Kapitel 16: Informationsintegration

In großen Unternehmen existiert oft ein breites Spektrum, weitgehend unabhängig voneinander entwickelter und auf unterschiedlichen Softwarekomponenten basierender, datenbankgestützter Anwendungssysteme, die mehr oder weniger „Insellösungen“ darstellen. Mehr und mehr entsteht jedoch der Bedarf zur Integration solcher Insellösungen, um betriebliche Abläufe globaler optimieren zu können und Marketing- wie auch unternehmensstrategische Entscheidungen besser vorbereiten zu können. Für die sich hiermit stellende Problemstellung der Integration heterogener, strukturierter, durch Schemata beschriebener Datenquellen wird generell eine sogenannte *Wrapper-Mediator-Architektur* verfolgt, die aufbaut auf

- je einem *Wrapper* pro Datenquelle, der die eigentliche Datenquelle (z.B. eine relationale Datenbank) kapselt und deren Daten oder Operationen in aufbereiteter, zur Integration vorbereiteter Form bereitstellt, und
- mindestens einem *Mediator* (Broker, Föderations-Integrator), der die eigentliche Integration der von den Wrappern bereitgestellten Daten oder Operationen leistet und den Clients integrierte Daten und/oder Operationen anbietet.

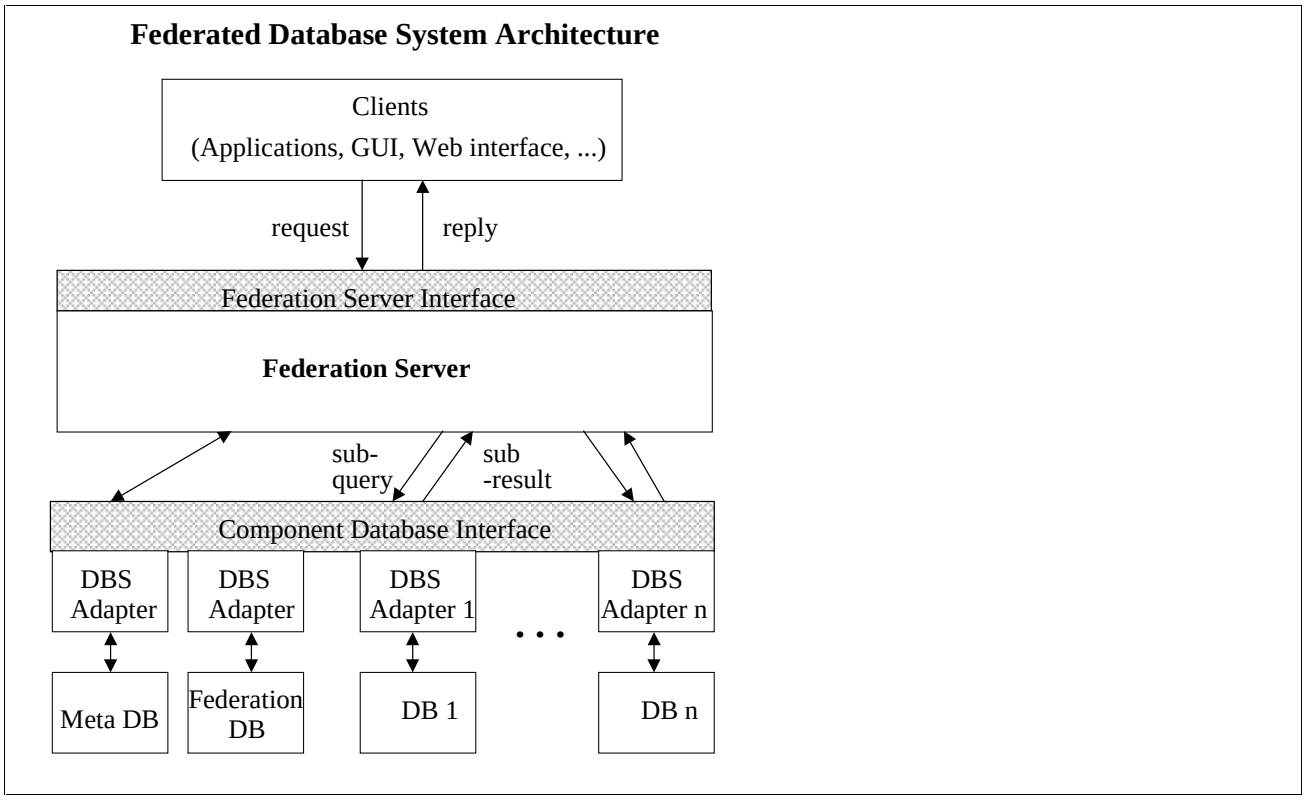
Je nach „Tiefe“ der Integration und Aufgabenteilung zwischen Wrappern, Mediator und ggf. den Clients ergeben sich verschiedene Familien von Integrationsarchitekturen für strukturierte Daten, die im folgenden skizziert werden.

16.1 Föderierte Datenbanken

Bei der Architektur einer aus mehreren *Komponenten-Datenbanken bestehenden Datenbank-Föderation* stellen Wrapper – hier auch *Datenbank-Adapter* genannt – ein aufbereitetes, zur Integration vorbereitetes Schema der jeweiligen Komponenten-Datenbanken zur Verfügung, lassen aber alle Daten unter der Kontrolle des jeweiligen Komponentensystems.

Häufig wird nur ein Teil der jeweiligen zur Integration benötigt bzw. freigegeben; daher wird das vom Wrapper bereitgestellte Schema häufig auch *Export-Schema* genannt. Dieses enstpricht quasi einer View auf die eigentlichen Daten des Komponenten-Systems. Der Wrapper muß daher zusätzlich eine Abbildung dieser View auf das Original-Schema der Komponenten-Datenbank, im wesentlichen also eine enstprechende Familie vordefinierter Queries, beinhalten.

Der Mediator – hier auch *Föderations-Server* genannt – integriert die exportierten „lokalen“ Schemata der Komponenten in ein globales Schema, das in einer Meta-Datenbank abgelegt wird, und unterstützt Queries und Update-Operationen auf den gemäß dieses globalen Schemas virtuell integrierten Daten. Dazu sind globale Queries in Subqueries zu zerlegen, die jeweils auf den exportierten Views ausgeführt werden, indem die jeweiligen Datenbank-Adapter die Viewdefinitionen in den Subqueries substitutieren und die resultierenden Queries ausführen. Der Föderations-Server übernimmt auch die Integration der von den Komponenten-Systemen bzw. Datenbank-Adaptern gelieferten Teilresultate zu einem globalen Resultat. Um temporäre Zwischenresultate weiterverarbeiten (z.B. aggregieren) zu können, kann dem Föderations-Server ggf. eine spezielle Föderations-Datenbank zur Verfügung stehen.



