Mediator-basierte heterogene verteilte Informationssysteme

Seminar im Wintersemester 1998/99
Abschlußbericht

Ulf Leser,
Susanne Busse,
Herbert Weber (eds.)

Autoren (alphabetisch):
Ralph Böhmert,
Christian Böhning,
Felix Bübl,
Christoph Holzheuer,
Holger Last,
Christian Möws,
Heiko Müller,
Felix Naumann,
Tom Ritter,
Leon Rosenberg,
Rodrigue Schäfer

Technische Universität Berlin

Inhaltsverzeichnis

Kapitel 1

Einleitung

Die zunehmende Bedeutung von Information als Ressource in Entscheidungsprozessen begründet die Entwicklung von kleinen, zentralisierten Datenbankanwendungen in Richtung der Schaffung großer flexibler und langlebiger Informationsverbünde, die auf einer Vielzahl autonomer Informationssysteme beruhen. Die kontinuierliche Evolution von Fachinhalten und Software-Systemen prägt - aufgrund der Langlebigkeit - die heutige Welt dieser Systeme. Dies umfaßt die Entwicklung neuer Infrastrukturen auf der Basis existierender „Alt-Systeme“, die unabhängige Weiterentwicklung einzelner Komponenten als auch die stetige Erweiterung und Änderung der integrierten Informationsstrukturen.

In dem Seminar “Mediator-basierte heterogene verteilte Informationssysteme” wurden aktuelle Entwicklungen auf dem Gebiet der Integration heterogener Informationssysteme betrachtet. Schwerpunkt der Arbeit lag dabei auf den Informationsaspekten, d.h. der logischen Integration von Datenbeständen unter Berücksichtigung ihrer heterogenen Struktur und Semantik.

In diesem Abschlußbericht soll eine Einordnung der betrachteten Konzepte und Systeme anhand identifizierter Klassifikationskriterien gegeben werden. Die Kriterien ergeben sich dabei aus den charakteristischen Eigenschaften der Autonomie, Heterogenität, Verteiltheit und Evolutionsfähigkeit, die von modernen Informationssystemen gefordert werden.

Abschließendes Ziel wird es außerdem sein, die - sehr uneinheitliche - Terminologie in diesem Bereich zu klären. Was verstehen wir z. B. unter einem Mediator-basierten Informationssystem (MBIS)? Was genau ist ein Mediator? Wie steht es in Bezug zu bekannten Begriffen der Datenbankwelt, wie föderierten Datenbanksystemen?

Zunächst werden wir die oben genannten vier wichtigsten Aspekte föderierter Informationssysteme näher betrachten. Abschnitt 2 diskutiert darauf aufbauend allgemeine Architekturen und Komponenten. Kapitel 3 beschreibt spezifische Kriterien zur Klassifikation und Beschreibung von MBIS, gefolgt von einer Darstellung von notwendigem “Handwerkszeug” für die Entwicklung von konkreten Mediatoren. Dazu zählen besonders ausdrucksstarke (und besonders ausdrucksschwache!) Datenmodelle, Methoden der Schemaintegration sowie klassische Anfrageoptimierungstechniken. In Kapitel 5 stellen wir schließlich fünf Systeme konkret vor und charakterisieren sie in bezug auf die vorab beschriebenen Kriterien aus den vorangegangenen Kapiteln. Kapitel 6 nimmt eine Abgrenzung von MBIS zu anderen Architekturen zur Informationsintegration vor. Kapitel 7 schließlich versucht, das gesamte Gebiet der “Mediation” genauer zu definieren.

1.1 Der rote Faden

Wir betrachten Systeme, die einem Benutzer den Zugang zu mehreren Informationsquellen ermöglichen. Probleme treten dabei besonders auf wegen der Autonomie (Abschnitt 1.2) und Verteilung (Abschnitt 1.4) der Quellen sowie wegen der vielfältigen Heterogenität (Abschnitt 1.3) sowohl zwischen Quellen als auch zwischen Quellen und der integrierenden Schicht. Darüber hinaus muß man auch beachten, daß sich Quellen aufgrund ihrerer Autonomie ohne Vorwarnung verändern können (Abschnitt 1.5). Insbesondere die Autonomie und die daraus entstehende Heterogenität unterscheiden solche Systeme von verteilten Datenbanken (siehe Abschnitt 6.2). Im Unterschied zu Data Warehouses (siehe Abschnitt 6.1) werden außerdem Daten nicht vollständig kopiert (materialisiert), sondern immer nur die für eine konkrete Anfrage benötigten Daten kopiert.

Es gibt mehrere architekturelle Ansätze, eine solche integrierte Schicht zu erreichen. Sie reichen von der reinen Bereitstellung einer gemeinsamen Anfragesprache (siehe Abschnitt 2.2.2) bis zu mehrstufigen Schematransformations-Architekturen (siehe Abschnitt 2.2.3). Soll dem Benutzer ein einheitliches Schema für seine Anfragen zur Verfügung gestellt werden, so stellt sich die Frage, wie man dieses erhält (Schemaintegration - siehe Abschnitt 4.1) sowie welches Datenmodell man dafür verwendet (z.b. semantisch reichhaltige Description Logic, siehe Abschnitt 4.2 oder semistrukturierte Modelle, siehe Abschnitt 4.4).

Die im Seminar näher betrachteten Systeme bieten meist ein solches globales Schema an. Aufgabe eines Systems ist dann die Zerlegung von Anfragen an diese Schema in geeignete Subqueries, die Auswahl von Quellen für jede Subquery, die Übersetzung der Subqueries in Ausdrücke, die von den gewählten Quellen verstanden werden kann, das Abschicken der übersetzten Anfragen sowie das Sammeln und Integrieren der Resultate. Innerhalb dieses groben Rahmens gibt es noch vielfältige Freiheitsgrade, wie z.B.:

- Wird pro Subquery nur eine Quelle gewählt (wie in SIMS, 5.4) oder werden alle möglichen bemüht (wie im Information Manifold, 5.2) ?

- Werden Anfrage-Operatoren grundsätzlich zentral ausgeführt oder versucht, sie zu verteilen (wie in GARLIC, 5.1) ?

- Werden Daten tatsächlich gesammelt oder nur Verweise (wie in MeBro, 5.3) ?

- Werden Quellen mit inhaltsbezogenen Metadaten (siehe Abschnitt 4.3) beschrieben (wie in MeBro, 5.3), oder mit Anfragebeschreibungen (wie im Information Manifold, 5.2), oder mit logischen Ausdrücken (wie in SIMS, 5.4) ?

- Auf welchen Mechanismen beruht die Anfragezerlegung: auf klassischen View-Mechanismen (wie in 4.5), auf logischer Subsumption (wie in SIMS, 5.4), auf einfachen Indizes (wie in MeBro, 5.3), auf speziellen Sprachen (wie in TSIMMIS, 5.5), oder auf semantischer Anfrageoptimierung (wie im Information Manifold, 5.2) ?

Ausgangspunkt sind in allen Systeme a-priori bekannte Quellen, die für das System entsprechend beschrieben werden. Dynamische Quellenfindung und -aufbereitung, wie in 2.1 vorgesehen, findet nirgendwo statt. Genauso verfügt keines der Systeme über mächtige Mechanismen zur echten Integration von Ergebnissen, z.B. zum Erkennen gleicher Real-Life Objekte in verschiedenen Repräsentationsformen.

1.2 Autonomie

Die Autonomie der Komponentensysteme besagt ganz allgemein: Die Systeme sind unabhängig voneinander entstanden, werden unabhängig voneinander betrieben und weiterentwickelt. Die Teilnahme eines Komponentensystems an einer Föderation darf nicht die Arbeit existierender lokaler Anwendungen beeinträchtigen. Im Idealfall braucht das Komponentensystem keine Kenntnis von bzw. keinen Extraaufwand durch seine Teilnahme an einer Föderation zu haben. Man unterscheidet vier Arten von Autonomie:

- **Entwurfsautonomie:** Die einzelnen Komponenten sind unabhängig voneinander entworfen worden. Das beinhaltet sowohl den Diskursbereich, als auch das Datenmodell und die Namensgebung. Die Funktionalität der Systeme (die bereitgestellten Operationen) kann verschieden sein. Auch die Semantik ist oft spezifisch und nicht auf andere Anwendungen übertragbar (was dann aber mehr unter semantische Heterogenität fällt). Bleibt bei einer Föderation dieser Entwurf unberührt, so ist die Entwurfsautonomie gegeben.

  Ein großes Problem bei der Entwurfsautonomie ist die nachträgliche Änderung der lokalen Schemata. Streng genommen müßte sich die Autonomie darauf erstrecken, in der Praxis jedoch ist der damit verbundene Integrationsaufwand als zu hoch einzuschätzen. Es gibt auch keine Techniken um diesen Vorgang zu unterstützen.

- **Kommunikationsautonomie:** Bei dieser Autonomieart geht es darum, daß jedes Komponentensystem selbst entscheiden kann mit welchen anderen Komponentensystemen es kommuniziert (gemeint ist eine Entscheidung durch den Administrator). Besonders in der Import - Export Architektur (seihe Kapitel 2.2) spielt diese Autonomie eine große Rolle, da dort die Komponentensysteme über die gegenseitige Kommunikation verhandeln.

- **Ausführungsautonomie:** Jedes Komponentensystem hat einen eigenen Scheduler und kann in Folge dessen selbst darüber bestimmen, wann welche Anfragen bearbeitet werden. Globale Operationen haben demnach keinen Vorrang vor den lokalen, sondern werden (im günstigen Fall) als gleichwertig angesehen. Das Komponentensystem muß nicht einmal das FDBS davon unterrichten, wann und wie die Operation ausgeführt wird. Die Ausführung kann sogar aus bestimmten Gründen, wie zum Beispiel lokale Constraints, abgelehnt werden.

  Die Wahrung der Ausführungsautonomie wirft natürlich neue Probleme für die globale Transaktionsverwaltung auf. Deswegen ist es in föderierten DBS Brauch, daß die Komponentensysteme den Föderierungsdienst zumindest davon unterrichten, in welcher Reihenfolge die Ausführung der Operationen stattfinden wird, um eine effizientere und einfachere globale Transaktionsverwaltung zu ermöglichen. Bei Informationssystemen mit nur lesenden Zugriffen ist diese Autonomie entsprechend leichter zu gewährleisten.

- **Assoziationsautonomie:** - die manchmal zu der Kommunikationsautonomie bzw. der Entwurfsautonomie gezählt wird - stellt die Freiheit des Komponentensystems dar, selbst entscheiden zu können, ob und wann es an der Föderation teilnimmt, und seine Daten und Funktionalität mit anderen teilt. In der Praxis wird dies jedoch auf der Administrationsebene entschieden, denn wenn jedes Komponentensystem nur "Andere" nutzen will, und selbst aber für diese "Anderen" keine Leistungen verbringen will, wird es keine Kooperation und in Folge auch keine Föderation geben.

Es ist klar, daß der Erhalt vollständiger Autonomie bei einem Zusammenschluß von Systemen kaum möglich ist (z.B. werden manche lokale Änderungen an andere Partner propagiert werden müssen). Ziel ist es deshalb, die Autonomie so weit wie möglich zu erhalten, wobei die konkreten Anforderungen und Gegebenheiten die Wichtigkeit einzelner Autonomieaspekte bestimmen.

1.3 Heterogenität

Aufgrund der autonomen Entwicklung der Komponentensysteme treffen wir bei deren Integration auf die verschiedensten Formen der Heterogenität - von unterschiedlicher Hardware über verschiedene Datenmodelle bis hin zu differierendem Verständnis über die Semantik der enthaltenen Daten. Es existiert eine Vielzahl verschiedener Klassifikationen von Heterogenität. Eine mögliche wird in diesem Abschnitt vorgestellt. Im Vordergrund der Betrachtungen steht das relationale Datenmodell, anhand dessen die große Spanne an Konflikten besonders im Bereich der semantischen Heterogenität verdeutlicht werden soll.

1.3.1 Syntaktische Heterogenität

Technische Heterogenität

Unter technischer Heterogenität weren unterschiedliche Hardwareplattformen, Protokolle und Zugriffsmethoden zusammengefaßt. Die eingesetzte physikalische Hardware wird dabei durch die Wahl des Betriebssystems (Unix und Derivate, Windows, ...) und vor allem durch die Netzinfrastruktur und der darauf aufsetzenden Kommunikationsprotokolle (TCP/IP, OSI/LAN, ...) gekapselt.

