ Postorder Indexstruktur die Option 2 aus 3.2 die bessere der beiden ist. Obwohl der Ausführungsplan für beide Anfragen jeweils einen Nested Loop-Join verwendet und in Oracle 9i identische Kosten vorhersagt, hängt die Anfragezeit der Option 1 von dem Pre- bzw. Postorder Rank – je nach verwendetem Index – ab. Je höher

dieser ist, desto länger dauert die Anfrage, da zuerst alle Tupel selektiert werden, die einen Pre- bzw. Postorder Rank haben, der kleiner ist als der des gegebenen Knotens. Bei Option 2 werden nur Tupel selektiert, die einen Preorder Rank innerhalb der gegebenen Grenzen besitzen. Die Antwortzeit für eine Anfrage mit einem Blattknoten variiert von 0,1 bis 2000 ms bei Option 1, während bei Option 2 die Zeiten unter 0,8 ms bleiben. Die Anfrage an die transitive Hülle liefert für viele der gewählten Knoten am schnellsten alle Nachfahren, wobei sie maximal um Faktor 4 schneller ist, was bei Zeiten um 1 ms nicht stark ins Gewicht fällt.



(a) Größe (log) der Indizes für generierte Bäume / DAGs.

(b) Anfragezeit an NCBI Taxonomy.

(c) Anfragezeit an Gene Ontology.

Abbildung 2: Größe der Indizes und Laufzeiten für Nachfahren eines Knotens v. $*$: transitive Hülle, $\triangle$: Pre-/ Post Option 1, $\blacksquare$: Pre-/ Post Option 2, $\bullet$: rekursive Funktion. Messungen mit Oracle 9i, auf DELL dual Xeon mit 4 GB RAM.

5 Zusammenfassung und Ausblick

Der Pre- und Postorder Index benötigt für Bäume und baumähnliche DAGs deutlich weniger Speicherplatz als die transitive Hülle. Anfragen nach allen Nachfolgern eines Knotens können mit beiden Indexstrukturen in konstanter Zeit beantwortet werden, wobei Anfragen an die transitive Hülle geringfügig schneller beantwortet werden. Auch andere Speicherstrukturen, die sowohl speicherplatzeffizient sind als auch Anfragen in konstanter Zeit erlauben, sollten nicht aus den Augen gelassen werden, wie der 2-hop-cover [5].

Literatur

[1] DL Wheeler, C Chappey, AE Lash, DD Leipe, TL Madden, GD Schuler, TA Tatusova, and BA Rapp. Database resources of the National Center for Biotechnology Information. *Nucleic Acids Research*, 28(1):10 – 14, Jan 2000.

[2] Gene Ontoloy Consortium. The Gene Ontology (GO) database and inforamtics resource. *Nucleic Acids Research*, 32:D258 – D261, 2004. Database issue.

[3] P. Valduriez and H. Boral. Evaluation of recursive queries using join indices. In L. Kerschberg, editor, *First International Conference on Expert Database Systems*, pages 271–293, Redwood City, CA, 1986. Addison-Wesley.

[4] Torsten Grust. Accelerating XPath location steps. In *Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, pages 109–120. ACM Press, 2002.

[5] Ralf Schenkel, Anja Theobald, and Gerhard Weikum. Efficient creation and incremental maintenance of the hopi index for complex xml document collections. In *ICDE*, 2005.

SERVICE PERSONALIZATION FOR USER SUPPORT

I. Vaynerman
Heinz-Nixdorf Endowed Chair for Practical Informatics
Institute of Computer Science
Friedrich-Schiller-University Jena
Jena, 07743, Germany

ABSTRACT

Mobile users need access to information sources. However, for a number of reasons, the traditional representation of query results is not useful in mobile environments. There is also a pressing question of using network services to provide mobile users with required information. The potential to dynamically bind services at run time is one of the big advantages of service oriented computing. However, dynamic service binding requires rather sophisticated service requests. These requests need to be precise enough to allow for automatic invocation of the service without user intervention, while at the same time flexible enough to allow finding all possible candidate services and could be rather clearly represented on a mobile device. Obviously, it is not trivial to formulate such requests. In this paper, we examine how the user could be supported in this complex task.

Keywords: service, personalization, priorities, mobile device, user support

INTRODUCTION

To provide mobile users with sophisticated means of information access is a problem of growing importance, in particular since the numbers of mobile users are growing continuously. Although mobile devices and wireless networks are steadily improving, some restrictions will remain for the foreseeable future. As compared with their stationary counterparts, mobile devices have smaller displays and less comfortable input possibilities.

Nowadays, the problems of mobile device limitations are solved either by splitting the information volume into smaller units or by picking the most important results. These existing approaches assume that a user's required information can be recognized beforehand. Moreover, the users themselves are believed to be able to determine and precisely formulate which information they need. In our opinion, both of these assumptions are unrealistic. It is thus necessary to develop new specific information delivery procedures for mobile and wireless users to adapt the information to the characteristics of these systems and user demands. Such a way of information delivering could be the personalisation of network services. In the next sections we present our approach of service personalisation.