Die größten Schwierigkeiten im Aufbau einer Datenbankföderation liegen in

1) der *Heterogenität der Komponentendatenbanken* bzgl. ihrer Struktur und vor allem ihrer Schemasemantik und
2) der *Autonomie der Komponentensysteme*, die neben globalen Queries und Transaktionen weiterhin rein lokale Anwendungen bedienen und Beeinträchtigungen der lokalen Last hinsichtlich Leistung und Verfügbarkeit so weit wie möglich ausschließen möchten (ohne jedoch globale Inkonsistenzen der Daten zu riskieren).

Ersteres führt auf das Problem der *Schemaintegration*, letzteres auf Probleme mit der Behandlung *globaler Transaktionen*.

Bei der Integration unabhängig voneinander entworfener Komponenten-Schemata treten verschiedene Arten von strukturellen und vor allem semantischen Konflikten auf, die auf Föderations-Ebene zu bereinigen sind:

- *Strukturkonflikte* entstehen, wenn dieselbe Information in verschiedenen Komponenten mit unterschiedlichen Datenmodellkonstrukten repräsentiert wird.
  Beispiel:
  DB1: VerwAngestellte (PersNr, Name, AbtNr, AbtBez)
  WissAngestellte (PersNr, Name, AbtNr, AbtBez)
  DB2: Angestellte (PersNr, Name, PersKategorie, AbtNr)
  Abteilungen (AbtNr, AbtBez)
  In DB1 werden Abteilungen als Attribute von Angestellten repräsentiert, in DB2 in Form einer eigenen Relation. In DB1 sind Verwaltungsangestellte und wissenschaftliche Angestellte in zwei Relationen partititioniert, in DB2 sind alle Angestellten in einer Relation und die Spezialisierung ist durch ein zusätzliches Attribut repräsentiert.
  In ausdrucksstärkeren Datenmodellen, insbesondere objektorientierten Datenmodellen oder semantischen Datenmodellen à la ERM, sind derartige Strukturkonflikte tendenziell noch augenfälliger, wenn z.B. dieselbe Information einmal als Attribut und einmal als Objektklasse bzw. Entitymenge repräsentiert ist.

- *Namenskonflikte* entstehen, wenn dieselbe Informationseinheit in verschiedenen Komponenten unterschiedlich benannt ist (*Synonyme*) oder gleich benannte Informationseinheiten unterschiedliche Bedeutung haben (*Homonyme*).
  Beispiel:
  DB1: Teile (TeileNr, TeileBez, ..., Einkaufspreis)
  DB2: Teile (TeileNr, TeileBez, ..., Preis)
  DB3: Teile (TeileNr, TeileBez, ..., Preis)
  In DB1 und DB2 (z.B. separaten Datenbanken für Lagerverwaltung und Buchführung) könnten Einkaufspreis und Preis Synonyme sein, wohingegen die Preise von DB2 und DB3 (letzteres z.B. eine Datenbank für den Vertrieb) Homonyme wären, wenn der Preis in DB3 ein Verkaufspreis ist.

- *Datentypkonflikte* liegen vor, wenn dieselbe Informationseinheit mit unterschiedlichen Datentypen repräsentiert wird.
  Beispiel:
  DB1: Teile (..., Preis, ...) als Float oder Number
  DB2: Teile (..., Preis, ...) als Varchar

- *Codierungskonflikte* liegen vor, wenn dieselbe Informationseinheit mit demselben Datentyp auf unterschiedliche Weise codiert wird, z.B. aufgrund landes- oder sprachspezifischer Konventionen.
  Beispiel:
  DB1: Bestellungen (BestNr, ..., BestDatum, ...)
  mit Datum als Varchar in der Form "5.2.1999"
  DB2: Order (OrderNo, ..., OrderDate, ...)
  mit OrderDate als Varchar in der Form "2/5/99"

- *Dimensionskonflikte* liegen vor, wenn dieselbe Informationseinheit in unterschiedlichen Maßeinheiten angegeben ist.
  Beispiel:
  DB1: Angestellte (PersNr, ..., Gehalt, ...)
  mit Gehalt in DM pro Monat als Bruttogehalt
  DB2: Employees (EmpNo, ... Salary, ...)
  mit Salary in $ pro Jahr nach Abzug der Steuern

Bei der Schemaintegration sind daher - sehr grob gesprochen - folgende Schritte zur Bereinigung dieser Konfliktarten notwendig:

- Strukturvereinheitlichungen durch Umstrukturierung der exportierten Daten
- Namensanpassungen durch Umbenennung von Homonymen und Synonymen
- Festlegung von Konvertierungsprozeduren für Datenwerte zur Bereinigung von Datentyp-, Codierungs- und Dimensionskonflikten

Bei der abschließenden Zusammenfassung der bereinigten Komponentenschemata sind dann ggf. Generalisierungen zu bilden, wenn spezialisierte Daten desselben virtuellen "Supertyps" aus verschiedenen Datenbanken vereinigt werden sollen.

Beispiel für die Vorgehensweise bei der Schemaintegration (zur Illustration der Problematik):

DB1 (Bibliotheksdatenbank):
Buch (ISBN, Name, Verlag, BibliotheksName)
Autor (Name, Adresse)
Bibliothek (BibName, Adresse)
Deskriptoren (ISBN, Bezeichnung)
Autorenschaft (ISBN, AutorenName)
DB2 (Literaturdatenbank):
Publikationen (Name, Titel, Verlag)
Stichwörter (Code, Fachgebiet)
Katalog (Titel, Verlag, Code)

1) Korrespondierende Informationseinheiten bestimmen:
   DB1.Autor.Name entspricht DB2.Publikationen.Name
   DB1.Buch.Name entspricht DB2.Publikationen.Titel bei Büchern
   DB1.Deskriptoren.Bezeichnung entspricht DB2.Stichwörter.Fachgebiet
   usw.
2) Umbenennungen:
   DB1.Buch.Name in DB1.Buch.Titel umbenennen
   DB1.Autor.Name und DB2.Publikationen.Name in AutorenName umbenennen
   DB2.Stichwörter.Fachgebiet in DB2.Stichwörter.Bezeichnung umbenennen
   usw.
3) Generalisierungen:
   DB1.Buch als Spezialisierung von DB2.Publikationen interpretieren
   oder eine gemeinsame Generalisierung bilden

Ein mögliches, aus den beiden Komponenten DB1 und DB2 erstelltes föderatives Schema (das nur einen Teil der Information übernimmt und integriert) wäre z.B.:

FDB: Publikationen (Titel, Verlag)
Bücher (Titel, Verlag, ISBN)
Autoren (Titel, Verlag, Autorenname)
Stichwörter (Titel, Verlag, Bezeichnung)

Um auf die integrierten Daten über das FDB-Schema zugreifen zu können, benötigt der Föderations-Server noch die entsprechenden Transformationen, die quasi die Relationen der FDB als Views auf die Relationen von DB1 und DB2 definieren:

FDB.Publikationen (Titel, Verlag) :=
π[Name, Verlag] (DB1.Buch) $\cup$ π[Titel, Verlag] (DB2.Publikationen)
FDB.Bücher (Titel, Verlag, ISBN) :=
π[Name, Verlag, ISBN] (DB1.Buch)
FDB.Autoren (Titel, Verlag, Name) :=
π[Name, Verlag, AutorenName] (DB1.Autorenschaft |X| DB1.Buch) $\cup$
π[Titel, Verlag, Name] (DB2.Publikationen)
FDB.Stichwörter (Titel, Verlag, Bezeichnung) :=
π[Name, Verlag, Bezeichnung] (DB1.Buch |X| DB1.Deskriptoren) $\cup$
π[Titel,Verlag,Fachgebiet] (DB2.Publikationen |X| DB2.Katalog |X| DB2.Stichwörter)

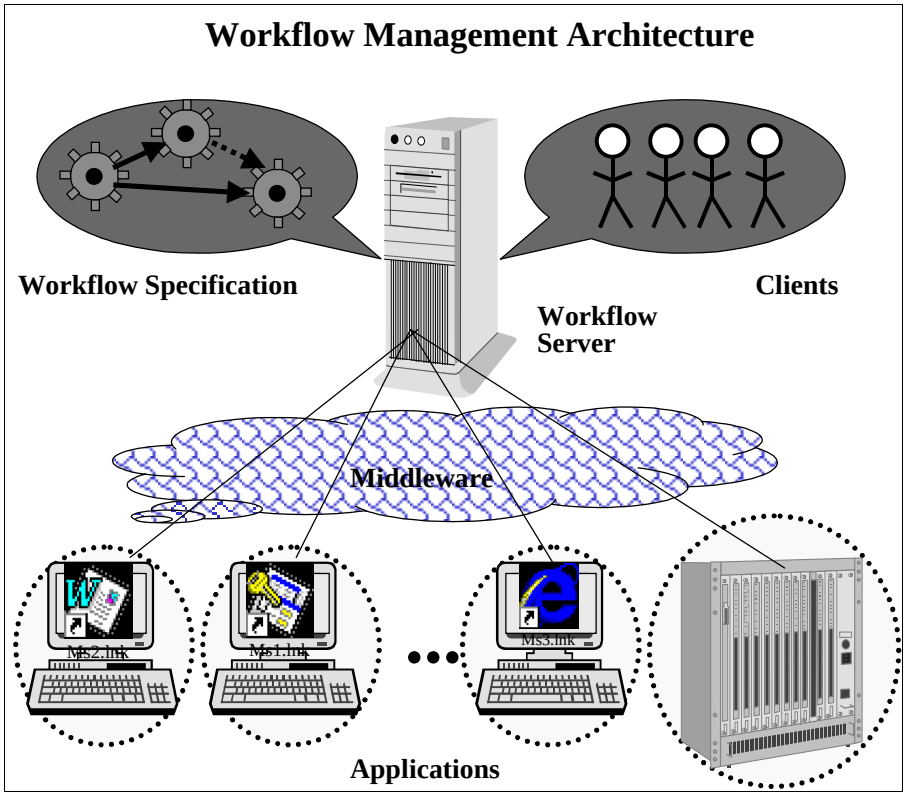
16.2 Workflow-Management und Middleware

Bei der Integration über ein Workflow-Management-System bleiben die Daten der Komponentensysteme gekapselte Objekte in dem Sinne, dass auf sie nur über die lokal existierenden Applikationen der Komponentensysteme zugegriffen werden kann. Das Workflow-Management-System steuert den Kontroll- und Datenfluß globaler Abläufe durch Interpretation einer Workflow-Spezifikation. In diese Workflows sind Applikationsaufrufe eingebettet.

Damit auf diese Weise ein möglichst breites Spektrum verschiedenartiger Applikationen auf unterschiedlichen Plattformen integriert werden kann, müssen Applikation durch entsprechende Wrapper erst workflow- bzw. integrationsfähig gemacht werden und werden dann über entsprechend standardisierte "Middleware" aufgerufen. Die in der Praxis etabliertesten Middleware-Architekturen sind

- CORBA (Common Object Request Broker Architecture), spezifiziert vom Industriekonsortium und Standardisierungsgremium OMG (Object Management Group) und
- COM bzw. DCOM (Distributed Component Object Model) von Microsoft.

Beide erwarten für jede aufrufbare Applikation eine IDL-Schnittstelle (Interface Definition Language), die ggf. vom entsprechenden Wrapper zu realisieren ist und realisieren auf dieser Basis - ggf. über einen sogenannten Object Request Broker (ORB) - ortstransparente Methodenaufrufe auf beliebig verteilten, heterogen (z.B. in verschiedenen Programmiersprachen) implementierten Objekten. Aus den existierenden Applikationen werden somit "Business Objects", die in standardisierter Form aufrufbar und damit auch leichter wiederverwendbar sind.


Workflow Management Architecture
Workflow Specification
Clients
Workflow
Server
Middleware
Ms2.lnk
Ms1.lnk
Ms3.lnk
Applications