Zum Zugriff auf und für die Kommunikation von Datenbanken untereinander existieren verschiedenste Methoden. Ein Ansatz ist die Verwendung von Middleware-Standards wie z.B. CORBA, ActiveX oder OLE/COM. Im Bereich der Windows-orientierten Betriebssystemen hat sich der ODBC-Standard herausgebildet; über das Internet wird weitestgehend mit dem HTTP-Protokoll kommuniziert. Aufgrund der großen Verbreitung von TCP/IP und dem HTTP Protokoll sind viele Datenquellen über diese Lösung technisch erreichbar.

Weitere systemspezifische Unterschiede gibt es in der Realisierung von Konzepten von Datenbankmanagementsystemen, die sich beispielsweise in unterschiedlichen Transaktionsverwaltungsparadigmen hinsichtlich Sperr- und Commit-Protokollen sowie in verschiedenen Recovery-Verfahren äußern können.

Schnittstellenheterogenität

Jedes Datenmodell und auch Datenbanksysteme, die mit dem gleichen Datenmodell arbeiten, können verschiedene Anfragesprachen zur Erzeugung, Speicherung und der Manipulation von Daten verwenden. Retrievalsysteme beispielsweise kapseln große Datenmengen durch semi-standardisierte

Schnittstellen; Suchmaschinen mit heterogenen Benutzeroberflächen sind momentan die einzige Möglichkeit, effizient im Datenbestand der durch das Internet zugänglichen Informationsspeicher zu recherchieren.

Viele Schnittstellen werden zwar mit Standardanfragesprachen beschrieben, wie z.B. SQL, erlauben aber nur eine eingeschränkte Verwendung. Beispielsweise begrenzen fast alle Datenbanksysteme die maximale Anzahl von Joins; Web-Interfaces erlauben nur bestimmte Selektionen und keine Projektionen; und oft trifft man auf sogenannte *Binding Pattern*, d.h. Anfragen, bei denen z.B. bestimmte Werte spezifiziert sein müssen.

1.3.2 Logische Heterogenität

Datenmodellheterogenität

Die Daten können in unterschiedlichen Systemen in unterschiedlichen Datenmodellen vorliegen. Die am häufigsten zur Modellierung von Daten verwendeten Datenmodelle sind relationale Datenmodelle, Netzwerkdatenmodelle oder objektorientierte Datenmodelle. Jedes Datenmodell besitzt eigene Modellierungskonstrukte, mit denen Ausschnitte der realen Welt abgebildet werden können. Bezogen auf einen gleichen Ausschnitt kommt es so zu einer unterschiedlichen Modellierung der Daten, die sogar bei gleichem Datenmodell aufgrund unterschiedlicher Verwendung der Modellierungskonstrukte differieren kann.

Weiterhin muß unter dem Aspekt der Verwendung unterschiedlicher Datenmodelle auch die Implementierung von Integrationsbedingungen berücksichtigt werden, die u.a. durch geeignete Modellierung implizit im Datenmodell enthalten sein können, während sie an anderer Stelle explizit angegeben werden müssen.

Semantische Heterogenität

Unterschiede in Daten und Schema kann man unter semantischer Heterogenität zusammenfassen. Hierzu gehören Unterschiede hinsichtlich des abgedeckten Diskursbereichs (Extensionsmengen) sowie Beschreibungskonflikte wie verschiedene Benennungen, Wertebereiche oder Integritätsbedingungen.

- **Tabellennamenkonflikte / Attributnamenkonflikte:** Tabellennamen werden i.d.R. durch die Semantik der Entitäten bestimmt, die in ihnen gespeichert werden. Der Vorgang der Benennung der Tabellen kann von mehreren Parametern wie z.B. der verwendeten Entwurfsmethode, den Richtlinien des Einsatzgebietes und dem Administrator abhängen. Dabei kommt es zu Namenskonflikten bei gleichen Tabellen (Attributen) mit verschiedenem Namen (Synonyme) bzw. gleichen Namen (Homonyme) für verschiedene Tabellen (Attribute). Dieser Konflikt kann auch durch unterschiedliche Benennung von Klassen (Categorisation mismatch [48]) auftreten.

  Die vollständige Schema-Übereinstimmung (gleiche Tabellen- und Attributnamen) bei gleichzeitig unterschiedlicher Semantik stellt einen Sonderfall der Homonym-Konflikte dar (Concept mismatch [48]), dem nur unter Einbeziehung von Modellierungsaspekten und mit Interpretation der Attributwerte (Daten) erkannt und behoben werden kann. Diese Vorgehensweise muß ebenfalls angewendet werden, wenn die Schemata völlig unterschiedlich (verschiedene Tabellen- und Attributnamen) sind, da die Möglichkeit besteht, daß der gleiche Diskursbereich abgebildet wird (Term and definiens mismatch).

- **Tabellenstrukturkonflikte:** Die Attribute einer Tabelle definieren die Struktur. Auf dieser Ebene können Konflikte durch fehlende Attribute oder fehlende, aber implizierte Attribute auftreten [8]. Beide Konflikte können auch gleichzeitig auftreten. Identisch sind dann der Tabellenname und der zu modellierende Diskursbereich, während die Attribute teilweise fehlen bzw. im anderen Schema implizit vorhanden bzw. anders benannt sind (Definiens mismatch [48]).

- **Integritätsbedingungskonflikte bei Tabellen:** Integritätsbedingungen können die darstellbare Informationsmenge beschränken. Konflikte treten auf, wenn unbeschränkte (n:m) Relationen auf durch Integritätsbedingungen eingeschränkte (1:1, 1:n) Relationen abgebildet werden sollen.

- **Integritätsbedingungskonflikte bei Attributen:** Die Datentypdefinition legt den Wertebereich von Attributen fest (vgl. Integritätsbedingungen bei Tabellen). Konflikte entstehen, wenn die Typdefinition für semantisch äquivalente Attribute unterschiedlich ist - wie z.B. bei der Abbildung großer Wertebereiche auf kleinere oder bei unterschiedliche Datentypen ([8],[21]).

- **Default-Wert-Konflikte:** Zu jedem Attribut können Werte definiert sein, die bei einer Initialisierung bzw. Neueingabe von Daten vorgegeben werden. Bei gleichen Attributen können unterschiedliche Default-Werte vorgegeben sein.

- **Falsche oder veraltete Daten:** Wenn ein Objekt aus der realen Welt in unterschiedlichen Datenbeständen repräsentiert ist, kann es zu falschen Daten kommen, die nicht durch Integritätsbedingungen sichergestellt werden können und deshalb schwer zu entdecken sind. Bei der Integration von Datenquellen, die ohne modellierten Zeitstempel arbeiten, kommt es zu Konflikten zwischen veralteten und aktuellen Einträgen.

- **Unterschiedliche Repräsentationen:** Verschiedene Ausdrücke oder Schreibweisen für die gleiche Information stellen ebenfalls ein Problem dar. Im Gegensatz zu Synonymen handelt es sich in diesen Fällen jedoch nur um abweichende, nicht jedoch völlig verschiedene Repräsentationen. Abweichungen können sich auf ein bestimmten Attributwert oder auf eine Menge von Attributwerten beziehen. Unter Umständen kann es sich - wie bei implizierten Attributen - um implizierte Teile eines Attributwertes handeln [21].

Ebenfalls zu Konflikten kommt es bei der unterschiedlichen Modellierung und Anwendung von Einheiten und der Verwendung von unterschiedlichen Genauigkeiten (Skalen) für gleiche Sachverhalte.

1.3.3 Strukturelle Heterogenität

Wenn der Diskursbereich mit Hilfe eines Datenmodells, aber unter Verwendung verschiedener Modellierungskonzepte; z.B. Attribut vs. Assoziation abgebildet wird, tritt strukturelle Heterogenität auf. Folgende Konfliktklassen können unterschieden werden:

Tabellen-Attribut-Konflikte

Semantisch äquivalente Informationen können in unterschiedlich normalisierten Relationen vorliegen (Integration von Tabellen) bzw. jeweils als Attribut oder Tabelle modelliert worden sein (Integration von Tabellen und Attributen).

Strukturkonflikte

Besonders, nicht nur in der objektorientierten Modellierung von Diskursbereichen, treten Konflikte in der hierarchischen Beziehung zweier Klassen auf. Zwei gleiche Klassen können beispielsweise durch semantisch unterschiedliche Relationen verbunden sein und damit unterschiedliche Diskursbereiche abbilden (Structure mismatch [48]). Ein weiteres Problem ist die korrekte Assoziation einer Attributklasse zu einer Klasse (Attribute assignment mismatch [48]).

Abstraktionslevel-Konflikte

Durch unterschiedliche Anwendung von Modellierungskonstrukten wie Generalisierung und Aggregation kommt es zu Konflikten, deren Spezialfälle unter Schema- und semantischer Heterogenität verdeutlicht werden.

Schematische Heterogenität

Die schematische Heterogenität bildet eine besondere Form der logischen Heterogenität. Sie tritt auf, wenn innerhalb einer Komponente Daten modelliert werden, die in einer anderen in Form eines Schemas repräsentiert sind.

Beispielsweise können gleiche Sachverhalte durch verschieden granulare Aufteilung in Klassen abgebildet werden (Aggregation-level mismatches [48]). Die Daten einer Klasse mit höherer Granularität können in einem anderen Fall als Schema modelliert worden sein.

Im relationalen Datenmodell treten diese Konflikte bei der Modellierung des gleichen Sachverhaltes einmal als Attribut und einmal als Attributwert [8]. Auch hier kommt es zu einer Vermischung von Schema und Daten.

1.3.4 Zusammenfassung

Um Daten zu integrieren, sind eine Vielzahl von Heterogenitätsproblemen zu lösen. Technische Konflikte werden durch zunehmende Standardisierung immer weniger eine Rolle spielen. Auch im Bereich der strukturellen Hetetogenität und der Datenmodellheterogenität sind in der letzten Zeit Lösungen entwickelt worden, mit denen man diesen Konflikten durch die Implementierung entsprechender Schnittstellen begegnen kann. So ist es beispielsweise möglich, relationale Datenmodelle in objektorientierte Datenmodelle zu überführen und umgekehrt. Die unterschiedliche semantische Ausdruckskraft dieser Modelle muß dabei durch einen Schritt zur semantischen Anreicherung bzw. durch die Inkaufnahme eines Verlustes von Semantik ausgeglichen werden.

Im Vordergrund stehen deshalb die vielfältigen semantischen Heterogenitätsprobleme, deren Lösung maßgeblich den Erfolg einer Integration von Daten bestimmt. Ein wichtiger Faktor bei der Behandlung von semantischen Konflikten ist die Berücksichtigung der Bedeutung der Daten, die in Form des Begriffes "Concept" besonders in [48] herausgearbeitet wurde. Die Intention des

Modellierers bestimmt in erster Linie die Verwendung von Namen für Tabellen und Attribute. Die unter dem Begriff semantische Heterogenität geschilderten Konfliktklassen behandeln zum größten Teil die Diskrepanz zwischen dem Definierten und dem Definierenden in verschiedenen Schemata.

Interessante Grenzfälle treten auf, wenn völlige Gleichheit in der Modellierung auftritt, aber etwas anderes "gemeint" ist. Deshalb wird es immer erforderlich sein, den Kontext und den Zusammenhang, in denen modellierte Daten stehen, zu betrachten. Informationen über die Modellierung, den Kontext und die Bedeutung von Daten sind jedoch in den meisten Fällen nicht oder nur in geringem Maße vorhanden. Einige in diesem Papier geschilderten Ansätze werden sich mit der Gewinnung und Speicherung von zur Integration benötigten Metainformationen intensiver auseinandersetzen.

Bezogen auf die Lösungsmöglichkeit von Konflikten kann festgestellt werden, daß einzelne Konfliktklassen zwar in einem beschränkten Umfang automatisch aufgelöst werden können (z.B. durch einen Thesaurus-Abgleich); sofern jedoch Wissen über die Bedeutung der Daten erforderlich ist, ist eine manuelle Bearbeitung zwingend erforderlich.