SERVICE ORIENTED COMPUTING

Service orientation is a promising paradigm for distributed resource usage. Service orientation has the potential to fully automate resource sharing. Full automation makes particular sense for repetitive queries. As an example we will consider several steps of a business process on some mobile device (Figure 1). Let us assume that we should create a report in PDF format, print it and then send it via a courier service to our headquarters. We are using some mobile device e.g. a PDA or smartphone. At first, we create a report in .RTF format (e.g. in Pocket Word), but to make the further operations we need a .PDF document. The process of converting into PDF format needs high processor performance and the execution on a mobile device can take very long. Therefore, it makes sense to use a suitable network service instead of using the mobile application. Thus, we have to find and call such a service. So, after report converting the user needs to print it. He can do this in two ways. The first one is to use a rigidly determined printer. The user knows the name of this printer e.g. from previously using it, and sends the report to it. If our business process makes this step automatically we need to have 100% guarantees that the report will be printed.

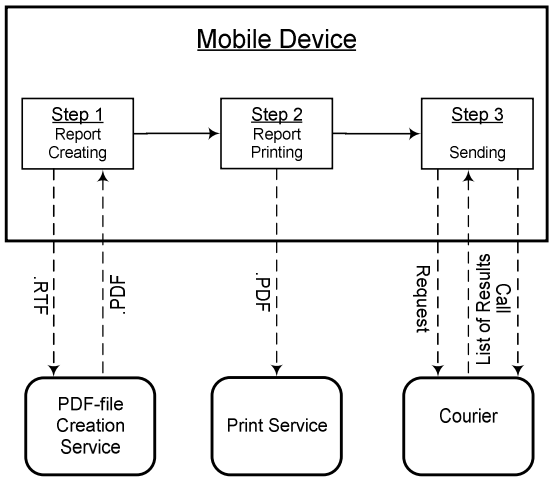


Figure 1. Business Process on a Mobile Device.

However, it could be that the user has gone from one place to another (i.e. from one building to another) and there is no connection with the remote printer, or it doesn't work. In this case, we should be able to choose the suitable printer each time before printing. Thus, it arises the second way to print the report: using a network printing service. Instead of calling the remote printer, we should send the request to print the report with some parameters to the service. Possible parameters would be, e.g., resolution, time or place, where the user can pick up the printed report. In an ideal case, the user should only send a request; the service chooses the appropriate printer and takes all necessary actions. When the report is printed and picked up, we need to send it by courier. Again, a service could be used. Obviously, if these three steps are carried out quite frequently, it should not require human intervention each time. Therefore, what is needed is a system that allows automatic service matching and binding. With such a system it would be possible to describe once which functionality should be provided and then automatically find the best match for this request at each invocation.

Full automation of service matching and binding requires complex service requests. The reason for this is that in conventional systems, the search results are presented to the user who then filters them and selects the most appropriate service provider. In a fully automated system, this manual filtering and selecting does not take place. This means that the initial request formulated by the user needs to be precise enough to strictly restrict the search result to services that offer exactly the functionality needed. Also, the request needs to contain all the information that is needed to rank among possible service providers. At the same time, service offers need to be described precisely, too, so that automatic matching becomes feasible. Within the DIANE project [3], suitable language for service request and offer description, DIANE Service Description (DSD), was developed [1]. It allows flexibly describing and processing the semantics of service offers and requests. If offers and requests are formulated in this language, automatic service matching and binding is possible.

Unfortunately, user requests become rather complex and nontrivial, if they are to match the requirements mentioned above. In our opinion, it is unlikely, that the user will succeed in formulating such a request without support by the system. The most likely scenario is that the user will formulate an initial request and will then check with a trial run whether the services found with this request match his expectations. If this is not the case, the user will adapt the request, rerun it, check the results once more and so on until, finally, the desired result set is retrieved. The request that produced this result set is then stored and used in future iterations of the process.

REQUIREMENTS FOR USER SUPPORT

Let us return to the third step of the business process introduced above. When the report is printed and picked up, we need to send it by courier. The main problem is to choose the appropriate courier company. Again, the user should be able to specify once what the characteristics of such a company are; it should then be ensured that during future iterations of the business process the request is processed and the optimal courier company for the task at hand chosen. The request can contain a lot

of parameters and priorities, since the decision on what courier service to use may be a complex one. The role of parameters in such a request could play: Price of Delivery, Term of Delivery and additional courier services (Online Billing Electronic Shipping, Delivery Tracking). Priorities here are the user-specific preferences in request parameters. Other words some request parameters could be more important to user than others. In DSD as such a mechanism to express user priorities in request serves "Connecting Strategy" (CS). Basically, CS is some mathematical formula, which integrated in each element of request. It specifies how different request parameters, which concern to this very request element, should be together combined. As a little example we consider the possible CS for courier service request: *(price*0.8 + term*0.2)/2*. Here *price* and *term* are the results of matching of service request and service offer. The coefficients *0.8* and *0.2* are actually the mathematical representation of user priorities for request parameters "Price of Delivery" and "Term of Delivery". These coefficients show that prise in this request is much important for user than term. During the request processing all its CS's will be evaluated in order to calculate the estimation of service offer. More details about CS and DSD you can find in the documentation of DIANE Project.