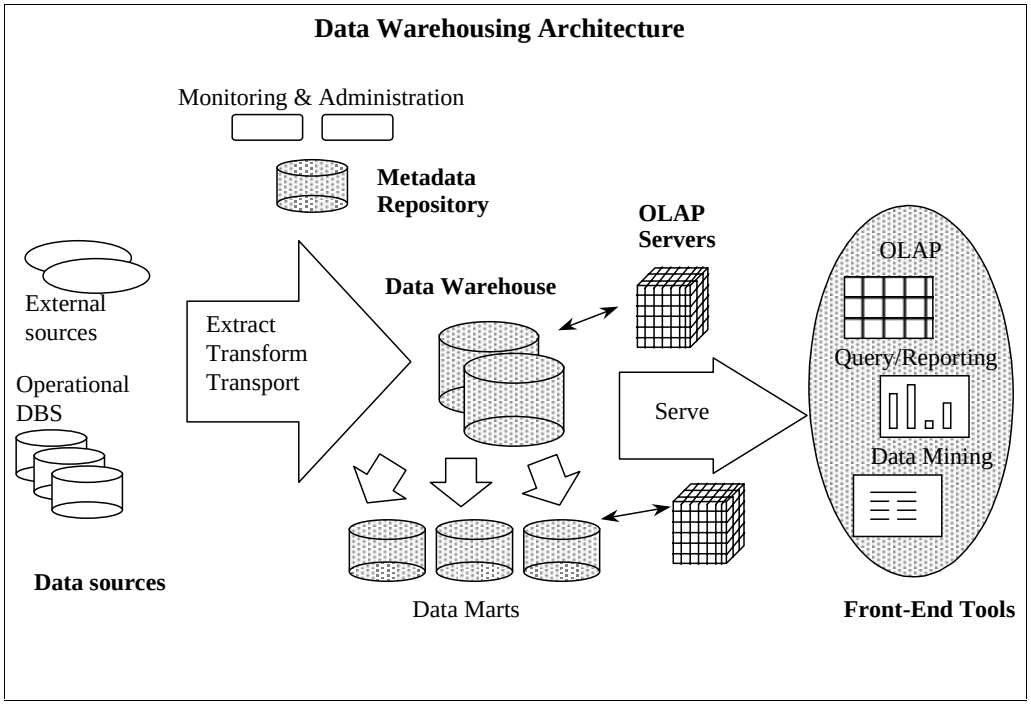
16.3 Data-Warehouses

Wenn die Integration verschiedener Datenbanken ausschließlich entscheidungsunterstützenden Queries und Datenanalysen mit rein lesendem Zugriff dient, bietet sich der Aufbau eines sogenannten Data-Warehouses an, in das periodisch oder kontinuierlich Daten aus verschiedenen Datenquellen kopiert und dabei bedarfsgerecht umstrukturiert werden. Im Gegensatz zur Datenbank-Föderation werden also wirkliche Daten und nicht nur Schemata integriert. Dennoch ist dieser Ansatz einfacher realisierbar als ein vollständiger Föderations-Server, da

- die Daten im Data-Warehouse nur für Lesezugriffe zur Verfügung stehen müssen,
- die Autonomie der zugrundeliegenden Datenquellen hinsichtlich lokaler Performance und Verfügbarkeit weitgehend unangetastet bleibt und
- das Schema des Data-Warehouses auf spezifische Datenanalysen - *sogenannte OLAP-Anwendungen (Online Analytical Processing)* zugeschnitten werden kann.

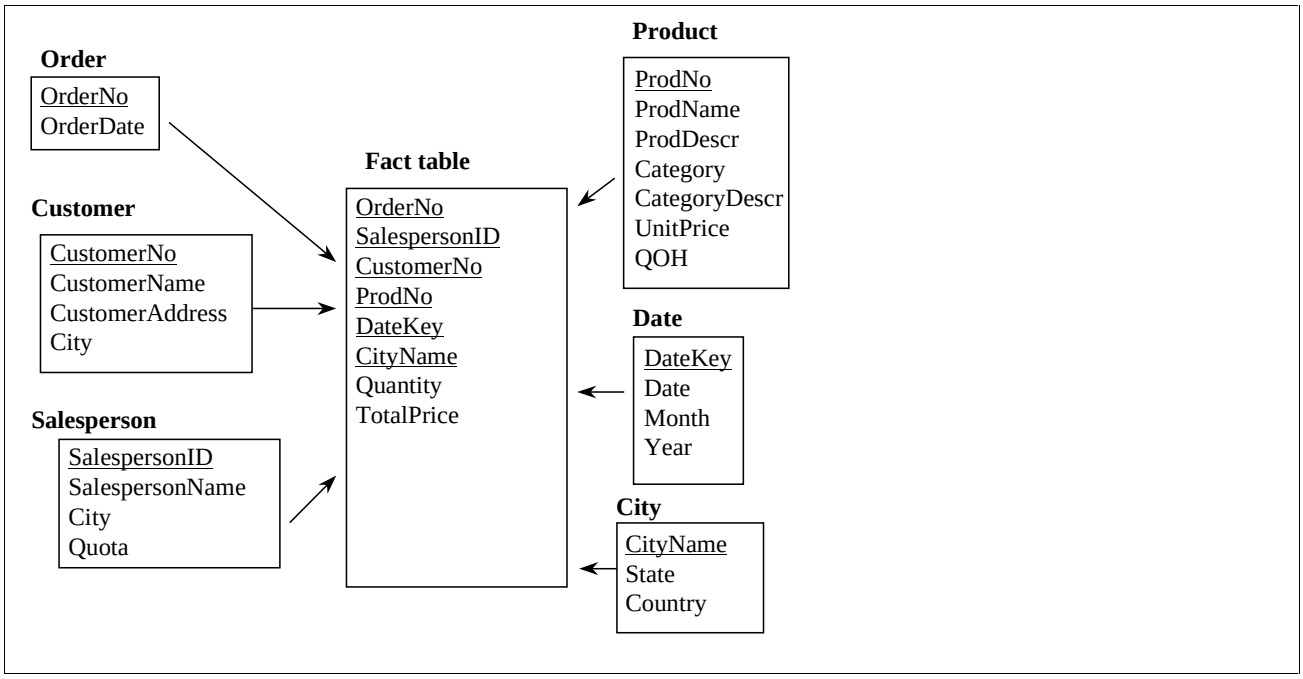
Architektur, Aufbau und Pflege eines Data-Warehouses umfassen also die folgenden Komponenten bzw. Schritte:

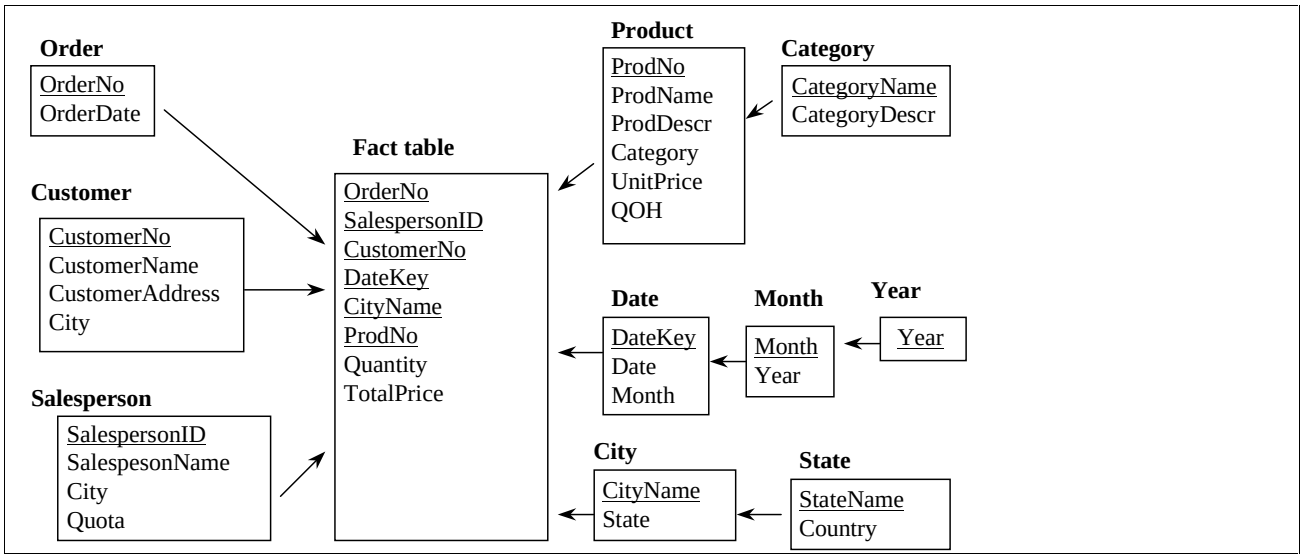
- Entwurf eines Schemas für das Data-Warehouse
- Vorbereitung der Datenextraktion aus den Datenquellen
- Bereinigung, Transformation und Kopieren der Daten in das Data-Warehouse
- Betrieb eines OLAP-Servers auf den Daten des Data-Warehouse
  (ggf. Extraktion von Teilmengen in spezielle OLAP-Tools der Clients)
- Belieferung spezialisierter, kleinerer Data-Marts
- Physischer Datenbankentwurf und Tuning des Data-Warehouses
  (Indexauswahl, Einsatz von Bitmap-Indizes, Join-Indizes, View-Materialisation, usw.)