1.4 Verteilung

Da wir beim Aufbau großer Informationssysteme in der Regel existierende Informationssysteme berücksichtigen müssen, in denen die Daten originär erfaßt und verwaltet werden, entsteht automatisch ein verteiltes System. Die Verteilung wird in vielen Fällen zusätzlich durch die Organisationsstrukturen vorgegeben, die sich aus den historisch gewachsenen Kompetenzbereichen ergibt.

Jedes Komponentensystem kann intern wiederum verteilt organisiert sein, wobei verschiedene Gliederungen möglich sind (vgl. [9], [38]). Die interne Verteilung ist jedoch für das übergreifende Informationssystem irrelevant, da die Komponentensysteme als gekapselte Einheiten mit definierter Schnittstelle betrachtet werden.

1.5 Evolutionsfähigkeit

Jedes langlebige Softwaresystem unterliegt einer kontinuierlichen Evolution. Bereits bei der Entwicklung sollte deshalb eine hohe Evolutionsfähigkeit, d.h. leichte Änderbarkeit des Systems, eine zentrale Zielrichtung sein.

Bei der hier betrachteten Klasse von Systemen – unter besonderer Berücksichtigung der logischen Aspekte – sollen dabei die folgenden Arten von Änderungen zur Diskussion herangezogen werden:

- Logische Änderungen
  - Änderung eines Komponenten-Schemas
  - Lokale Erweiterung eines Komponenten-Schemas
  - Erweiterung eines Komponenten-Schemas mit Auswirkung auf den Informationsverbund
  - Änderung des globalen Modells (falls existent) aufgrund neuer Informationsbedürfnisse

- Erweiterung des globalen Modells (falls existent) aufgrund neuer Informationsbedürfnisse

- Konfigurations-Änderungen
  - Änderung der Zahl der integrierten Informationssysteme
  - Änderung des Orts eines Komponentensystems

- Systemtechnische Evolution
  - Lokale Änderung einer Komponente (z.B. Änderung der Anfragemöglichkeiten durch Austausch von DBS oder anderer interner Komponenten)
  - Erweiterung der Funktionalität des übergreifenden Informationssystems (muß intern klare def. Schnittstellen haben)
  - Integration des entwickelten Informationssystems in einen neuen größeren Informationsverbund (Jedes System von heute kann eine Komponente von morgen sein!)

- **Syntaktische Heterogenität**
  - Technische Heterogenität
    - Unterschiedliche Hardware
    - Unterschiedliche Protokolle
    - Unterschiedliche technische Zugriffsmethoden
  - Schnittstellenheterogenität
    - Unterschiedliche Anfragesprachen
    - Unterschiedliche logische Zugriffsmethoden
- **Logische Heterogenität**
  - Datenmodellheterogenität
    - Unterschiedliche Datenmodelle
    - Unterschiedliche Modellierungskonstrukte
  - Semantische Heterogenität
    - Tabellennamen, Attributnamen, und Klassennamenkonflikte
    - Tabellenstrukturkonflikte (fehlende oder implizierte Attribute)
    - Integritätsbedingungskonflikte bei Tabellen und Attributen
    - Default-Wert-Konflikte bei Attributen
    - Falsche oder veraltete Attributwerte (Daten)
    - Unterschiedliche Repräsentationen (Ausdrücke, Einheit, Genauigkeit)
  - Strukturelle Heterogenität
    - Tabellen-Attribut-Konflikte
      - Strukturkonflikte (Relation zwischen Klassen)
      - Abstraktionslevel-Konflikte (Generalisierung, Aggregation)
    - Schematische Heterogenität
      - Aggregation-Level Mismatches
      - Attribut-Attributwert Konflikte

Tabelle 1.1. Tabelle: Zuordnung von Konflikten zu Konfliktklassen

Kapitel 2

Architektur Mediator-basierter Informationssysteme

Mediator–basierte Informationssysteme (MBIS) sollen dem Nutzer den Zugang zu einer Kombination von Informationen aus verschiedenen Quellen bieten. Deren Autonomie und Heterogenität zueinander stellen dabei die zu lösenden Herausforderungen dar. Bisherige Lösungsansätze weisen einen Trend zu einer 3–Layer–Architektur auf, wie sie Fig. 2.1 schematisch zeigt.

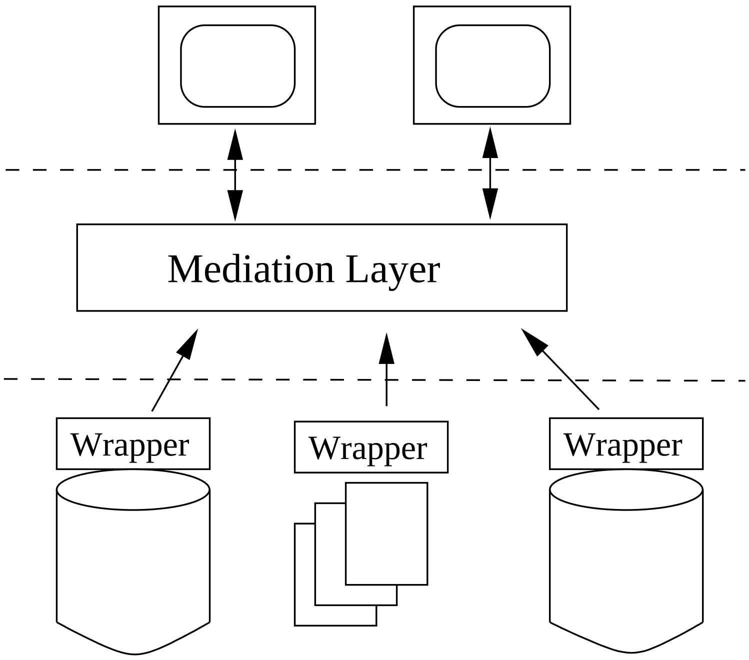

Abbildung 2.1. 3–Layer–Architecture

Die Basis des Informationssystems bilden Datenquellen, die mit Hilfe eines jeweils spezifischen Wrappers für Softwaremodule des Mediation–Layers zugänglich gemacht werden. Wrapper adressieren damit explizit Probleme der Schnittstellen- und Datenmodellheterogenität. Sie sollen den

Zugriff auf eine Datenquelle mit Hilfe einer neuen nach außen sichtbaren Schnittstelle unabhängig vom Datenmodell und Anfragemechanismus der Quelle ermöglichen. Es ist zu erwarten, daß die Wrapper nach oben zum Mediator eine einheitliche Schnittstelle anbieten.

Konzeptionell unabhängig davon soll der Mediation–Layer die Integration und Anreicherung der Daten leisten. Hier sollen Mediatoren die Probleme der logischen Heterogenität weitgehend lösen, damit dem Nutzer genau die gewünschten Informationen redundanzfrei in der Anwendungsschicht präsentiert werden können. Nach [51] ist Mediation

> „an activity which reduces data to information by applying knowledge about resources, search strategies, and user requirements“

Der Mediator (oder kollaborierende Mediatoren) muß die Integration leisten, die bis heute noch zumeist von den Nutzern erbracht werden muß. Der Grad der Integration, den ein Mediator leisten kann, variiert über die Datenquellen und Anforderungen der darauf aufbauenden Anwendung. Spezielle Probleme sind in Kapitel 4 detaillierter dargestellt.

Der Application–Layer hat zwei wichtige Aufgaben. Dort wird die Benutzerschnittstelle zum MBIS geschaffen, die

- die Spezifikation der Anfrage und
- die Präsentation der Ergebnisse

in einer benutzerfreundlichen Form leisten muß. Inwieweit diese Schicht darüber hinaus Anwendungslogik enthält, ist den spezifischen Anforderungen überlassen.

2.1 I^3–Referenzarchitektur

Die Probleme und Anforderungen an Mediatoren sind vielfältig, und es erscheint sinnvoll, eine Strukturierung des Diskursbereiches zu haben, die dem Entwickler eines MBIS hilft. Dieses Ziel verfolgt die I^3–Referenzarchitektur[1] mit einer grundlegenden und umfassenden Klassifikation der notwendigen Funktionen für ein solches Informationssystem. Dabei geht es nur um abstrakte Darstellungen und nicht um implementierungsspezifische Details.

> „A reference architecture model describes a system in terms of the interconnection of basic functional elements and the interfaces between them. It clarifies where protocols must be defined and identifies groupings of functionality. It does not imply a physical implementation“[19]

Diese Referenzarchitektur adressiert genau die im Kapitel 1 aufgezeigten Probleme:

- Autonomie von Informationsquellen
- Heterogenität von Informationsquellen
- Größe und Verteiltheit der Systeme
- Evolution von Informationsquellen

[1] I^3 = Intelligent Integration of Information

Man geht davon aus, daß mit dieser Referenzarchitektur die theoretische Grundlage für wiederverwendbare Komponenten geschaffen werden kann. Diese Komponenten werden I^3–Services genannt. In dieser Architektur wird auf das Modell der Dienste gesetzt, die einzeln relativ klein und handhabbar sind und durch eine sinnvolle Kombination komplexere Aufgaben erledigen können.

Fig. 2.2 spiegelt den Gesamtrahmen der Dienste eingeteilt in vier Familien wider. Dieser komponentenbasierte Ansatz in Verbindung mit dem Konzept der Services eröffnet sicher auch den Rahmen, mit Evolution an den verschiedenen Stellen eines MBIS umgehen zu können.

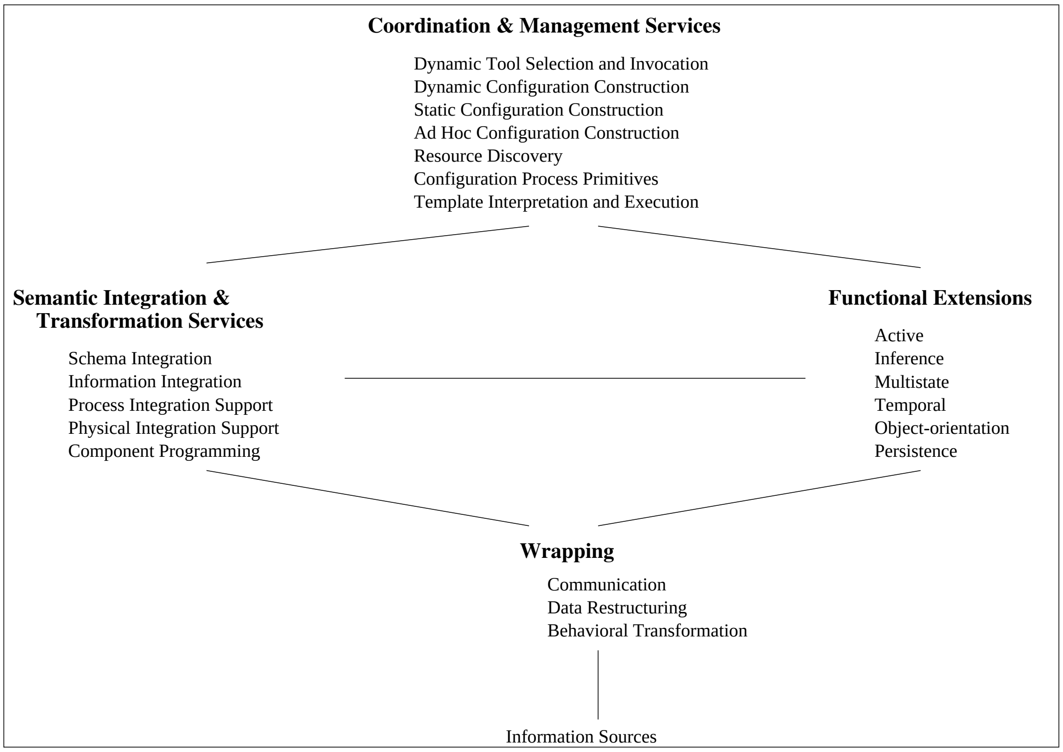

Abbildung 2.2. I^3–Klassifikation von Diensten

Die Datenquellen werden in dieser Architektur nicht betrachtet. Sie sollen, so wie sie bestehen, gekapselt und benutzt werden. Es werden keine Anforderungen an sie gestellt. Damit bleibt jede Autonomie der Informationsquellen unberührt.