Thus, for a successful service request, i.e. one that results in the choosing of a matching service, the user needs

- To know which parameters can be specified for this particular type of service
- To determine which settings extend of each parameter is acceptable to him. For instance, what deviation in delivery price is acceptable? If user sets as value of prise parameter 10 euro, then the value 10,50 euro is still suitable to him, or not? The bounds of suitable parameter values should be defined by user in service request.
- To describe how different parameters should be prioritized, that is which parameter is how important. For instance, if the price of delivery is more critical for user than term, then he should be able to convey these priorities in the service request.

All these demands make the formulation of the query rather hard for the user. Also, if the list of the answers contains a lot of appropriate answers with similar characteristics, it should be possible to automatically rank them. Again, the parameters for the ranking need to be provided in the user query.

We mentioned above, that the user will probably not be able (even with support by the system) to come up with the "perfect" request, i.e. the request that produces exactly the intended output, from scratch. Rather, the formulation of the query will be an iterative process involving trial runs. These trial runs should be realized in a way that minimizes network load. In the next section, we take a closer look at the determination of preferences in the user request.

PRIORITY PARAMETERIZATION OF THE USER REQUESTS

In the previous Section we have shown that the process of service using becomes more and more complicated and there are a lot of objective reasons to provide user with means of support during the initial request formulation and further interaction. In our opinion the most suitable way to realize such a support is the service personalisation. Under this concept we can understand a process of service modification, which allows to consider user preferences during the interaction. So, user should be able to transmit his personal abstract interests (preferences) in some formal form that would be machine-understandable and enough flexible in dynamic iterative communication. Here we should more closely consider this term "abstract interests". What actually are these interests? To answer this question we should determine what usually does user to define a request. User sets the concretes values of request constituents, which basically are the request parameters. Hence he could have personal preferences for these parameters. In the rest part of this article we will name these favours "priorities". So, as we have said above, user should be able to transmit his priorities to the service, another words user priorities should be integrated in some formal way direct into the request. We name this integration "priority parameterization of the user request". To explain this complex concept we consider the concrete example.

Let us consider using of the courier-service from the previous section more closely. The user needs to send a report and he is searching for the company, which offers the best conditions. For these purposes he uses the network searching-service. The user defines the requirements for the courier-company. Now, it is necessary to determine the relations between the request parameters. These relations define the user's priorities, i.e. the hierarchy of the parameters. The priorities show what service aspects are more important than others. In other words the priorities define what request parameters are more critical for the user at service searching. They help the user to express his preferences in a request. During the processing of the request, it is necessary to separate the appropriate service offers from the not suitable. Also, the best matching service offer has to be determined. The DIANE Service Description (DSD) includes methods and algorithms to perform the matching. However, for these methods to work, the user request needs to contain precise information. For instance, it may be necessary to specify how certain attributes of an object should be matched. That is, the user may need to tell the system when to consider two values to be similar or which value of two given values to prefer. For example, with price it is better when it is smaller, but with printer resolution this is not so. This information needs to be provided somewhere. There is also a question about the range of parameter value. If there is no exact answer, in what borders it is reasonably to search for it? Since the answers to these questions need to be provided as part of the user request, the system should guide the user in pointing out what additional information is needed to allow for a meaningful matching of the user's request to service offers.

What we have described up to now are support mechanisms that will ensure that the user request is syntactically correct and contains all the information needed to perform service matching. However, while this is a necessary first step, it does not guarantee, that the user request does indeed describe the service the user is looking for. Therefore, the system should offer a mechanism that allows the user to perform trial runs of his requests. The mechanism should then display the results obtained and allow the user to adapt the request parameters. Even better would be a system that displays the results obtained, allows the user to indicate how close they are to the intended result and which then automatically adjusts the request parameters to obtain results closer to the intended ones.

In our work we have developed three approaches to realise such a complex set of support mechanisms. These three ways are based on the methods from neuronal networks theory, fuzzy-logic and mathematic functions. We separate the different types of user request and user priorities and according to the concrete case the specific methods are used. To provide user with means of efficient service using we are developing the Semantic User Profile. There user priorities could be saved and stored in a form of ontology.

RELATED WORK

In the sections above, we have shown that user support is required to enable users to successfully formulate complex requests. The main difficulty in formulating these requests is the correct prioritizing of user preferences. Therefore, in this section, we take a look at existing approaches that allow users to formulate preference based queries. This topic has enjoyed quite a lot of attention over the last few years. We will show that the existing approaches offer a valuable basis for our work. However, we will also show that they fall short of providing the kind of user support needed here.

Skline Queries [8] are preference based queries that return as a result the "skyline" of best results. These are those results that are better than others along at least one dimension, i.e. pare to optimal. Moving along the skyline is equivalent to shifting the relative importance of the different dimensions. With respect to our problem, a skyline algorithm seems a good starting point to find a set of candidate services. However, in addition to what the skyline algorithm provides, we need a method to decide which point of the skyline to choose as the best match of the user's expectations. Unfortunately, the existing algorithm does not offer any support here.