Als Richtlinie für den konzeptionellen Datenbankentwurf des Data-Warehouses werden häufig sogenannte Stern- oder Schneeflocken-Schemata propagiert. Einem Stern-Schema liegt eine - in der Regel sehr große - sogenannte *Fakten*-Relation als "Zentrum" zugrunde, die zu einer Anzahl kleinerer, sogenannter *Dimensions*-Relationen jeweils eine Fremdschlüsselbeziehung hat; die Dimensions-Relationen haben untereinander keine Fremdschlüsselbeziehungen. Ein Schneeflocken-Schema verallgemeinert diesen Ansatz dahingehend, daß die Dimensions-Relationen Fremdschlüsselbeziehungen zu hierarchisch übergeordneten Dimensions-Relationen haben können.

Beispiele für ein Stern-Schema und ein Schneeflocken-Schema:





Typische Queries auf einem Data-Warehouse sind von der Art:

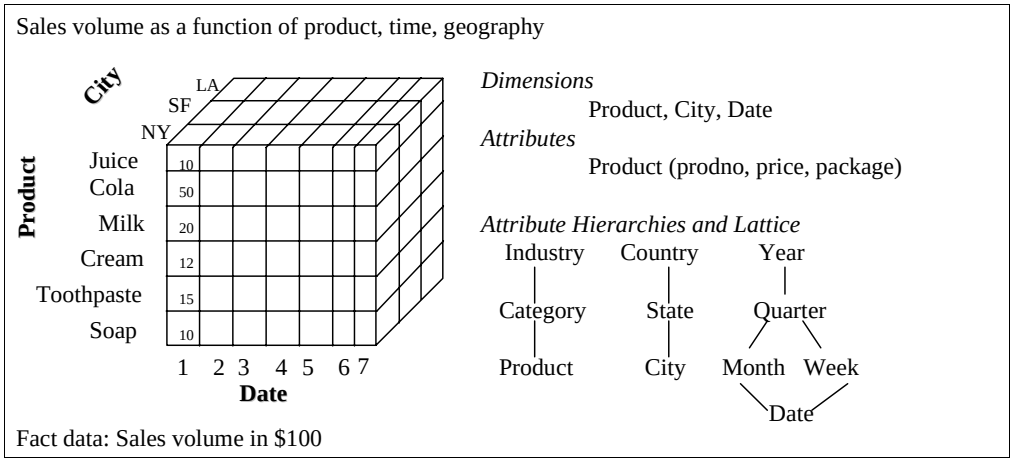
- Wie gut verkaufen sich Badeartikel im Saarland während der Wintermonate?
- Welche Entwicklung haben die Umsätze mit PC-Zubehör in den letzten drei Jahren genommen – bundesweit und aufgeschlüsselt nach Regionen? Wie sehen diese Trends bei bestimmten Arten von Zubehör aus (Peripherie, Software, Spiele, usw.)?
- Welches sind unsere umsatzstärksten Kunden? Mit welchen Produkten verdienen wir das meiste Geld? (bundesweit, in spezifischen Regionen, usw.)
- Um wieviel (und warum) sind die Verkaufszahlen für Badeartikel im Saarland schlechter als in Schleswig-Holstein?

Strukturell ähnliche Queries treten auch in nichtbetrieblichen Anwendungen auf, z.B. medizinischen Anwendungen:

- Wie hängt die Dauer der stationären Klinikaufenthalte von den verschriebenen Medikamentgruppen ab, aufgeschlüsselt nach Geschlecht, Altersgruppe und Berufsgruppe der Patienten?

Derartige Queries und Analysen auf den Daten eines Data-Warehouse erfolgen vielfach mit speziellen OLAP-Tools, die aus einer Client-Komponente zur Datenvisualisierung und einem OLAP-Query-Server auf der Seite des Data-Warehouse bestehen. Sehr häufig werden die Daten dabei als mehrdimensionaler Würfel (engl.: Data Cube) dargestellt, dessen Zellen verschiedene Werte der Fakten-Relation für die entsprechenden Werte der Dimensions-Relationen enthalten. Auf solchen Würfeln ist ein Spektrum von intuitiv einfach handhabbaren Basisoperationen ausführbar, die vom OLAP-Server in entsprechende SQL-Queries umgesetzt werden müssen:

- *Roll-up*: Vergröbernde Aggregationen durch Gruppierung in einer oder mehreren Dimensionen mit Berechnung von Aggregationsfunktionen wie Sum, Avg, usw.
- *Drill-down*: Verfeinern der Werte(gruppen) einer oder mehrerer Dimensionen mit entsprechender Aufschlüsselung der aggregierten Faktenwerte
- *"Pivot", "Slice&Dice"*: Projektionen des Würfels auf bestimmte Dimensionen mit entsprechender Werteaggregation ("Drehen und Schneiden des Würfels")



Solche an der Würfel-Metapher orientierte Operationen können häufig mit OLAP-spezifischen Erweiterungen von SQL kombiniert werden. Derartige Erweiterungen sind u.U. auch OLAP-Server-intern zur effizienteren Auswertung der OLAP-Queries sinnvoll.