2.1.1 Coordination & Management Services

Diese „Familie“ von Diensten unterstützt die Erstellung von I^3–Konfigurationen für eine spezifische Aufgabe. Dazu gehört es, geeignete Komponenten zu lokalisieren und Informationsquellen zu finden, die zur Lösung beitragen. Es gehören Services dazu, die diese Komponenten starten und steuern, sowie das Scheduling übernehmen. Hier sind insbesondere die dynamischen Aspekte (Brokering[2] und Facilitation[3]) eines mediator–basierten Informationssystems angesiedelt.

[2] Dynamic Tool Selection and Invocation
[3] Dynamic Configuration Construction

2.1.2 Semantic Integration & Transformation Services

Diese Klasse von I^3–Services unterstützt die notwendigen semantischen Manipulationen zur Integration und Transformation von Informationen zur Lösung einer Aufgabe und sollen dem Entwickler entsprechender Anwendungen die Integration mehrerer, heterogener Datenquellen in eine „Integrationsbasis“ erlauben. Die primären Services erfüllen Dienste zur Schema- und Datenintegration (vgl. Kapitel 4). In dieser Referenzarchitektur wird damit ein sehr weitgehender Ansatz verfolgt. Die Mediatoren sollen hier die Integration vollständig durchführen und an die Coordination & Management Services integrierte Ergebnisse liefern.

2.1.3 Functional Extension Services

Die Functional Extension Services sollen Dienste bereitstellen, die die semantische Ausdrucksfähigkeit von Informationsquellen verstärken. Diese Services dienen insbesondere dazu, die Funktionalität der anderen I^3–Services zu erweitern.

2.1.4 Wrapping Services

Die Wrapping Services bilden die Basis in der Klassifikation der Dienste der I^3–Referenzarchitektur. Es ist notwendig, Transformationen von Kommunikationsschnittstellen, Datenstrukturen und Programmlogiken vorzunehmen. Die Wrapping Services sollen technische Inkompatibilitäten auflösen und idealerweise eine Menge von wiederverwendbaren, gekapselten Informationsquellen schaffen, die über weithin akzeptierte Standards zugänglich sind.

2.2 Föderierte Datenbanksysteme

Ein Datenbanksystem kann zentralisiert oder verteilt sein[4]. Während das zentralisierte Datenbanksystem aus einem Datenbankverwaltungssystem und einer Datenbank, die auf einem Computer laufen besteht, handelt es sich beim verteilten Datenbanksystem um ein Datenbankverwaltungssystem, das mehrere Datenbanken verwaltet. Multidatenbanksysteme bestehen dagegen aus mehreren Komponentendatenbanksystemen die von jeweils eigenen und möglicherweise verschiedenen Datenbankverwaltungssystemen verwaltet werden. Ein Komponentendatenbanksystem in einem Multidatenbanksystem kann sowohl zentralisiert, als auch verteilt sein. Haben alle Komponenten das gleiche Datenbankverwaltungssystem, so nennen wir eine solche Multidatenbank homogen, ansonsten ist sie heterogen. Systeme, die einen periodischen, nichttransaktionalen Datenaustausch zwischen mehreren Datenbanksystemen erlauben oder jene, welche Zugriff nur zu einem Datenbankverwaltungssystem gleichzeitig erlauben, gehören nicht zu den Multidatenbanksystemen.

Die Klassifizierung der Föderationsarten hängt im wesentlichen vom Grad der Autonomie ab, die sie den Komponentendatenbanksystemen bieten. Sind die Komponenten eines MDBS autonom, so handelt es sich dabei um ein föderiertes Datenbanksystem, wobei der Grad der Autonomie die Unterscheidung zwischen den verschiedenen Arten der Föderation unterscheidet (Siehe Fig. 2.3).

[4] Die Darstellung dieses Kapitels lehnt sich eng an [8] an.

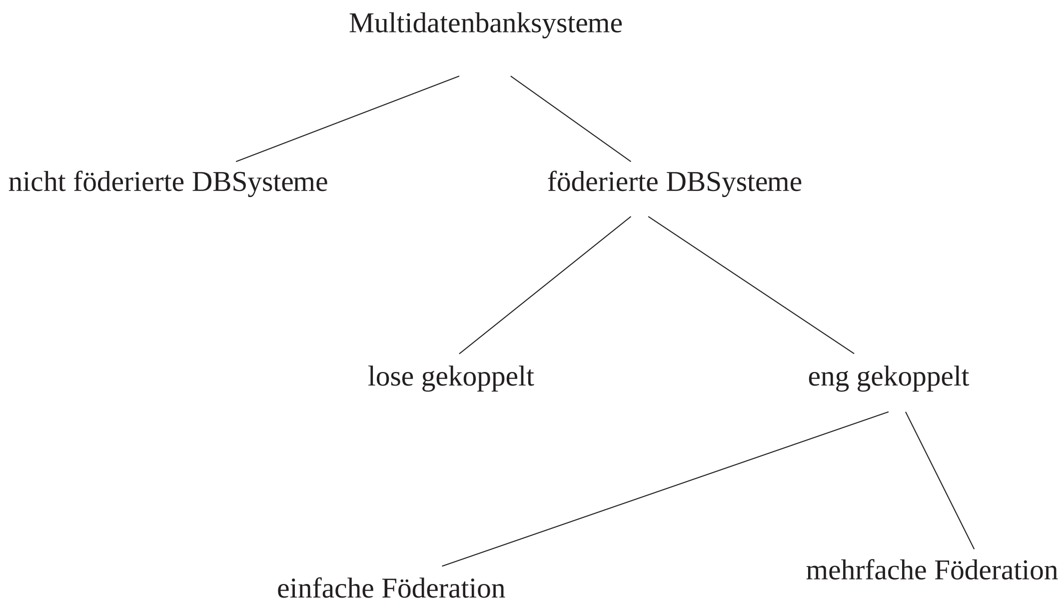

Abbildung 2.3. Klassifikation von Multidatenbanksystemen ([41])

In erster Linie unterscheidet man zwischen lose und eng gekoppelten föderierten Datenbanksystemen. Bei dem ersten bestimmt der Benutzer selbst, welche Komponenten an einer Föderation teilnehmen; jeder Benutzer stellt sozusagen seine eigene Föderation zusammen und verwaltet sie. Entsprechend hat er sich um die Integration zu kümmern. Ein typisches Beispiel für die lose Kopplung ist die Multidatenbankarchitektur (siehe unten).

Bei den eng gekoppelten föderierten Datenbanksystemen wird die Föderation von einem Administrator bzw. einer Gruppe von Administratoren (in Zusammenarbeit mit den Administratoren der Komponentendatenbanksysteme) zusammengestellt. Dieser Administrator legt ein globales Schema fest, das von allen globalen Anwendungsprogrammen und Benutzern zum Zugriff auf die Föderation benutzt wird. Ebenfalls ist die Integration der Komponenten in die Föderation seine Aufgabe. Dies macht den Benutzern das Leben ungemein leichter. Ein typischer Vertreter hierfür ist die 5 Ebenen Schema Architektur (siehe Abb.Fig. 2.4). Die Klassifizierung von eng gekoppelten FDBS kann man weiter verfeinern hinsichtlich dessen, ob sie nur eine oder mehrere Föderationen zulassen, bzw. wie viele globale Schemata es gibt. Die mehrfache Föderation ist oft aus Gründen der Rechenleistungsverteilung oder anderen Gründen sinnvoll. Jedoch ist eine mehrfache Föderation meistens nichts anderes als eine Sammlung einfacher Föderationen über demselben Komponentenbestand. Die Unterschiede zwischen den beiden Fällen sind leider aus der Literatur nicht ersichtlich. Einen möglichen Unterschied könnten bestimmte Zusammenhänge zwischen den einzelnen föderierten Schemata innerhalb einer mehrfachen Föderation ausmachen.

Die drei wichtigsten Vertreter der föderierten Datenbanksysteme sind die Import-Export-Schema Architektur, die Multidatenbank Architektur und die 5 Ebenen - Schema Architektur. Während die 5 Ebenen - Schema Architektur enge Kopplung unterstützt, sind die anderen beiden Vertreter der lose gekoppelten Architekturen.

2.2.1 Import/- Export Architektur

Die Import/- Export Schema Architektur zeichnet sich im wesentlichen dadurch aus, daß in jedem Komponentensystem ein Import Schema, das die zu importierenden Komponentensysteme beschreibt, und ein Export Schema, das die für die Föderation freigegebenen Inhalte beschreibt, eingeführt wird. Dabei muß die Föderation stets zwischen zwei Komponenten, die sich gegenseitig nutzen wollen, ausgehandelt werden (jedes Komponentensystem, das auf ein anderes Komponentensystem zugreifen will, muß es mit diesem Komponentensystem explizit aushandeln). Dadurch entsteht ein großer Integrationsaufwand für die Administratoren, der oft mehrmals notwendig wird. Außerdem muß die Integration von "eigenen" und "fremden" Inhalten innerhalb einer Anwendung vorgenommen werden. Dies bleibt dem Benutzer überlassen.

2.2.2 Multidatenbank Architektur

Bei der Multidatenbank Architektur wird eine zusätzliche Schicht geschaffen, die alle Komponentensysteme umfaßt und Zugriffe auf mehrere dieser Systeme erlaubt. Alle Komponentensysteme müssen ihre lokalen Schemata in ein gemeinsames Datenmodell abbilden, das kanonische Modell. Über diese Schemata kann ein Benutzer komponentensystemübergreifende Anfragen stellen. Er kann auch Views definieren, die sich über mehrere Komponentensysteme erstrecken, und mit diesen arbeiten. Für diese Operationen wird eine spezielle Multidatenbanksprache zur Verfügung gestellt, die meistens eine Erweiterung von SQL ist (z.B. MSQL [28] oder SchemaSQL [22]). Für die Beschreibung von globalen Integrationsbedingungen werden Abhängigkeitsschemata definiert.

2.2.3 5 - Ebenen - Schema Architektur

Die 5 - Ebenen - Schema Architektur ist, meiner Ansicht nach, die logische Erweiterung der ANSI / SPARC Architektur zu föderierten Datenbanksystemen und erlaubt enge Kopplung und mehrfache Föderation. Ähnlich wie bei der vorigen Architektur gibt es auch hier 5 Schemata (Abbildung Fig. 2.4):

- **Lokales Schema:** ist das konzeptionelle Schema eines Komponentendatenbanksystems. Oft liegen die lokalen Schemata in unterschiedlichen Datenmodellen vor.

- **Komponentenschema:** ist das in das föderierte Datenmodell (also das Datenmodell das für die föderierte Sicht benutzt wird) transformierte lokale Schema der Komponentendatenbank.

- **Exportschema:** (bzw. die Exportschemata bei der mehrfachen Föderation) - entsprechen dem externen Schema bei der ANSI / SPARC Architektur und beschreiben den Ausschnitt aus dem Komponentenschema, der für die jeweilige Föderation bereitgestellt wird. Es wird im gleichen Datenmodell erstellt wie das Komponentenschema, um zusätzliche Transformationen zu vermeiden.