Preference SQL [6] is more flexible and allows the combination of different preferences not only as pareto preference, but also as prioritized preference or as a weighted sum of preferences. Thus, preference SQL allows expressing a wide variety of preferences. However, Preference SQL encounters the same problem that the formulation of requests with DSD poses: It is difficult for the

user to judge beforehand, how a certain setting will influence the query result. Preference SQL offers an explanation component that at least allows comprehending how a result was produced once this has been done. Again, Preference SQL seems like an interesting basis for implementation of preference based user requests. It does not address the fundamental problem described in this paper, that is how to come up with the correct request in the first place. The same is true for the XPath counterpart of Preference SQL, Preference XPath [5].

Service Globe [4] is another example of a system that allows (at least to a certain degree) the formulation of preference based queries. Again, no support for designing correct queries is given.

CONCLUSION

In this paper, we have shown that automatic service matching and binding require user support in formulating appropriate service requests. These requests will have to be preference based. We have discussed existing approaches to preference based querying and have shown that while they offer methods to compute query results based on a given query, they do not support the user in asking the correct query in the first place. Given the complexity of the queries required here, we believe that such a support is an absolute necessity. In our future work, we intend to develop a system that offers just this kind of support. This system should at the same time support the user efficiently and make economic use of sparse network resources.

The information about our previous works in the area of network services can be found in [10–13].

REFERENCES

1. Klein M., König-Ries B. “Combining Query and Preference - An Approach to Fully Automatize Dynamic Service Binding”. In: *Proc. of IEEE International Conference on Web Services (ICWS 2004)*, 6.-9. Juli 2004, San Diego, CA, USA.
2. Klein M., König-Ries B. “Coupled Signature and Specification Matching for Automatic Service Binding”. In: *Proc. of European Conference on Web Services (ECOWS 2004)*, 27.-30. Sept. 2004, Erfurt.
3. D I A N E - Projekt http://www.ipd.uka.de/DIANE/
4. Keidl M., Seltzsam S., Kemper A. “Reliable Web Service Execution and Deployment in Dynamic Environments”. *TES* 2003, pp. 104-118.
5. The official web-site of WWW Consortium http://www.w3.org/TR/xpath
6. Kießling W., Köstler G. “Preference SQL - Design, Implementation, Experiences”. VLDB 2002, pp. 990-1001.
7. Semantic Web Services http://www. daml.org/ services/
8. Kossmann D., Ramsak F., Rost S. “Shooting Stars in the Sky: An Online Algorithm for Skyline Queries” In: *Proc. of VLDB 2002*, Hong Kong, China, 2002.
9. Klein M., König-Ries B. “A Process and a Tool for Creating Service Descriptions based on DAML-S B”. In: *Proc. of 4th VLDB Workshop on Technologies for E-Services (TES'03)*. Berlin, 8 September 2003.
10. Koenig-Ries B., Popov D., Muelle J., Plechova O. "Multidimensional Query Result Navigation for Mobile Users". In: *Proc. “Computer Science and Information Technologies CSIT’2002”*. University of Patras, Greece, 2002,. http:// www. lar.ee.upatras.gr/ csit2002/.
11. Popov D.V., Vainerman I.A.: "A Prototype of the System to Provide Mobile Users with Services in Wireless Networks ". In: *Making Decisions under Indeterminacy Conditions*. USATU, Ufa, Russia, 2003, pp.14-22. (In Russian).
12. Yussupova N.I., König-Ries B., Popov D.V., Vainerman I.A. “Data Structures for a System to Dynamically Provide Mobile Users with Information in Wireless Networks”. In: *Proc. of the 5th International Workshop Computer Science and Information Technologies CSIT’2003*. Ufa, Russia, 2003, Vol. 1, pp. 100-107.
13. Yussupova N.I., Popov D.V., Vainerman I.A., König-Ries B. “RDF-Technologies to Provide Mobile Users with Services in Wireless Networks”. In: *Proc. of the 6th International Conference Complex Systems: Control and Modeling Problems*. Samara, Russia, 2004, pp. 203-208.

Towards Adaptive Ontology-Based Image Retrieval

Johanna Vompras

Institute of Computer Science
Heinrich-Heine-University Düsseldorf
D-40225 Düsseldorf, Germany
vompras@cs.uni-duesseldorf.de

Abstract

Since the use of large image databases gains in importance nowadays, efficient querying and browsing through image repositories becomes increasingly essential. Compared to text retrieval techniques there are even more problems with image retrieval. Particularly, the *semantic gap* between low-level visual features of images and high-level human perception of inferred semantic contents decreases the performance of traditional content-based image retrieval systems. The first important step for the correlation of image data with cognitive processes is the identification of discriminative features in the data. The next decisive step is to extract high-level knowledge from this data in order to provide a confident interpretation of signals into symbols. In this paper we demonstrate our first conceptual notions about the integration of spatial context and semantic concepts into the feature extraction and retrieval process using the relevance feedback procedure.

1 Introduction

Today, the management, storage, and retrieval of large image data repositories is a challenging problem. Due to their properties images are hard to handle with, particularly because of their diversity and the inadequate possibility to estimate and represent perceptional semantics. The basic difficulties emerging in analyzing and retrieval of image data often result from the following facts: Images mostly contain a non-uniform image background or a textual noise in the foreground, such as phone numbers or URLs, which make it impractical to differentiate between the relevant or irrelevant patterns. There are other problems emerging due to the data quality, like sharpness, contrast, and brightness of images. The heterogeneity of image data is also an issue to address to. The contents of the images may range from grey-scale to 24-bit color images and may have different sizes and formats.