Beispiele:

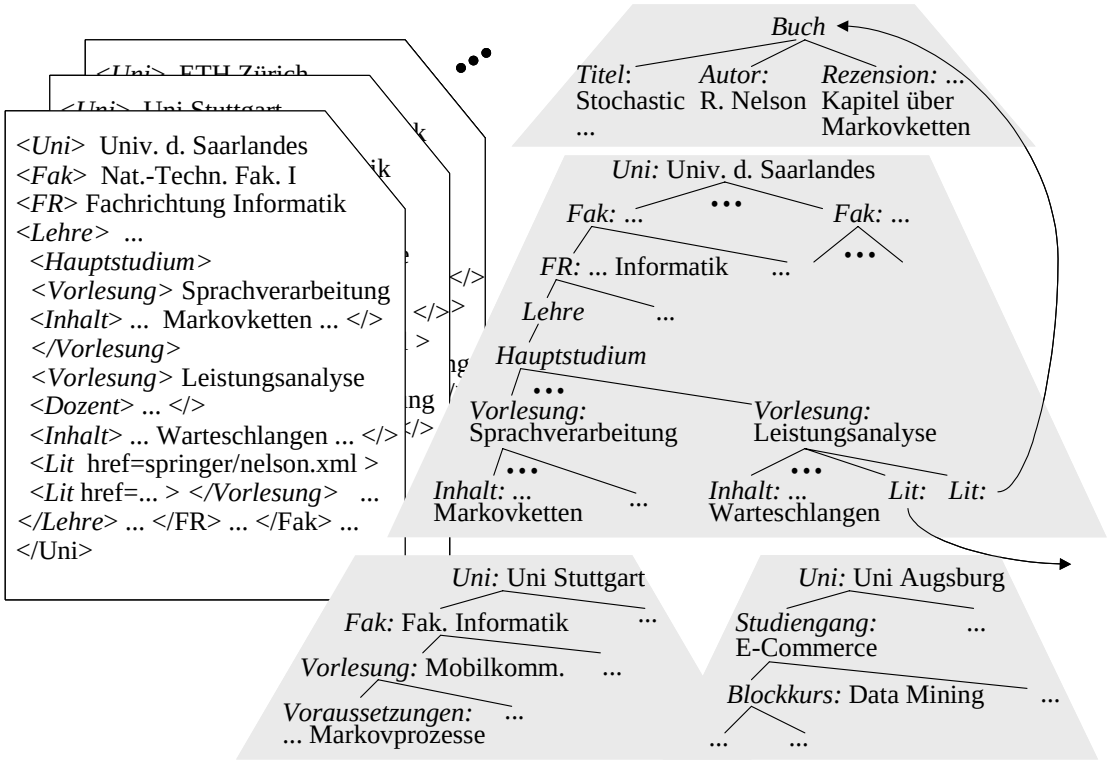
- Zusätzliche Aggregationsfunktionen wie
  - *Median* (Fakten.Attribut),
  - *Mode* (Fakten.Attribut),
  - *MovingAverage* (Fakten.Attribut, 3 Rows) und *MovingSum* (...) bzw.
    MovingAverage (Fakten.Attribut, 3 Months)

  usw.
- Ranking bezüglich aggregierter Werte, z.B.
  - *Rank* (...)
    zur Bestimmung des Rangs eines Kunden, z.B. der 10 umsatzstärksten Kunden
  - *Tile* (...)
    z.B. zur Bestimmung der meistverkauften Produkte, die zusammen einen bestimmten Prozentsatz des Gesamtumsatzes ausmachen
- Mehrfachgruppierung und entsprechende Aggregationen:
  Group By A1, A2, ..., Am Cube entspricht mehreren Gruppierungen
  Group By A1; *Group By A1, A2*; ...; *Group By A1, A2, ..., Am*
  in einer einzigen Query

16.4 XML

Der W3C-Standard XML (Extensible Markup Language) ist von zentraler Bedeutung zur Datenrepräsentation und zum Datenaustausch für Business-Anwendungen im Web und in Intranets. Mit ihm können sowohl strukturierte Daten, wie man sie typischerweise in (objekt-) relationalen Datenbanken speichert, als auch semistrukturierte Dokumente, wie man sie typischerweise im Web findet, einheitlich beschrieben werden. XML-Daten können schematisiert sein, müssen es aber nicht. XML hat von daher ein großes Potential als einheitliche Beschreibungssprache beim Aufbau von Unternehmens-Portalen für das Web, bei Föderationen autonomer Datenbanken (z.B. Produktkatalogen oder Proteindatenbanken) und nicht zuletzt für den Datenfluß in Business-to-Business-Workflows. Kommerzielle Datenbanksysteme unterstützen bereits heute die weitgehend automatische Konversion von objekt-relationalen Daten in XML und umgekehrt.

Anders als HTML erlaubt XML frei definierbare, themenspezifische "Tags" zur Strukturierung und semantischen Annotation von Dokumenten in Form sog. *Elemente*. Die Schachtelung von Elementen führt auf Bäume, die die Struktur von XML-Dokumenten wiedergeben und auch Hyperlinks auf andere Bäume beinhalten können. Die folgende Abbildung zeigt Ausschnitte aus XML-Dokumenten mit Elementnamen *Uni*(versität), *Fak*(ultät), *Vorlesung*, *Lit*(eratur) usw. sowie die abstrahierte Darstellung dieser Daten als Bäume, die zu einem Datengraphen verbunden sind. Die adäquate Präsentation solcher Daten im Web-Browser ist nicht Teil von XML, sie wird durch andere Standards und Werkzeuge abgedeckt.



Formal besteht ein XML-Element – wie ein HTML-Element – aus einem öffnenden Tag der Form <elementname attributliste> und einem schließenden Tag </elementname> sowie einem Elementinhalt zwischen den beiden Tags. Eine Attributliste besteht aus einer jeweils durch Blank getrennten Liste von Einträgen der Form *attributname="attributwert"*. Beispielsweise könnte im o.a. Beispiel das Uni-Elemente um zwei Attribute wie folgt erweitert werden:

<Uni TelNr="0681/302-0" Email="verw@uni-saarland.de">.

Attributnamen können genauso wie Elementnamen frei vergeben werden. Attribute können auch Hyperlinks oder Verweise auf Elemente im selben oder in anderen XML-Dokumenten sein.

Der formale Aufbau eines XML-Dokuments kann durch eine sog. DTD (Document Type Description) oder ein XML-Schema eingeschränkt werden. Eine DTD ist im wesentlichen eine kontextfreie Grammatik, die spezifiziert, welche Elementnamen in Dokumenten des spezifizierten Typs verwendet werden dürfen, welche Attribute diese Elemente haben können und auf welche Weise Elemente geschachtelt werden dürfen. Die folgende Abbildung zeigt eine mögliche DTD für eine Kollektion von Vorlesungsverzeichnissen. Bei DTDs werden für die Element- und Attributinhalte nur wenige verschiedene Typen unterschieden; im wesentlichen gibt es nur Strings (PCDATA). Bei einem XML-Schema dagegen wird eine breite Palette von Datentypen unterstützt (z.B. verschiedene numerische Typen, Datums- und Währungsangaben).

```
<!DOCTYPE Uni [
  <!ELEMENT Uni ((Fak | FR)+, Präsident?, Vizepräsident*)>
  <!ATTLIST Uni TelNr CDATA #REQUIRED>
  <!ATTLIST Uni EMail CDATA #IMPLIED>
  <!ELEMENT Präsident (#PCDATA)>
  <!ELEMENT Vizepräsident (#PCDATA)>
  <!ELEMENT Fak (FR+)>
  <!ELEMENT FR (Lehre, Forschung?, Verwaltung)>
  <!ELEMENT Lehre (Grundstudium, Hauptstudium)>
  <!ELEMENT Hauptstudium (Vorlesung+, Seminar*, Praktikum*)>
  <!ELEMENT Vorlesung (Dozent*, Inhalt?, Lit*)>
  <!ELEMENT Vorlesung (#PCDATA)>
  ... usw.
]>
```