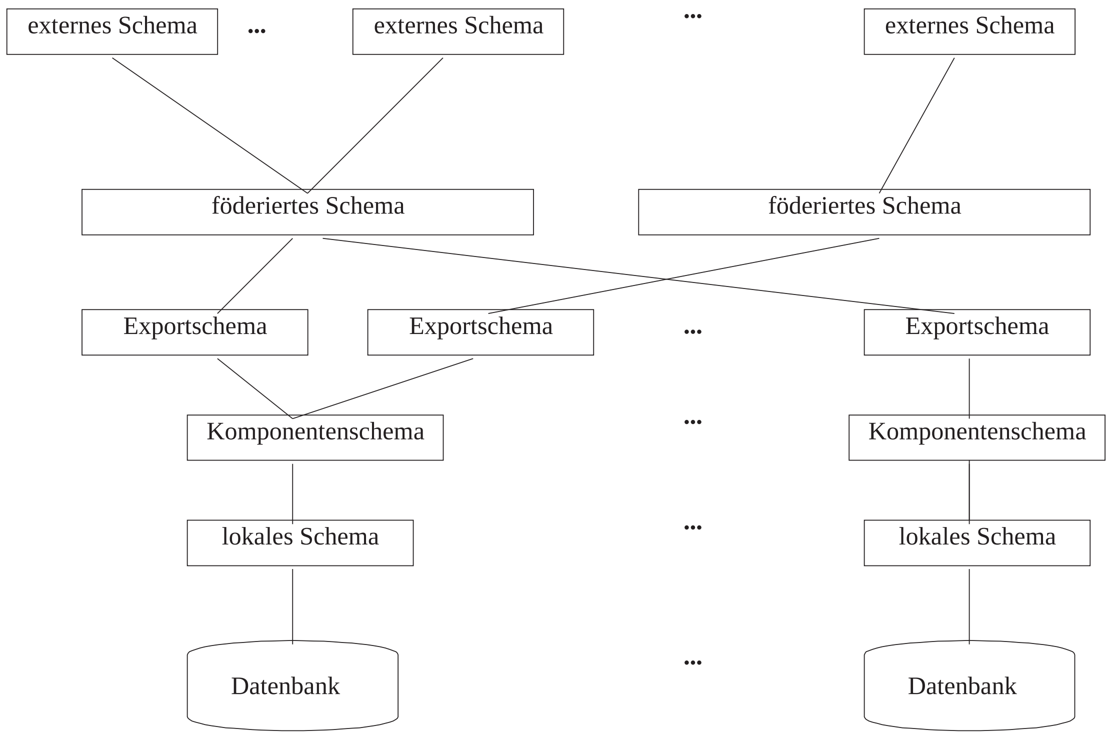

Abbildung 2.4. Fünf-Schichten Architektur (nach [41])

- **Föderiertes Schema:** ist die Zusammenfassung mehrerer verfügbarer Exportschemata (maximal eins von jeder Komponente) zu einem globalen Schema. An dieser Stelle treten die meisten Integrationsprobleme auf. Das globale Datenmodell, das für das föderierte Schema (wie auch für die Export- und Komponentenschemata) verwendet wird, wird auch oft *kanonisches Datenmodell* benannt.

- **Externes Schema:** basierend auf dem föderierten Schema werden für die globalen Anwendungen, die über den Föderierungsdienst auf die Daten zugreifen wollen, externe Schemata definiert. Diese Schemata entsprechen den externen Schemata in der ANSI / SPARC Architektur (allerdings gelten sie für die gesamte Föderation und nicht für ein einzelnes Komponentensystem, wie das beim Exportschema der Fall ist). Hier werden die klassischen Aufgaben der externen Schemata wahrgenommen, z.B. die Zugriffskontrolle für Benutzer und Benutzergruppen. Externe Schemata können in einem anderen Datenmodell als dem kanonischen erstellt werden, wenn z.B. eine Anwendung das verlangt.

Wie leicht zu erkennen ist, wird bei dieser Architektur ein zusätzliches Schema im Komponentensystem eingeführt, was eine Verletzung der Entwurfsautonomie ist. Sollte das Komponentendatenbanksystem von Anfang an dasselbe Datenmodell benutzen wie die Föderation, so entfällt das Komponentenschema und die Autonomie ist bewahrt (das Exportschema ist dann das externe Schema). Ist das nicht der Fall, kann man das Komponentenschema als das externe Schema der Komponentendatenbank betrachten, wobei das Exportschema entweder entfällt und das gesamte lokale Schema der Föderation zur Verfügung gestellt wird, oder aber nach außerhalb des Komponentensystems verlegt wird, was für das Komponentensystem äquivalent (zur kompletten Freigabe des lokalen Schemas) ist. Diese Autonomieverletzung kann man vermeiden, wenn das Komponentenschema und das Exportschema zu einem Schema zusammengefügt werden. In diesem Fall würden sie einem externen Schema des Komponentendatenbanksystem entsprechen. Dies ist auch bei mehrfacher Föderation möglich (mit Hilfe mehrerer zusammengefügter Komponenten und Export Schemata).

Kapitel 3

Merkmale Mediator-basierter Informationssysteme

Neben den klassischen Kriterien zur Charakterisierung föderierter Informationsysteme, die wir in Kapitel 1 dargestellt haben, kann man noch weitere Unterscheidungsmerkmale finden. Wir stellen diese in einem eigenen Kapitel dar, da sie erst mit der Kenntnis der Grundarchitekturen von MBIS verständlich sind.

3.1 Enge versus lose Integration

Eine enge Integration unterscheidet sich von einer losen Integration im Vorhandensein eines integrierten (oder föderierten) Schemas ([41]). Enge Integration faßt alle Komponentenschemata bzw. die Komponenten-Exportschemata in einem homogenen Schema zusammen. Dies kann durch Schemaintegration oder auch ad-hoc erfolgen. Benutzer stellen Anfragen an dieses Schema; das System muß diese in Anfragen an die Komponenten zerlegen.

Lose Integration dagegen beinhaltet kein integriertes Schema. Hier müssen Benutzer stattdessen ihre "eigene Integration" durchführen, d.h. jede Benutzeranfrage adressiert direkt die Relationen oder Klassen der Komponenten.

Eine Grauzone wird betreten, wenn ein an-und-für-sich lose integriertes System die Definition von Views gestattet, die allen Benutzern zur Verfügung gestellt werden. Eine entsprechende Menge solcher Views bildet dann in gewissem Sinne ein integriertes Schema.

3.2 Lesen und/oder Schreiben

Viele Ansätze zur Informationsintegration untersuchen von vornherein nur den lesenden Zugriff auf die Komponenten. Dies liegt daran, daß schreibender Zugriff viele Schwierigkeiten mit sich bringt:

- Komponenten müssen in sehr starkem Maße ihre Autonomie abgeben.
- Verteilte Transaktionsverwaltung ist zum einen konzeptionell schwierig, und verlangt zum anderen entsprechende Fähigkeiten in den Komponenten.
- Eng integrierte System stehen oft vor dem klassischen "Update through views" Problem.

3.3 Datenmodell des Mediator

Jede eng-integrierende Komponente muß auf einem bestimmten Datenmodell aufbauen, in dem das integrierte Schema definiert ist (das "canonical data model" nach [41]).

Interessant ist die Frage, inwiefern die Integration von Komponenten unterstützt wird, die ein anderes Datenmodell verwenden. Dies betrifft die Frage, inwieweit Schemata in verschiedenen Modellen sich überhaupt verlustfrei und sinnvoll ineinander abbilden lassen.

3.4 Anfragesprachen in Wrappern

Obwohl viele Ansätze die Integration von sehr heterogenen Quelltypen explizit adressieren, z.B. WWW Quellen oder einfache, strukturierte Files, beachten viele die Probleme nicht, die dabei bzgl. der *Anfragebearbeitung* entstehen. Z.B. kann eine WWW Quelle, auch wenn sie klar formatierte HTML Tabellen erzeugt, noch lange keine Bedingungen auf den Tabellenelementen ausführen (im Sinne eines SQL SELECTs).

Insofern sind Systeme dahingehend zu unterscheiden, welche Anfragebeschränkungen sie tolerieren können. Wichtig ist auch die Frage, wie sie solche Einschränkungen kompensieren - die triviale Möglichkeit, Quellen immer zunächst komplett zu spiegeln und dann lokal, im Mediator, die Anfragen auszuführen, ist dabei im allgemeinen nicht befriedigend (siehe z.B. [39, 20, 47]). Außerdem kann man noch unterscheiden, in welcher Komponente eines MBIS die Kompensation vorgenommen wird - im Mediator, im Wrapper, in einer speziellen, anderen Komponente ?

3.5 Sammeln, Integrieren, Verweisen, Abstrahieren ...

Mediatoren lassen sich bzgl. ihrer Fähigkeiten im Umgang mit Daten unterscheiden:

- Einfachste Stufe ist das pure Sammeln von allen Daten so, wie sie von den Quellen bereitgestellt werden.
- Besser ist schon die Integration der Daten, z.B. in bezug auf:
  - Erkennen gleicher Objekte
  - Erkennen redundanter Daten
  - Unterstützung in der Auflösung von Datenkonflikten
- Nach [50] sollten Mediatoren außerdem in der Lage sein, Daten zu abstrahieren und zu aggregieren, d.h. nach semantischen oder strukturellen Gesichtspunkten zusammenzufassen.
- Eine weitere Möglichkeit ist es, die Daten überhaupt nicht anzufassen, sondern nur Verweise auf die Originaldaten zu liefern[1]

[1] So wie WWW Suchmaschinen i.A. auch nur Links, und nicht Pages zurückliefern.

3.6 Virtuelle Anfragebearbeitung versus Materialiserung

Alle Mediator-basierten Systeme werden im wesentlichen eine virtuelle Strategie verfolgen, d.h. Daten nicht in großem Maßstab replizieren [49]. Ausnahmen sind aber möglich: z.B. werden sehr oft Metadaten repliziert, wie z.B. Komponentenschemata, oder bestimmte, wichtige Daten wie Objektnamen. Hybride Architekturen sind auch denkbar.

Anfragebeantwortung auf materialisierten Daten geht wie folgt vor (siehe auch Abschnitt 6.1):

- Sämtliche Information wird zu bestimmten Zeitpunkten aus den Quellen in eine Data Warehouse Datenbank materialisiert (importiert).
- Anfragen an die Quellen werden nicht jedesmal an die Quellen durchgereicht, sondern direkt vom Data Warehouse beantwortet, welches ja alle nötigen Informationen aus den Quellen enthält.

Der wichtigste Vorteil der Verwendung eines Data Warehouses ist die geringere Komplexität der Anfragebearbeitung. Denn dadurch, daß alle Daten schon in einer vollwertigen Datenbank enthalten sind, können die Anfragen mit der Mächtigkeit moderner Datenbanken gestellt und beantwortet werden. Die benötigten Technologien müssen nicht neu entworfen und implementiert werden, sondern sind schon auf dem Markt verfügbar. Das Data Warehouses übernimmt in diesem Fall die Rolle des Mediators.

Virtuelle Anfragebeantwortung dagegen garantiert, daß die in der Antwort enthalten Daten aktuell sind. Der Nachteil ist die hohe Komplexität der Anfragebeantwortung, die teilweise neu zu entwickelnde Technologien benötigt.

Die virtuelle Anfragebearbeitung bietet außerdem weitere Vorteile:

- Der hohe Speicheraufwand und die hohe Performancebelastung beim Datenimport in das Data Warehouse finden nicht statt. Es werden nur die angefragten Daten extrahiert.
- Ein View ist zur Gewinnung von Daten aus einem Datenfluß besser geeignet. Die Verfügbarkeit rein "flussartiger" Quellen im Internet steigt immer weiter an.

3.7 Bottom-Up versus Top-Down

MBIS können unter zwei verschiedenen Gesichtspunkten entwickelt werden.

- **Top-Down** geht von den Anforderungen an das MBIS aus. Typischerweise beginnt der Prozeß mit der Definition eines globalen Schemas, das alle Konzepte beinhaltet, die auf der Föderationsschicht von Interesse sind. Ausgehend von diesem Schema werden dann relevante Komponenten bestimmt und logisch und semantisch mit dem globalen Schema in Beziehung gesetzt.
- **Bottom-Up** beginnt mit der Anforderung, eine bestimmte Menge von Komponenten in ein integriertes System zu verwandeln. Ausgangspunkt sind also die Komponentenschemata, von denen aus ggf. ein globales Schema abgeleitet wird.

Dieses Kriterium ist für die Evolutionsfähigkeit von Systemen von großer Bedeutung. Bottom-Up Ansätze laufen eher Gefahr, ein relativ abgeschlossenes System zu schaffen, das nur schwer auf neue Komponenten, Anforderungen oder andere Veränderungen reagieren kann. Bei Top-Down Systemen dagegen findet man oft relativ schwache Bindungen von Komponenten an das System (bestes Beispiel: Integration von WWW Quellen). Auch ist die "semantische Trennschärfe" oft geringer, da die Komponenten nicht von vorneherein im Entwicklungsprozeß beteiligt sind.

Kapitel 4

Hilfsmittel für Mediatoren

In den vorangegangenen Kapiteln wurden eine Menge Anforderungen und Klassifikationskriterien für Mediator-basierte Informationssysteme formuliert. Es sollen jetzt einige Konzepte, Techniken und Methoden vorgestellt werden, die geeignet sind, bestimmte dieser Anforderungen zu erfüllen oder zu unterstützen.