Beside all these problems, the most important disruptive factor in image retrieval is the *lacking semantics*. Images are not structured data, which can be browsed on certain patterns which are arranged in a predefined manner. Views and shots can be taken from a variety of camera positions or images may contain many types of objects, like persons or animals. The objects may differ from each other at the visual feature level despite of belonging to the same semantic category.

The known traditional retrieval techniques based on textual image annotations ignore these difficulties. Here, the images are first labeled manually by keywords and are retrieved by their corresponding annotations [1]. Due to the high manual effort and the inconsistency in image interpretation and keyword assignment among indexers, this approach becomes impracticable [2].

Efficient and accurate information retrieval is of great demand in the field of image databases. A first important pre-processing step in the image mining process is the feature selection and extraction which have been the basis for numerous heuristic retrieval methods and machine-learning methods with relevance feedback in content-based image retrieval (CBIR).

Nevertheless, users are highly interested in querying images at a conceptual and semantic level, not only in terms of features like color, texture, or shape [3]. Since it is very difficult to involve the human perception to an automated image retrieval task, methods which combine human cognition ability and the automatic computation are of great interest [4].

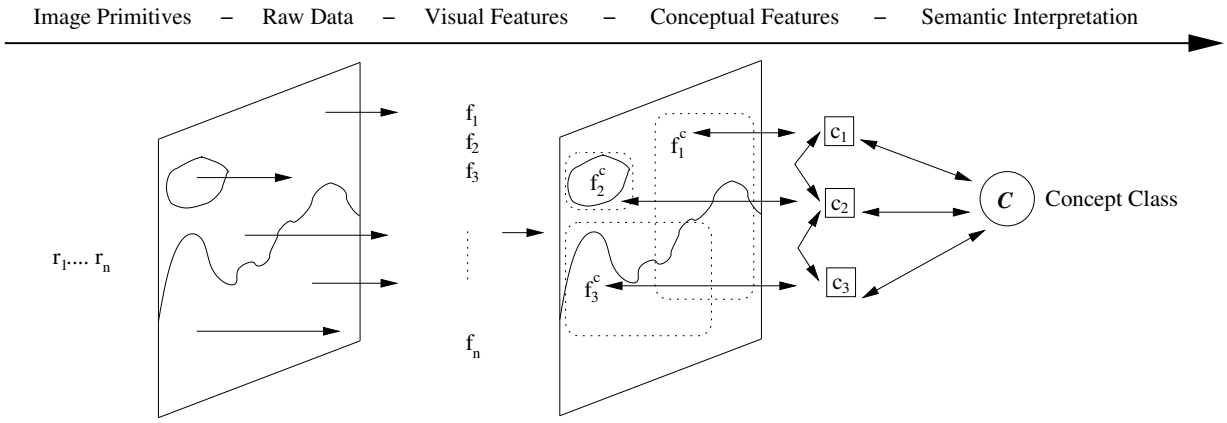


Figure 1: Levels of Image Content Representation

In this paper we demonstrate our first conceptual notions about the integration of spatial context and semantic concepts into the feature extraction and retrieval process using the relevance feedback procedure. Through an interactive system which allows both feeding the system with additional semantic information and capturing the correspondences between the low-level features and high-level *perceptual semantics*, the quality of retrieval results could be improved. Thus, query formulations at the semantic level will enhance current information systems.

2 Data Representation

In order to achieve our above-mentioned aims, an image and semantic data model has to be developed which can be coupled for retrieval purposes. Thus, a modified multimedia object model from [2] is used for representing low-level features and high-level perceptual semantics. Accordingly, an image object O is represented as

$$O = O(D, F, R, C, S) \tag{1}$$

where D represents the raw image data, $F = \{f_i\}$ a set of low-level visual features and $R = \{r_{ij}\}$ a set of representations for a given feature f_i. The model introduced in [2] is extended by the component C which denotes the set of semantic concepts $\{c_k\}$ and their inter-concept relationships $S = \{s_{kl}\}$. A given concept is also described by a set of features f_i^c which are characteristic for it in the low-level space. In addition, a semantic feature space is built with references to image objects or image regions in the database. The semantic network is represented by a set of keywords having links to the image objects or regions in the images. The initial network can be formed during the extraction of images from the web or digital television by using textual information. More concepts can be learned from the user's feedback and thus expand the semantic network.

In order to support a multiple-level description of image contents, weights are used at various levels. The aim of relevance feedback is not only the optimization of the weights to model the users information need but also modeling high-level data and thus forming a semantic space.

2.1 Image Representation

Since there is no semantic information in the data at the beginning of the retrieval session, the contents of image objects are only characterized by their visual features f_i which result from the computation of the image processing primitives $r_1 \ldots r_n$.

2.2 Semantic Representation

An ontology gives an explicit specification of an abstract view of the world, which is represented by inter-relationships among a set of representational terms that we call *concepts*. Let us suppose that our world can be described by the set $\mathcal{K}$ of structured concepts which are hierarchically organized. Our real-world

model is organized as a net of concepts linked together and providing a formal description of the relationships of the concepts. Since the description of the whole world is relatively voluminous, it is desirable to use a simplified specification for a particular application field which reveals the correspondences between low-level features and semantic knowledge.