XML selbst ist eigentlich nur eine Syntax zur Datenstrukturierung und -annotation. Eine wichtige Bedeutung dieses Standards liegt darin, dass er ein sehr großes Moment erzeugt, Daten im Web besser zu organisieren und damit effektiver suchbar zu machen, indem zu bestimmten Themen Elementnamen gewählt werden, die - so weit wie möglich und vom jeweiligen Autor gewünscht - die kanonische Terminologie und Begriffshierarchie des Themengebiets reflektieren. In diesem Kontext entstehen derzeit zahlreiche "standardisierte Ontologien", beispielsweise für die Vielfalt derivativer Finanzinstrumente (siehe www.fpml.org) oder Teilgebiete der Bioinformatik (siehe z.B. www.geneontology.org). Bei Vorlesungsverzeichnisdaten im Stil von Abbildung 1 hätte man dann Elemente wie <Bachelor-Studiengang>, <Stammvorlesung> oder <Praktische Informatik>, und bei XML-Dokumenten mit mathematischen Inhalten würde man Elemente wie <theorem>, <vector space>, <rank> oder <eigenvalue> erwarten.

Wie sehen nun Anfragen auf solchen XML-Daten aus? Die in verschiedenen Prototypsystemen und auch einigen kommerziellen Produkten unterstützten XML-Anfragesprachen (z.B. XQL, XPath, XQuery usw.), darunter auch der ganz frisch verabschiedete W3C-Standard XQuery, kombinieren logische Bedingungen a la SQL mit Pfadausdrücken (ähnlich denen von OQL aber nicht unbedingt schematisiert) und allgemeineren Suchmustern für Baumstrukturen. Pfade beziehen sich dabei auf Elementnamen im Datenbaum, und Suchmuster können Wildcard-Zeichen (* bzw. //) sowie reguläre Ausdrücke über Elementnamen enthalten. Zwei Beispiele für Anfragen in den – sehr mächtigen – Sprache XML-QL (a) und XQuery (b) sind die folgenden:

1) Welche Fakultäten welcher Universitäten bieten eine Vorlesung mit Titel „XML“ an?

a) in XML-QL:

```
WHERE <Uni> $u
            <*.Fak> $f
               <*.Vorlesung> XML </>
            </>
          </>
IN www.alleunis.de/unis.xml
CONSTRUCT  <result>
                 <Universität> $u
                    <Fakultät> $f </Fakultät>
                 </Universität>
               </result>
```

b) in XQuery:

```
FOR $u IN document("www.alleunis.de/unis.xml")/Uni
LET $f := $u//Fak
WHERE $f//Vorlesung = "XML"
RETURN <result>
            <Universität> $u
              <Fakultät> $f </Fakultät>
            </Universität>
          </result>
```

2) An welchen Universitäten gibt es in der Informatik einen Dozenten, der sowohl in der theoretischen Informatik wie auch in der praktischen Informatik Vorlesungen hält?

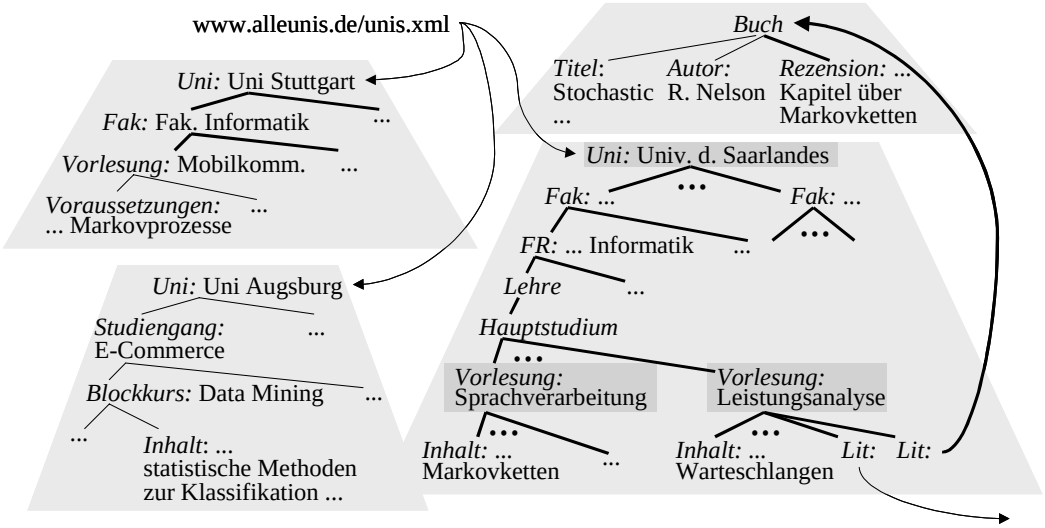
a) in XML-QL:

```
WHERE <Uni> $u
            <*.(Fak | FR | FB).Name?> Informatik
               <*.Vorlesung> $v1
                  <*.Dozent> $d </> </>
               <*.Vorlesung> $v2
                  <*.Dozent> $d </> </>
            </>
          </>
IN www.alleunis.de/unis.xml,
   $v1 LIKE ‘%theoretisch%’, $v2 LIKE ‘%praktisch%’, NOT ($v1 = $v2)
CONSTRUCT  <result> $u $d </result>
```

b) in XQuery:

```
FOR $u IN document("www.alleunis.de/unis.xml")/Uni
LET $f := $u//(Fak | FR | FB)  |  $u//(Fak | FR | FB).Name,
      $v1 := $f//Vorlesung, $v2 := $f/Vorlesung,
      $d1 := $v1//Dozent, $d2 := $v2//Dozent
WHERE $d1 = $d2
AND contains ($v1, "theoretisch")
AND contains ($v2, "praktisch")
AND NOT ($v1 = $v2)
RETURN <result> $u $d </result>
```

Die an der Universität des Saarlandes entwickelte Anfragesprache XXL (Flexible XML Search Language) unterstützt den Kern von XQuery mit einer enger an SQL angelehnten Syntax. Es können Suchmuster in Form von regulären Ausdrücken über den Elementnamen auf Pfaden des Datengraphen mit logischen Bedingungen auf Elementinhalten verbunden werden. Die Suche nach Universitäten, an denen in der Informatik Vorlesungen angeboten werden, die Markovketten behandeln, kann auf diese Weise sehr kompakt mit der in der nachfolgenden Abbildung gezeigten Anfrage ausgedrückt werden; über diese textuelle Spezifikation hinaus gibt es ein Werkzeug mit graphischen Benutzeroberflächen, das den Benutzer bei der Anfragespezifikation führt und die eigentliche Anfrage generiert. In dem Anfragebeispiel sind # ein Wildcard-Zeichen für Pfade mit keinem, einem oder mehreren Elementen, ? ein Symbol für optionale Elemente, | eine Disjunktion, + die einmalige oder mehrmalige Wiederholung eines Teilpfades und % ein Platzhalter für beliebige Strings in Elementinhalten. Das Suchergebnis - alle Teilgraphen des Datengraphen, die die Suchbedingungen erfüllen - ist in der Abbildung durch die farbigen Markierungen dargestellt.