Begonnen wird mit der Diskussion der Schemaintegration - einer Technik zur Erstellung eines globalen Datenbankschemas auf der Grundlage der Komponentenschemata. Danach werden verschiedene Ansätze zur Mediation und deren Repräsentationstechniken diskutiert. Dazu gehören Ontologien und Wissensrepräsentationssprachen, Metainformationskonzepte sowie die Behandlung semistrukturierter Daten. Abschließend gehen wir gesondert auf die Anfragebearbeitung und -optimierung ein, die bei jedem virtuellen Mediationsansatz von Bedeutung ist (und meist wird ein virtueller Ansatz gewählt).

Neben der Diskussion der Nützlichkeit soll insbesondere auch deutlich gemacht werden, wie die Konzepte und Techniken in Beziehung stehen bzw. kombiniert für einen flexiblen Mediationsentwurf eingesetzt werden können.

4.1 Schemaintegration

Vorbemerkung

In der Literatur wird unter dem Begriff Schemaintegration sowohl Viewintegration als auch Datenbankintegration verstanden. Die Datenbankintegration befaßt sich mit der Erzeugung eines globalen, integrierten Schemas aus vorgegebenen lokalen Schemata. Viewintegration bedeutet im Gegensatz dazu das Ableiten eines Datenbankschemas aus der Summe der Nutzersichten bzw. Nutzeranforderungen.

Hier wird der Begriff Schemaintegration ausschließlich im Sinne der Datenbankintegration verwendet, d.h. wir betrachten hier die Integration von strukturierten Daten, wie sie durch Datenbankschemata beschrieben werden. Die Anwendung der weiter unten vorgestellten Integrationsmethoden auf semistrukturierte Daten ist nicht vorgesehen. Sie würde zumindest eine zusätzliche "Umsetzungsschicht" erfordern, die semistrukturierte Datenquellen auf wohldefinierte Schemata abbildet. Inwieweit die Schemaintegration im Kontext semistrukturierter Daten sinnvoll ist, bleibt fraglich. Die im folgenden betrachteten Anforderungen an ein globales, integriertes Schema liegen im Konflikt mit der den semistrukturierten Daten innewohnenden Vagheit.

Ein wesentlicher Aspekt der Datenbankintegration ist die Autonomie der lokalen Schemata, das heißt, das integrierte Schema muß erzeugt werden können, ohne Anpassungen an den lokalen Schemata vorzunehmen.

Bei einer Schemaintegration müssen in der Regel eine Reihe von Integrationskonflikten aufgelöst werden, die sich unter anderem aus der unterschiedlichen Modellierung der entsprechenden Diskursbereiche ergeben. Da diese Konflikte nicht maschinell erkannt und gelöst werden können, kann der Gesamtprozeß der Schemaintegration nicht automatisiert werden.

Gesucht werden also unterstützende Formalismen, die dem jeweiligen Datenbankentwerfer in Form von geeigneten Werkzeugen bei seiner Aufgabe helfen.

4.1.1 Anforderungen

Selbstverständlich muß das integrierte Schema bestimmten Anforderungen genügen. In [5] werden folgende Kriterien vorgeschlagen, an deren Erfüllung sich eine Schemaintegrationsmethode messen lassen muß:

- completeness

  Das integrierte Schema muß alle Konzepte der lokalen Schemata enthalten.

- minimality

  Äquivalente Konzepte der lokalen Schemata sind im integrierten Schema nur einmal repräsentiert.

- correctness

  Alle im integrierten Schema enthaltenen Informationen müssen in mindestens einem der lokalen Schemata bedeutungsgleich vorhanden sein.

- understandability

  Das integrierte Schema soll leicht verständlich sein.

4.1.2 Vorgehensweise

Ebenfalls in [5] wird folgendes 3-Phasenmodell als Vorgehensweise zur Schemaintegration vorgeschlagen:

Bei der sogenannten Vorintegration wird die Integrationsstrategie und das Integrationsverfahren bestimmt. Auf mögliche Integrationsstrategien werden wir noch näher eingehen. Um Konflikte zwischen den lokalen Schemata finden zu können, müssen sie vergleichbar gemacht werden. Dazu wird jedes der zu integrierenden lokalen Schemata in einem einheitlichen Datenmodell ausgedrückt. Dieses auch 'CDM' (für "Common Data Model") genannte Datenmodell wird von der jeweiligen Integrationsmethode zur Verfügung gestellt.

Im zweiten Schritt, der Vergleichsphase, werden Übereinstimmungen zwischen den lokalen Schemata, sogenannte 'Inter-Schema-Korrespondenzen' festgehalten. Auch das Identifizieren von Konflikten durch den Datenbankentwerfer muß in dieser Phase durchgeführt werden.

Im dritten Schritt, der Zusammenführung und Restrukturierung, werden die erkannten Konflikte aufgelöst und die einzelnen Komponenten zum integrierten Schema zusammengeführt.

4.1.3 Integrationsstrategien

Unter dem Begriff Integrationsstrategie versteht man die logische Abfolge bei der Integration der einzelnen lokalen Schemata. Es wird zwischen binären und nicht binären Strategien unterschieden. Bei einer binären Integrationsstrategie werden solange immer zwei lokale Schemata zusammengefaßt, bist das globale Schema erreicht ist. Die nicht binären Strategien sind flexibler: Hier können auch mehr als zwei Schemata in einem Schritt bearbeitet werden. Unter Gewichtung versteht in diesem Zusammenhang die Bevorzugung eines lokalen Schemas bei der Integration. Das resultierende globale Schema ist dadurch stärker von diesem Schema geprägt, als es bei einer binären Strategie der Fall gewesen wäre. Mit der 'one shot' Strategie ist die Integration aller lokalen Schemata in einem Schritt gemeint, allerdings wird in der Literatur nicht näher darauf eingegangen.

4.1.4 Klassifikation von Integrationsmethoden

Der Themenkomplex der Schemaintegration ist durchaus kein neues Forschungsgebiet, schon Anfang der 80er Jahre wurden erste Vorschläge für Integrationsmethoden veröffentlicht. Um die verschiedenen Ansätze vergleichen zu können, werden im folgenden einige Klassifikationskriterien vorgestellt:

- **Modellierungskonzepte:** Die Integrationsmethode wird danach beurteilt, welche der gängigen Modellierungskonzepte sie unterstützt, beispielsweise Attribute, Entity- und Beziehungstypen oder Spezialisierung und Objektaggregation.

- **Konfliktklassen:** Die Integrationsmethode wird danach beurteilt, welche der verschiedenen Heterogenitäts-Konfliktklassen sie behandeln kann. Dazu zählen z.B. extensionale Konflikte, Beschreibungskonflikte und strukturelle Konflikte. Auf die einzelnen Konfliktklassen soll hier nicht mehr näher eingegangen werden (siehe Abschnitt 1.3).

- **Mögliche Integrationsstrategien:** Auch die Unterstützung der oben vorgeschlagenen Integrationsstrategien kann als Klassifikationskriterium dienen. Bei einer Methode, die beispielsweise nur eine binäre Strategie zuläßt, ist eine Gewichtung einzelner Schemata nicht möglich.

4.1.5 Integrationsverfahren im Überblick

Nach der Diskussion einiger allgemeiner Klassifikationskriterien folgt in diesem Abschnitt ein Überblick über einige Schemaintegrationsmethoden:

Zusicherungsbasierte Integration: GDM

Bei zusicherungsbasierten Integrationsmethoden werden Aussagen über Gemeinsamkeiten und Abweichungen zwischen den zu integrierenden Schemata getroffen, die bei der eigentlichen Integration anhand festgelegter Regeln berücksichtigt werden. Als Beispiel für diesen Methodentyp sei die GDM - Methode nach [42] genannt. GDM steht für 'Generic Data Model'. Wie der Name schon andeutet, stellt diese Methode als Grundelement einen Objekttyp zur Verfügung, der einem Entity- oder Relationshiptyp eines klassischen ER-Diagramms entspricht.

Der oben genannte GDM-Objekttyp enthält Wertattribute und eventuell sogenannte Links. Ein Link entspricht einer Referenz auf einen anderen GDM-Objekttyp und drückt die in einem ER - Diagramm modellierten Beziehungen aus. Da die Methode ihren Ursprung im relationalen Umfeld hat, ist die Integration von Klassenhierarchien aus objektorientierten Datenmodellen nicht vorgesehen.

Nachdem die Entity- und Relationshiptypen der Ausgangsschemata als GDM-Objekttypen dargestellt sind, werden Zusicherungen sowohl für die Objekttypen als auch für sogenannte Pfade aufgestellt. Ein Pfad beschreibt die Beziehungen zwischen den Objekttypen eines Schemas, beispielsweise 'Kunde-kauft-Ware'.

Da die Zusicherungen immer Aussagen über jeweils genau zwei Objektypen, Links oder Pfade darstellen, impliziert GDM immer eine binäre Integrationsstrategie.

Upward Inheritance

Methoden zur Integration verschiedener Klassenhierarchien werden oft unter dem Begriff 'upward inheritance' zusammengefaßt. Bei dieser 'Vererbung nach oben' werden bei der Integration einzelner Objektklassen gemeinsame Oberklassen erzeugt, die diejenigen Eigenschaften ausdrücken, die in allen Ausgangsklassen äquivalent vorhanden sind. Die jeweilige Vererbungshierarchie der zu integrierenden Schemata bleibt im globalen Schema unverändert erhalten. Daher ist das Ergebnis der upward Inheritance - Methoden meist eine recht umfangreiche und dementsprechend unübersichtliche Klassenhierarchie.

Flexible Integration: GIM

Diesen Nachteil versucht das 'Generic Integration Model' nach [40] zu vermeiden. Auch hier handelt es sich um eine Methode zur Integration von Klassenhierarchien, allerdings werden die ursprünglichen Vererbungshierarchien nicht direkt im globalen Schema abgebildet. Um dieses Ziel zu erreichen, wird zuerst eine sogenannte extensionale Analyse und Zerlegung durchgeführt. Die Extension einer Klasse entspricht der Menge aller Objekte dieser Klasse. Bei der extensionalen Analyse werden Überlappungen zwischen den Objektmengen der zu integrierenden Klassen ermittelt. Danach wird eine Zerlegung in disjunkte Klassen vorgenommen. Eine disjunkte Klasse repräsentiert also eine überlappungsfreie Objektmenge.

Diese Analyse setzt allerdings, wie auch die Suche nach Inter-Schema Korrespondenzen bei der GDM-Methode, eine genaue Kenntnis der Beziehungen zwischen den Realwelt - Objekten voraus, die in den Ursprungsschemata enthalten sind. Die gegebene Klassenstruktur einer Datenquelle reicht alleine nicht aus, um eine korrekte extensionale Analyse vornehmen zu können.

Nach diesem ersten Schritt folgt die intensionale Analyse und Zerlegung. Dabei werden die Attribute der zu integrierenden Klassen zu disjunkten Attributmengen zusammengefaßt. Das Ergebnis dieses Vorgangs ist eine GIM-Matrix in folgender Notation, die auch normalisiertes Schema genannt wird:

Jede Spalte einer GIM-Matrix entspricht einer disjunkten Extension. Die zu einer Extension gehörigen Attributmengen finden sich in den Zeilen der Matrix. Ein Kreuz in einem Feld gibt an, daß Objekte der entsprechenden Extension für das entsprechende Attribut Werte besitzen. Jede

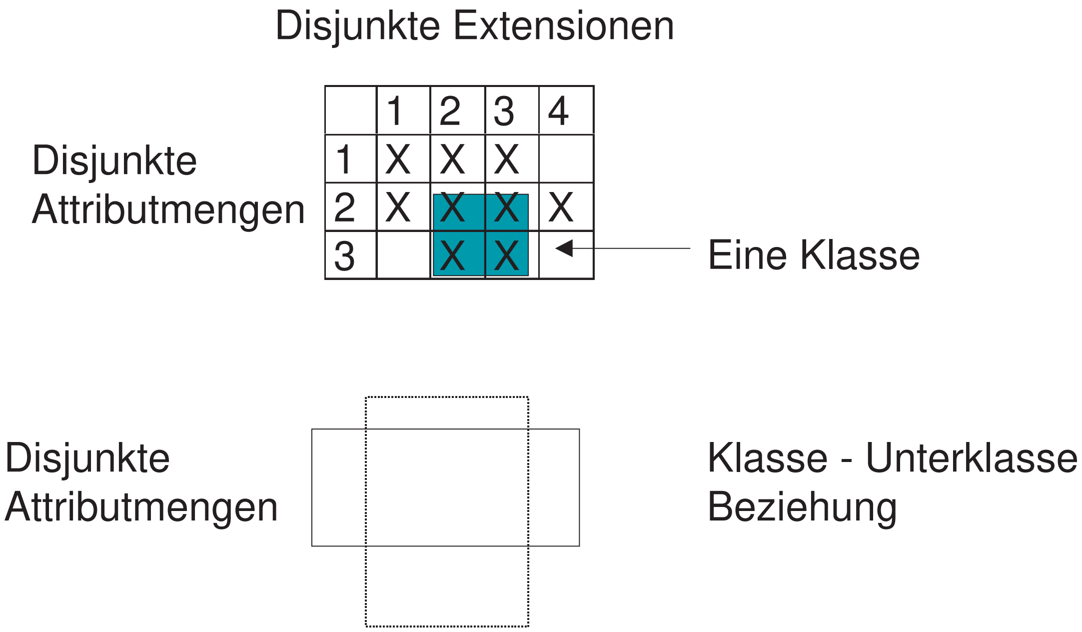

Abbildung 4.1. Gim Notation im Überblick

Untermatrix, ein mit Kreuzen markierter rechteckiger Bereich also, kann als eine Klasse betrachtet werden. Klasse - Unterklasse Beziehungen ergeben sich aus Überlappungen der Rechtecke.