In the semantic feature space, each image $\mathcal{I}$ includes a set of n different concepts which are represented by a set of m characteristics $p_1 \dots p_m$ and their weights w_{mkl}. Each characteristic is represented by one relationship between different concepts as displayed in Table 1 and the corresponding weights. From this it follows that the spatial and semantic relationships $rel(c_k, c_l)$ between two concepts c_k and c_l are values in a m-dimensional vector in the semantic space, where m is the overall number of existing relations between concepts. Thus, the semantic content of an image object is represented by a vector $\vec{v}$

$$\vec{v} = \begin{pmatrix} c_1 \\ c_2 \\ .. \\ .. \\ c_n \end{pmatrix} \text{ and } rel(c_k, c_l) = \begin{pmatrix} w_{1kl} \\ w_{2kl} \\ .. \\ .. \\ w_{mkl} \end{pmatrix}$$

where its components c_k are represented by:

$$c_k = \begin{cases} 1 & : \text{ if object is classified to concept } k \\ 0 & : \text{ else.} \end{cases}$$

The learning mechanism must be able to learn the classification of image objects into concepts, and thus deliver an appropriate answer to a query. An image region belongs to a particular concept c with its *partial concepts* or *subconcepts* $\bar{c}_1 \dots \bar{c}_n$ if

$$\sum_{1}^{n} w_i \bar{c}_k > \Theta \tag{2}$$

where Θ is a application specific threshold and the w_k represent the weights assigned to the subconcepts $\bar{c}_k$ within the concept c. The weights of the subconcepts are adapted during the learning process. For example, if we consider the concept *tennis match* which is characterized by description *'a match between two tennis players'* in a ontology: This concept has a `has-part` relationship with the terms *tennis ball*, *tennis players*, *tennis court*, and *green grass*. Through the relevance-feedback by the user the weights of the subconcepts are adjusted to the users perception and thus mirror the meaningfulness of the subconcepts in the whole context. For example, if there are also sand-colored courts in our world, the user will specify a lower weight to the component *green grass*.

The membership of the over-all image contents to a *concept class* Ω is computed by checking if its containing concepts c_k are located in immediate neighborhood N of the class Ω. The whole image is associated with class Ω if

$$N(\Omega, r) := \{c_k \in C \mid \forall k \text{ dist}(\Omega, c_k) \leq r\} \tag{3}$$

where r denotes the radius of the semantic neighborhood of the concept class Ω. The distance between two concepts in the ontology is measured along the shortest path, which is formed by the smallest number of undirected arcs that connect the entity classes. These undirected arcs represent semantic relationship between two concepts (e.g `is-a-kind-of` and the `has-part`). A concept class Ω denotes a superordinate concept which describes the aggregation of the subconcepts c_k in this special context is assigned to a certain image according to the semantic hierarchy or defined by the user during the relevance feedback procedure.

In order to link semantic knowledge of our world into the retrieval process, an appropriate representation supporting inter-conceptual and intra-conceptual modifications of weights and a suitable similarity function is needed.

3 Relevance Feedback Method in Retrieval

A decisive step in the embedding of semantic information in the retrieval process is the mapping of low-level feature space into a high-level semantic space. Every detected object in the image is an additional information of our world. Through this information the users' *semantic space* is formed out of the recognized concepts and their semantic relationships are updated in each retrieval step, so that this

Spatial and Topological Relations	Lexical and Semantic Relations
a `left-of` b	a `is-a-kind-of` c
a `right-of` b	a `has-part` c
a `above` b	a `is-synonym-for` c
a `below` b	a `is-antonym-for` c
a `near-to` b	a `is-member-of` c
a `far-to` b	
a `inside` b	
a `outside` b	
a `in-front-of` b	

Table 1: Spatial and semantic relations between objects in an image.

created knowledge is used in subsequent query sessions. The relationships between images are also captured in the semantic space. After several steps of relevance feedback the user has determined a pool of images which are relevant to his query [5]. We can make the assumption that these images belong to the same semantic class. Due to this result set of images, the subsequent query object O^q is extended by additional query points $q_1 \dots q_n$ in the feature space that are related with the same semantic class.

3.1 Construction of semantic feature space

Since the semantic description of many fields of application is infinite in size, the semantic space is formed out of a partial mapping of real world objects (ontology space) into a semantic feature space. This space should summarize the most important concepts and relations between them discovered by humans. In analogy to the *cognitive space* introduced in [6] it is in constant change as a result of internal processes and interaction with the user. The relevance feedback procedure is a user-centered loop [2] with the following iteration steps:

1. Initialization of the low-level representation's weights and the semantic representation's weights into a set of no-bias weights.
2. User query in form of a labeled sample image or formal description of low-level features. The query object O^q is described by its features f_i^q.
3. The similarity of the query object O^q to the image objects is computed iterative over all representation levels according to the corresponding similarity measure.
4. The returned k image objects are ordered by their similarity $d(.)$ to O^q.
5. User labels a subset of the retrieved objects according to his perception subjectivity.
6. Weights are updated according to the user's feedback such that the adjusted O^q is a better approximation to the user's information need.
7. The semantic space is extended such that the recognized concepts are inserted into the space hierarchy and the positive labeled image objects results in a incremental movement of concept points in the semantic space.
8. Go to 2 and start a new iteration of retrieval.