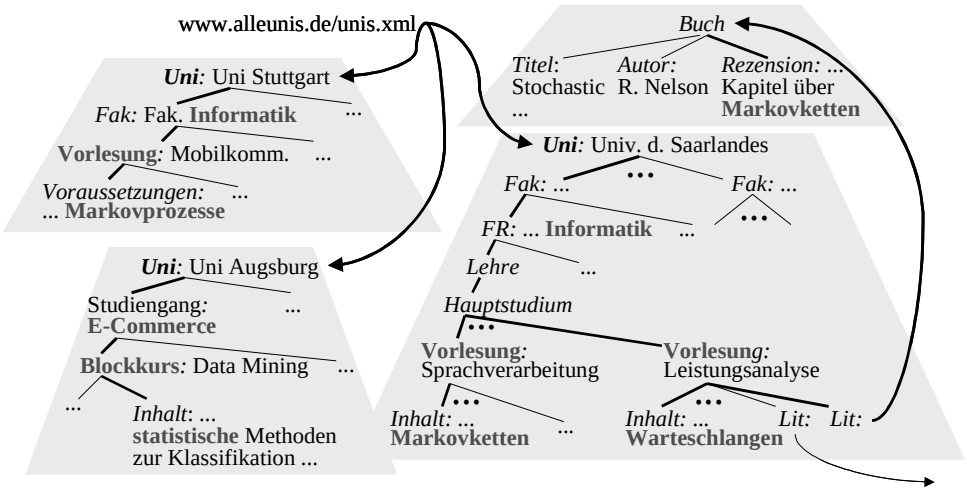



```
Select U, V From www.alleunis.de/unis.xml Where Uni As U
And U.#.Fak?.#.((FR | FB)+)? As F And F Like „%Informatik%“
And F.#.Vorlesung As V And V.# Like „%Markovketten%“
```

Nun gibt es zwar einen starken Trend zu kanonischen Elementnamen, aber (zum Glück unserer Kultur, jedoch als Handicap für die maschinelle Suchbarkeit) gibt es auch Variationen in Terminologie,

Stil und Strukturierung von Dokumenten zum selben Thema. Die eigentliche Besonderheit von XXL ist es, auch diese natürliche Diversität von Web- und Intranet-Daten beherrschen zu können, indem zusätzlich "semantische" Ähnlichkeitsbedingungen in Anfragen unterstützt werden. Das Suchresultat ist eine Rangliste von "ungefähren" Treffern, die nach Relevanz bzw. Ähnlichkeit zur Anfrage absteigend sortiert ist. Für das Beispiel aus der vorhergehenden Abbildung würde man mit der in der nachfolgenden Abbildung gezeigten Ähnlichkeitsanfrage die Suche verbreitern und bekäme als Resultat die folgende Rangliste (in absteigender Relevanz):

1. Vorlesung Sprachverarbeitung an der UdS
2. Vorlesung Leistungsanalyse an der UdS
3. Vorlesung Mobilkommunikation an der Uni Stuttgart
4. Blockkurs Data Mining an der Uni Augsburg



Select U, V From www.alleunis.de/unis.xml Where Uni As U
And U.# As F And F ~~ „Informatik“
And F.#.~Vorlesung.# ~~ „Markovketten“

Dabei sind ~ ein auf Elementnamen definierter unärer Ähnlichkeitsoperator und ~~ ein sich auf Elementinhalte beziehender binärer Vergleichsoperator. Die Berechnung der Relevanzwerte basiert auf einem probabilistischen Modell; die zugrundeliegenden "semantischen" Ähnlichkeitswerte zwischen verschiedenen Elementnamen (z.B. Vorlesung versus Blockkurs oder Lehre versus Studiengang) oder Termen in Elementinhalten (z.B. Markovketten versus Markovprozesse versus Warteschlangen versus statistische Methoden) werden aus einer hierarchischen Ontologie abgeleitet.

Ergänzende Literatur zu Kapitel 16:

T. Özsu, P. Valduriez, Principles of Database Systems, 2nd Edition, Prentice-Hall, 1999

S. Conrad, Föderierte Datenbanksysteme, Springer-Verlag, 1997

A. Dogac, L. Kalinichenko, T. Özsu, A. Sheth (Editors), Advances in Workflow Management and Interoperability, NATO Advanced Study Institute, Springer-Verlag, 1998

F. Leymann, D. Roller, Production Workflows - Concepts and Techniques, Prentice Hall, 2000

R. Orfali, D. Harkey, J. Edwards, Client/Server Survival Guide, John Wiley & Sons, 1999

W. Emmerich, Engineering Distributed Objects, John Wiley & Sons, 2000

A. Geppert, K. Dittrich, Component Database Systems, Morgan Kaufmann, 2000

IEEE Data Engineering Bulletin Vol.21 No.3, September 1998, Special Issue on Interoperability

S. Chaudhuri, U. Dayal, An Overview of Data Warehousing and OLAP Technology, ACM SIGMOD Record Vol.26 No.1, 1997

W.H. Inmon, Building the Data Warehouse, John Wiley & Sons, 1996

R. Kimball, The Data Warehouse Toolkit, John Wiley & Sons, 1996

IEEE Data Engineering Bulletin Vol.20 No.1, March 1997, Special Issue on Supporting On-Line Analytical Processing, available at http://www.research.microsoft.com/research/db/debull/

IEEE Data Engineering Bulletin Vol.21 No.3, September 1998, Special Issue on Interoperability

IEEE Data Engineering Bulletin Vol.22 No.1, March 1999, Special Issue on Data Transformations

IEEE Data Engineering Bulletin Vol.23 No.4, December 2000, Special Issue on Data Cleaning

IEEE Data Engineering Bulletin Vol.24 No.2, June 2001, Special Issue on XML Data Management

S. Abiteboul, P. Buneman, D. Suciu, Data on the Web: From Relations to Semistructured Data, Morgan Kaufmann, 1999

offizielle Dokumente zu XML, http://www.w3.org/xml

XML Technologies in Oracle8i and Oracle9i, http://technet.oracle.com/tech/xml/