Aus dieser Matrix können verschiedene Varianten des integrierten Schemas abgeleitet werden. Dazu werden Zeilen und Spalten der Matrix in Dreiecksform sortiert, um dann Klassen und Unterklassen ablesen zu können. Als letzter Schritt muß sichergestellt werden, daß alle Spalten der Ergebnismatrix, die Extensionen also, in unterscheidbare eigene Unterklassen abgebildet worden sind. Ist das nicht der Fall, wird ein zusätzliches Attribut eingeführt, das diese Unterscheidung ermöglicht.

Bei einer anderen Sortierung der Zeilen und Spalten bzw. einer alternativen Aufteilung in Klassen und Unterklassen ergibt sich jeweils eine andere Klassenhierarchie.

4.1.6 Fazit

Obwohl sich die Forschung schon über einen längeren Zeitraum mit dem Themenkomplex der Schemaintegration befaßt, kann die Entwicklung bei weitem nicht als abgeschlossen betrachtet werden. Die einzelnen Integrationsmethoden sind teilweise in einem unvollendeten Stadium geblieben oder auf einige, wenige Anwendungsgebiete beschränkt. Allerdings wäre die Forderung nach einer Methode für alle denkbaren Anwendungsfälle auch überzogen, zumal der entscheidende Teil der Schemaintegration, das Erkennen, Beurteilen und Auflösen der Integrationskonflikte immer dem Datenbankentwickler überlassen bleiben wird.

Ein weiteres zentrales Problem ist die mangelnde Evolutionsfähigkeit des integrierten Schemas: Bei der Änderung oder Erweiterung eines der zugrundeliegenden lokalen Schemata muß der gesamte Integrationsprozeß erneut erfolgen, was bei größeren Systemen einen nicht unerheblichen Aufwand bedeutet.

Nicht zuletzt deswegen sind föderierte Datenbanksysteme in der Praxis eher unüblich geblieben. Inwieweit sie sich noch durchsetzen werden, bleibt also abzuwarten.

4.2 Ontologien und Wissensrepräsentationssprachen

4.2.1 Grundidee: Festlegung des Diskursbereiches

Um logische Heterogenität zwischen zu integrierenden Informationsquellen zu überbrücken, ist es sinnvoll, zur Beschreibung eines globalen Schemas ein präzises Vokabular zu benutzen, um den gemeinsamen Diskursbereich zu beschreiben. Diese Beschreibung aller Elemente, Relationen oder Funktionen durch eine formale oder natürliche Sprache wird Ontologie genannt. Uns interessiert hier der formale Ansatz durch Sprachkonstrukte der Description Logic, die speziell zur Wissensrepräsentation entwickelt wurden. Mit Hilfe einer solchen Sprache wird ein globaler Diskursbereich für die Anforderungen der Föderationsschicht modelliert, der die relevanten Teile der Diskursbereiche der Informationsquellen enthält. Die Beschreibung des Diskursbereiches teilt sich in Konzepte (vergleichbar mit Klassen, Objekten, Entity-Typen), Rollen (vergleichbar mit Assoziationen, Relationship-Typen, Aggregationen), und Attribute (atomare Werte eines Konzeptes), die in einen Zusammenhang gestellt werden. Zusätzlich können Generalisierungen/Spezialisierungen, virtuelle Konzepte und Rollen, Schlüsselattribute, extensionale Eigenschaften und globale Axiome (Wissen über die Daten, z.B. in der Form "Häfen mit mehr als 10 Docks haben einen Tiefgang von über 5m") spezifiziert werden. Diese *globale Beschreibung* des Diskursbereiches wird *Domain Model* genannt, und unabhängig von den Quellen modelliert.

4.2.2 Integration mit Hilfe von Ontologien

Die Quellen werden anschließend mit Hilfe der Ontologie integriert. Dies geschieht durch eine Übersetzung der lokalen Schemata in die Sprache der Ontologie. Diese Übersetzung ist zunächst nur syntaktisch, da das Ergebnis noch unabhängig von Domain Model Konzepten ist. Stattdessen wird jeweils ein neues Konzept für ein Objekt des lokalen Schemas erstellt. Die Art des lokalen Schemas ist hierbei irrelevant. Diese Konzepte einer Informationsquelle ergeben zusammengenommen ein *Source Model*. Dessen Konzepte und Rollen werden dann mit den Konzepten und Rollen des Domain Models durch "IS-A"-Beziehungen assoziiert (siehe Fig. 4.2. Es handelt sich bei Ontologien, wie man leicht erkennen kann, um eine enge Integration und um einen Top-Down Ansatz.

Der Ontologie-basierte Ansatz zur Integration erhält die Autonomie der zugrundeliegenden Informationsquellen, da sie in erster Linie nur zur Informationsdarstellung verwendet wird, obwohl auch eine Datenmanipulation denkbar wäre.

Die Hauptproblematik, die eine Ontologie zu lösen versucht, ist die *logische Heterogenität*. Syntaktische Heterogenität wird im allgemeinen durch Wrapper gelöst, das heißt die Informationsquellen werden gekapselt, globale Anfragen in die lokale Anfragesprache übersetzt. Hier

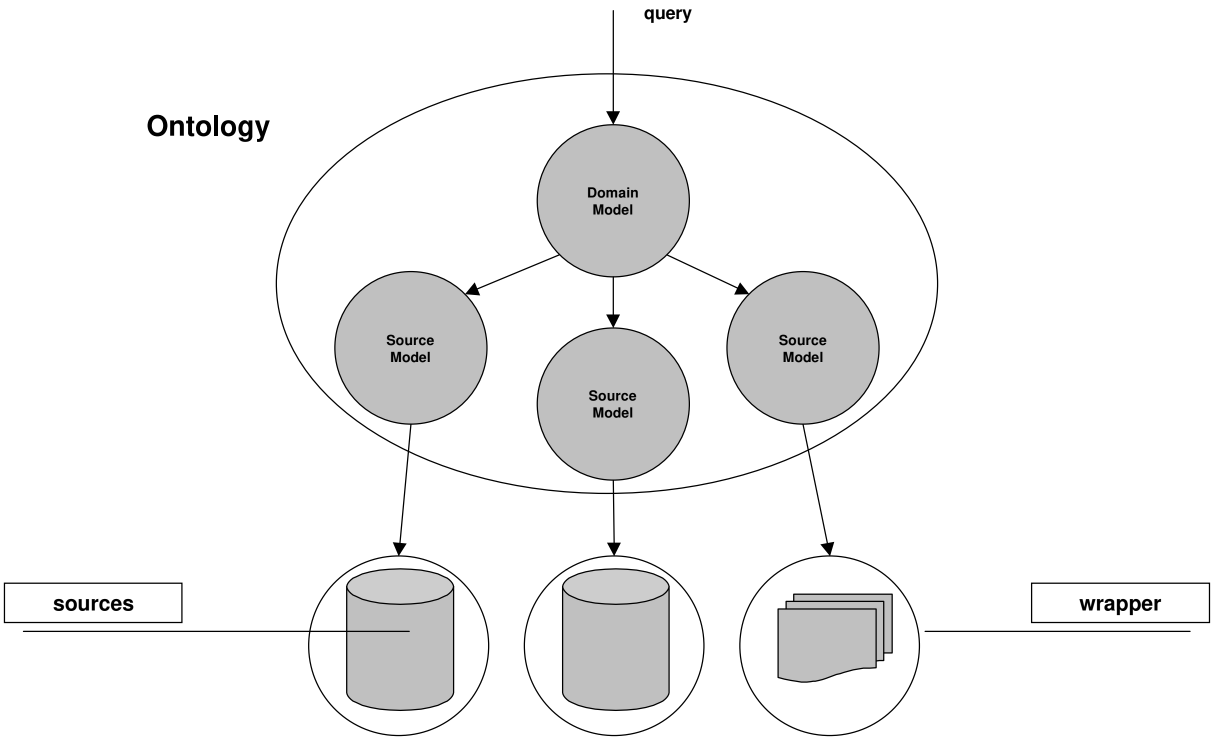

Abbildung 4.2. Semantisch Integration mit Hilfe einer Ontologie.

tritt allerdings das Problem der Ausdruckskraft der lokalen Anfragesprachen gegenüber der meist mächtigeren globalen auf, was unter Umständen zu einer unzureichenden Übersetzung führen kann, die nicht die erwarteten Daten liefern kann.

Die *Verteilung* der Quellen wird nicht direkt durch eine Ontologie behandelt. Die Schnittstellen müssen entsprechend gekapselt werden.

Bzgl. der *Evolutionsfähigkeit* eines solchen Systems läßt sich feststellen, daß es gegenüber kleinen Änderungen oder Erweiterungen eines Komponentenschemas stabil ist, solange sie sich durch Veränderungen der Model-Beziehungen abhandeln lassen. Problematisch sind größere Veränderungen, die eine Umstrukturierung des Domain Models erfordern. Erweiterungen des Domain Models dagegen haben keine Auswirkungen auf vorhandene Source Models, Änderungen können aber zu einer Umstrukturierung der Assoziationen zu den Source Models führen, was jedoch kein großer Arbeitsaufwand wäre. Konfigurationsänderungen der Komponententen führen ebenfalls zu keinem weiteren Aufwand innerhalb der Ontologie, allerdings müssten die Wrapper entsprechend verändert werden, zum Beispiel wegen einer Ortsänderung einer Komponente.

Die Integration von *semi-strukturierten Daten* ist theoretisch auch möglich, wenn sie mit Hilfe der Description Logic abgebildet werden können. Das Wrappen solcher Quellen ist unabhängig von der Ontologie und beinhaltet daher die normalen Einschränkungen bezüglich einer Anfrage. Die Art und Weise, wie Daten ausgewertet und kompensiert werden, ist ebenfalls unabhängig von der Ontologie, jedoch wird dieser Punkt im Beispiel SIMS näher beleuchtet (siehe Abschnitt 5.4).

In Ontologie-basierten Systeme wird als *Mediator* der Mechanismus bezeichnet, der Anfragen an die Ontologie an die Informationsquellen weiterleitet und die Ergebnisse auswertet, bzw. die Komponente die diese Aufgaben erfüllt. Er arbeitet virtuell, sozusagen transparent, auf den gekapselten Quellen. Metadaten über die Komponentenschemata sind im Source Model integriert, sie werden nicht erst zur Anfragebearbeitung von den Wrappern abgefragt. Die Fähigkeiten des Mediators in bezug auf das Erkennen gleicher oder redundanter Daten ist durch den Ansatz mit Description Logic sicherlich einfacher als in vergleichbaren Systemen, inwieweit solche aber umgesetzt werden ist eine andere Sache. SIMS kann dies beispielsweise nicht, es begnügt sich mit dem einfachen Sammeln der Daten. Die Wissensgenerierung aus Daten wird durch eine Ontologie ebenfalls vereinfacht, da hier eine logische Beschreibung der Daten zur Verfügung steht, aus der sich automatisiert eine Konklusion bilden läßt.