3.2 Similarity Measure

The similarity function may deliver a suitable distance measure for both low-level image features and users subjective semantic features. In step 4 of the feedback loop (Section 3.1) we use a distance metric function d to measure the similarity between the query vector O^q and the stored image objects. The query vector O^q is formed either by an image object or rather by its visual features like in the *query-by-example approach* or as a combination of an image object with semantic description. As all the weights in the system, the similarity measure is also a dynamic function depending on the current state of the system i.e. weights and semantic information. Generally speaking, the distance between an query object O^q and an image object O in the database is defined by

$$d(O^q, O) = \frac{\alpha\, d_l(O^q, O) + \beta\, d_s(O^q, O)}{\alpha + \beta} \tag{4}$$

where α and β are from range $[0 \ldots 1]$. They denote the degree of belief in the values computed by d_l and d_s which represent distance functions for both low-level and semantic representation of the query object O^q. The parameters α and β are determined by counting the number of query points in the semantic space which are close to the point of the query object. If the number is small the similarity computation is rather based on low-level features.

4 Future Work

CBIR is an important technology for an effective and efficient retrieval in image databases. Since it is difficult to capture high-level semantics of images when images are only represented and queried by low-level features, methods which embed learning semantic concepts are of great demand. The weighting of low-level features and high-level semantics both in the data representation and similarity computation assists in finding a set of images which should be close to the association of what users are looking for. Of course, there are some questions left:

- How can the accuracy of different similarity functions be refined in order to be optimized the precision of the query result?
- How can semantic feature representation enhance the detection of objects by inference mechanisms?
- How can the model be used to deliver a graphical framework for user interaction and knowledge extension?
- How can the semantic model be compressed and summarized in order to improve the retrieval performance?

References

[1] G. Salton and M. McGill. *Introduction to modern Information Retrieval.* New York.

[2] T. Huang, Y. Rui, M. Ortega, and S. Mehrotra. Relevance feedback: A power tool for interactive content-based image retrieval. *IEEE Transactions on Circuits and Systems for Video Technology*, 1998.

[3] J. Torres, A. Parkes, and L. Corte-Real. Region-based relevance feedback in concept-based image retrieval. *Proc. of the 5th International Workshop on Image Analysis for Multimedia Interactive Services, Lisboa, Portugal.*, 2004.

[4] A. B. Benitez and J.R. Smith. New frontiers for intelligent content-based retrieval. *Proc. SPIE Vol. 4315, Storage and Retrieval for Media Databases, 141-152*, 2000.

[5] X. He, O. King, W.-Y. Ma, M. Li, and H.-J. Zhang. Learning a semantic space from user's relevance feedback for image retrieval. *IEEE Transactions on Circuits and Systems for Video Technology, Vol. 13, 39-48.*, 2003.

[6] G. B. Newby. Cognitive space and information space. *Journal of the American Society of Information Science and Technology. 52, 1026-1048*, 2001.

Liste der Teilnehmer

Stephan Audersch
Zentrum für Graphische Datenverarbeitung e.V.
Rostock
Joachim-Jungius-Str. 11
18059 Rostock
audersch@rostock.zgdv.de

Ammar Balouch
Lehrstuhl Datenbank- und Informationssysteme
Universität Rostock
Albert-Einstein-Straße 21
18059 Rostock
ab006@informatik.uni-rostock.de

Sascha Berger
Institut für Informatik
Ludwig-Maximilians Universität München
Oettingenstr. 67
80538 München
bergers@pms.ifi.lmu.de

Joos-Hendrik Böse
Institut für Informatik
Freie Universität Berlin
Takustr. 9
14195 Berlin
boese@mi.fu-berlin.de

Prof. Dr. Stefan Braß
Institut für Informatik
Martin-Luther-Universität Halle-Wittenberg
Von-Seckendorff-Platz 1
06099 Halle (Saale)
brass@informatik.uni-halle.de

Prof. Dr. Stefan Conrad
Institut für Informatik
Datenbanken und Informationssysteme
Heinrich-Heine-Universität Düsseldorf
Universitätsstr. 1
40225 Düsseldorf
conrad@cs.uni-duesseldorf.de

Guntram Flach
Zentrum für Graphische Datenverarbeitung e.V.
Rostock
Joachim-Jungius-Str. 11
18059 Rostock
gf@rostock.zgdv.de

Tim Furche
Institut für Informatik
Ludwig-Maximilians Universität München
Oettingenstr. 67
80538 München
tim@furche.net

Christian Goldberg
Institut für Informatik
Martin-Luther-Universität Halle-Wittenberg
Von-Seckendorff-Platz 1
06099 Halle (Saale)
goldberg@informatik.uni-halle.de

Jürgen Göres
Fachbereich Informatik
AG Datenbanken und Informationssysteme
Technische Universität Kaiserslautern
Postfach 3049
67653 Kaiserslautern
goeres@informatik.uni-kl.de

Alexander Holupirek
Institut für Informatik und Informationswiss.
Universität Konstanz
Universitätsstr 10 / D 188
78457 Konstanz
alexander.holupirek@uni-konstanz.de

Katja Hose
Institut für Praktische Informatik und
Medieninformatik
Datenbanken und Informationssysteme
Technische Universität Ilmenau
Postfach 100 565
98684 Ilmenau
katja.hose@tu-ilmenau.de

Heiko Jahnkuhn
Lehrstuhl Datenbank- und Informationssysteme
Universität Rostock
Albert-Einstein-Straße 21
18059 Rostock
hj016@informatik.uni-rostock.de

Prof. Dr. Hans-Joachim Klein
Institut für Informatik und Praktische Mathematik
Technologie der Informationssysteme
Universität Kiel
Hermann-Rodewald-Str. 3
24118 Kiel
hjk@is.informatik.uni-kiel.de

Sascha Koch
Oldenburger Forschungs- und
Entwicklungsinstitut für Informatik-Werkzeuge
und –Systeme (OFFIS)
Echerweg 2
26121 Oldenburg
koch@offis.de

Hieu Le Quang
Institut für Informatik
Datenbanken und Informationssysteme
Heinrich-Heine-Universität Düsseldorf
Universitätsstr. 1
40225 Düsseldorf
lqhieu@cs.uni-duesseldorf.de

Thomas Leich
Institut für Technische und Betriebliche
Informationssysteme
Otto-von-Guericke-Universität Magdeburg
Universitätsplatz 2
39106 Magdeburg
leich@iti.cs.uni-magdeburg.de

Mazeyar E. Makoui
Institut für Informationssysteme
Fachgebiet Datenbanksysteme
Universität Hannover
Welfengarten 1
30167 Hannover
makoui@dbs.uni-hannover.de

Christian Mathis
Fachbereich Informatik
AG Datenbanken und Informationssysteme
Technische Universität Kaiserslautern
Postfach 3049
67653 Kaiserslautern
mathis@informatik.uni-kl.de

Prof. Dr. Wolfgang May
Institut für Informatik
AG Datenbanken & Informationssysteme
Georg-August-Universität Göttingen
Lotzestrasse 16-18
37083 Göttingen
may@informatik.uni-goettingen.de

Thomas Müller
Institut für Informatik
Datenbanken und Informationssysteme
Friedrich-Schiller-Universität Jena
Ernst-Abbe-Platz 2
07743 Jena
mueller@informatik.uni-jena.de

Markus Preißner
Eichendorffstr. 46
73230 Kirchheim/Teck
Markus.Preissner@gmx.de

Jan Rittinger
Institut für Informatik und Informationswiss.
Universität Konstanz
Universitätsstr 10 / D 188
78457 Konstanz
jan.rittinger@uni-konstanz.de

Matthias Rust
Zentrum für Graphische Datenverarbeitung e.V.
Rostock
Joachim-Jungius-Str. 11
18059 Rostock
mrust@rostock.zgdv.de

Dr.-Ing Eike Schallehn
Institut für Technische und Betriebliche
Informationssysteme
Otto-von-Guericke-Universität Magdeburg
Universitätsplatz 2
39106 Magdeburg
eike@iti.cs.uni-magdeburg.de

Christoph Schmitz
Fachbereich Mathematik/Informatik
Fachgebiet Wissensverarbeitung
Universität Kassel
Wilhelmshöher Allee 73
34121 Kassel
schmitz@cs.uni-kassel.de

Anke Schneidewind
Institut für Technische und Betriebliche
Informationssysteme
Otto-von-Guericke-Universität Magdeburg
Universitätsplatz 2
39106 Magdeburg
a.schneidewind@iti.cs.uni-magdeburg.de

Henrike Schuhart
Institut für Informationssysteme
Universität zu Lübeck
Osterweide 8
23562 Lübeck
schuhart@ifis.uni-luebeck.de

Petra Schwaiger
Lehrstuhl für Informationsmanagement
Universität Passau
Gottfried-Schäffer-Straße 20
94032 Passau
schwaige@im.uni-passau.de

Boris Stumm
Fachbereich Informatik
AG Heterogene Informationssysteme
Technische Universität Kaiserslautern
Postfach 3049
67653 Kaiserslautern
stumm@informatik.uni-kl.de

Silke Trißl
Institut für Informatik
Wissensmanagement in der Bioinformatik
Humboldt-Universität zu Berlin
Unter den Linden 6
10099 Berlin
trissl@informatik.hu-berlin.de

Igor Vaynerman
Institut für Informatik
Heinz-Nixdorf-Stiftungsprofessur für Praktische Informatik
Friedrich-Schiller-Universität Jena
Ernst-Abbe-Platz 1-4
07743 Jena
igor@informatik.uni-jena.de

Johanna Vompras
Institut für Informatik
Datenbanken und Informationssysteme
Heinrich-Heine-Universität Düsseldorf
Universitätsstr. 1
40225 Düsseldorf
vompras@cs.uni-duesseldorf.de

Christoph Wieser
Institut für Informatik
Ludwig-Maximilians Universität München
Oettingenstr. 67
80538 München
wieser@cip.ifi.lmu.de