4.2.3 Ontologie-Integration

Eine Ontologie kann leider nicht allumfassend sein. Wie immer findet sich in ihr die Sichtweise ihrer Entwickler wieder, die in einem anderen Kontext nicht optimal sein muß. Existieren verschiedene Ontologie-basierte Systeme zu einem Diskursbereich, kann es notwendig werden, diese ebenfalls zu integrieren, da ein einfacher Zusammenschluß nicht immer möglich sein wird, vor allem nicht, wenn diese Systeme unterschiedliche Wissensrepräsentationssprachen nutzen. Zu diesem Zweck wurde Ontolingua entwickelt ([16]). Ontolingua ist ein System mit einer sehr ausdrucksstarken, KIF-ähnlichen Sprache, die in andere Sprachen, z.B. Description Logic, übersetzt werden kann. Ontologien, die damit modelliert wurden, sind somit portabel, können dabei jedoch an Semantik verlieren, wenn die Zielsprachen nicht dieselbe Ausdrucksstärke wie Ontolingua besitzen.

Ontolingua hilft natürlich nicht bei der Integration von bereits existierenden Ontologien. Höchstwahrscheinlic ist der Prozeß der Ontologie-Integration mindestens so aufwendig wie eine Schemaintegration für föderierte Datenbanken (siehe Abschnitt 4.1).

4.2.4 Description Logics

Description Logics (DLs) stammen von der KL-ONE Sprache ab, die innerhalb der künstlichen Intelligenz Forschung entstanden ist. Der grundlegende Vorteil von DLs ist, daß sie nicht nur semantische Sachverhalte beschreiben können, sondern es auch möglich ist, automatisiert Schlußfolgerungen aus diesen Beschreibungen zu ziehen (reasoning), was durch Subsumption geschieht ([6]). Zum Beispiel wird die Beschreibung eines Kurses mit höchstens 10 Teilnehmern von einem Kurs mit höchstens 15 Teilnehmern subsumiert, da alle Kurse der ersten Art automatisch die Bedingung des Kurses der zweiten Art erfüllen. Subsumption entspricht in gewissem Sinne dem Generalisierungs / Spezialisierungs-Mechanismus aus der EER- und OO-Welt.

Alle DLs seit dem KRYPTON System nutzen eine formale lineare Syntax zur Beschreibung von Konzepten. Diese Beschreibungen entstehen durch Termkomposition ([6]). DLs haben mindestens zwei Termarten, Konzepte und Rollen. Die Syntax einer DL besteht aus Regeln zur Erzeugung zusammengesetzter Terme aus atomaren Symbolen und Termkonstruktoren. Solche Termkompositionen sind in der Regel wesentlich kürzer und menschenverständlicher als die semantisch äquivalente Notation in der Prädikatenlogik erster Stufe. Zusätzlich ist das automatisierte Schlußfolgern in dieser Notation performanter, da hier zum Beispiel zur Überprüfung einer Subsumption ein einziger Vergleichsoperator ausreicht. Dieser Unterschied entsteht hauptsächlich dadurch, das DLs auf Scoping und Substitutionsmechanismen verzichten. Ein weiterer Vorteil ist das Definieren von virtuellen Konzepten (vergleichbar mit Views), die existierende erweitern beziehungsweise verbinden. Anfragen an ein DL-basiertes Informationssystem werden durch eine Definition eines solchen virtuellen Konzeptes beschrieben.

Die Fülle unterschiedlicher DLs zeigt eine grundlegende Korrelation auf. Je ausdrucksstärker eine DL ist, desto unentscheidbarer und komplexer wird sie. Die Ausdrucksstärke wird durch die Anzahl und Art der zur Verfügung stehenden Termkonstruktoren beeinflußt. Je mehr davon zur Verfügung stehen, desto mehr wichtige semantische Aspekte können repräsentiert werden. Beispiele für DLs sind Loom und Classic.

4.3 Metainformation

4.3.1 Begriffsbestimmung

Der Begriff 'Metainformation' taucht in vielen Kontexten auf und wird teilweise sehr kontrovers diskutiert, so daß eine präzise Begriffsbestimmung und Abgrenzung zu verwandten Begriffen, wie 'Daten' oder 'Wissen' kaum möglich erscheint. Aufbauend auf den Definitionen in [31] und [7] wollen wir für unsere Diskussion folgende Terminologie festlegen:

Daten bezeichnen Werte bestimmter Merkmale von Objekten. In der Begriffswelt der Datenverarbeitung lassen sich Daten in einer maschinenlesbaren Form darstellen. Werden Daten vom Menschen bezüglich eines bestimmten Kontextes interpretiert, so entsteht *Information*. Der Informationsbegriff schließt also eine Bedeutung der Daten mit ein, die einen unmittelbaren Nutzen für den Menschen innerhalb des jeweiligen Kontextes bringt. Mit *Wissen* wird schließlich die Fähigkeit für eine Dateninterpretation bezeichnet. Wissen bedeutet also, Daten und die Fähigkeit zu besitzen, aus diesen Daten bei Bedarf Information abzuleiten.

Metainformation wird häufig als 'Information über Information' definiert. Es handelt sich dabei um Information, die andere Information beschreibt, wobei die Art der Beschreibung wiederum stark kontextabhängig ist. *Metadaten* stellen wiederum die maschinenlesbare Repräsentation von Metainformation dar.

Wir werden im folgenden die Begriffe Daten und Information im wesentlichen synonym verwenden. Doch auch die Grenze, wann wir etwas als Daten bzw. Information und wann als Metadaten bzw. Metainformation bezeichnen, ist äußert schwierig zu ziehen. Deshalb noch einmal die zwei wesentlichen Unterschiede:

- Da Metadaten Daten und Metainformation Information ist, finden sich auf der Metaebene also die gleichen Begriffe, Eigenschaften und Beziehungen wieder wie auf der Datenebene. Dies impliziert, daß sich im Kontext von Systementwicklungen für die Metaebene die gleichen Probleme bzgl. der Modellierung, des Entwurfs und der Realisierung stellen wie auf der Datenebene.

- Metainformation hat immer einen beschreibenden Charakter - sie beschreibt andere Information. Das heißt, dieser Begriff wird immer nur dann verwendet werden, wenn auch die Datenebene und die Beziehung zwischen den Ebenen betrachtet werden.

4.3.2 Allgemeine Zielsetzung von Metainformationskonzepten

Im Kontext mediator-basierter Informationssysteme taucht Metainformation verschiedener Charakteristik für ganz unterschiedliche Zielsetzungen auf. So können z.B. Metamodelle (Metadaten über Datenmodelle) verwendet werden, um Datenmodellkonflikte zu lösen, oder Metainformation über die Komponentensysteme gespeichert werden, um heterogene Schnittstellen etc. zu überbrücken (syntaktische Heterogenität). Anhand dieser zwei Beispiele läßt sich bereits die wesentliche Zielsetzung von Metainformationskonzepten erkennen: die Integration heterogener Komponenten, wobei die Art der Heterogenität auch die Charakteristik der Metainformation bestimmt (Kontextabhängigkeit).

Darüber hinaus wird Metainformation eingesetzt, um die Evolution von Softwaresystemen zu unterstützen. Indem Metadaten explizit erfaßt, im System gespeichert und als Daten genutzt werden, reduzieren sich Änderungsaufgaben häufig auf einfache Datenänderungen, wo früher Programme oder Konfigurationen geändert werden mußten.

Wir wollen uns nun aber nicht mit allen Arten von Metainformation beschäftigen, sondern nur mit denen, die die semantische Integration autonomer Informationsquellen betreffen.

4.3.3 Metainformationsbasierte Integration heterogener Datenquellen

In komplexen Anwendungsgebieten, wie etwa dem Umweltbereich, ist eine semantische Integration verschiedener Datenquellen über eine Schemaintegration nicht möglich [1]. Stattdessen werden standardisierte Metadatensätze definiert, mit denen zu vermittelnde Daten und Datenbestände beschrieben werden. Die Vermittlung erfolgt ausschließlich über diese Metadaten. Die Metadaten werden konkret dazu verwendet

- die Suche nach relevanten Daten zu unterstützen,
- Entscheidungen über die Verwendbarkeit der Daten zu ermöglichen und
- die Beschaffung der Daten zu erleichtern.

Da die inhaltliche Beschreibung Kern der Metadaten ist, sind die Metadatenmodelle von der Anwendungsdomäne geprägt. In einer Vielzahl von Anwendungsbereichen existieren bereits allgemein anerkannte Metadatenstandards, etwa die Dublin Core - Initiative [10] für die Annotation von WWW-Dokumenten, den Umweltdatenkatalog UDK oder FGDC im Umwelt- und Geobereich (vgl. [15]) oder HL7 [17] in der Medizin.

Wichtig zu bemerken ist, daß die Metadaten zum großen Teil *zusätzlich* erfaßte Daten sind, die genau wie die originären Daten erfaßt, verwaltet und gepflegt werden müssen. Nur ein Teil des Metadatenbestands kann aus dem Datenbestand extrahiert werden, wie z.B. der Titel eines Dokuments. Für die Bereitstellung zusätzlicher Daten, etwa Stichworte, werden häufig Ontologien oder genauer Thesauri als Referenz verwendet.

Das Szenario der Integration unter der Verwendung inhaltlicher Metainformation folgt der Abbildung Fig. 4.3: Zu jedem Datenbestand wird ein beschreibender Metadatenbestand angelegt. Nur dieser wird für die Integration verwendet. Verwendet jedes Komponentensystem ein eigenes, lokales Metadatenmodell, so erfordert der Ansatz die Integration dieser Metadatenschemata – wobei dieselben Techniken verwendet werden können wie wir sie bereits kennengelernt haben (z.B. eine Schemaintegration). Die Metadaten werden in der Regel auf der Föderationsschicht gehalten

[1] Im Bereich landesweiter Umweltinformationssysteme (UIS) wurde der Versuch unternommen, ein umfassendes integriertes Schema zu entwerfen, nach einigen Jahren jedoch abgebrochen.

(Materialisierung). Eine Anfrage wird zunächst mit Hilfe dieses Metadatenbestands bearbeitet. Im letzten Schritt erfolgt dann der Zugriff auf alle Datenobjekte, die durch die ausgewählten Metadatenobjekten beschrieben werden.

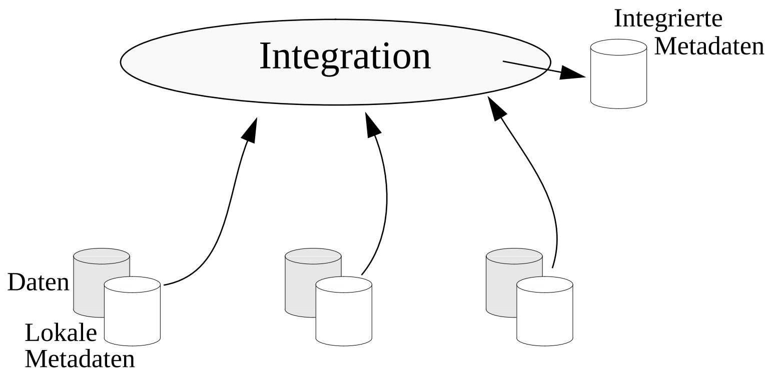

Abbildung 4.3. Metadatenbasierter Integrationsansatz

Vergleichen wir eine metainformationsbasierte Integration mit einer Integration auf der Basis der Komponentenschemata, so können folgende Unterschiede festgehalten werden:

- Bei einem metainformationsbasierten Integrationsansatz müssen die eigentlichen Daten nicht unbedingt modelliert worden sein - ein Vorteil, der sich bei der Integration unstrukturierter Daten zeigt. Die Daten müssen nicht einmal digital gespeichert sein; dann ist allerdings nur ihr Nachweis, nicht jedoch ein dynamischer Zugriff auf sie möglich.

- Eine metainformationsbasierte Integration vermittelt meist grobgranularere Datenobjekte, da meist mehrere Datensätze zusammengefaßt und mit einem Metadatensatz beschrieben und integriert werden. Sogar ganze Datenbestände können hier als 'Paket' in das System eingebracht werden.

- Metadaten sind eher klein und statisch, d.h. stabiler als Daten. Daraus folgt, daß Änderungen in einem Metadatenbestand weitaus seltener notwendig sind als in einem Datenbestand (allerdings nur, wenn der Metadatensatz entsprechend generisch entworfen wurde). Dies wirkt sich natürlich auf die Evolutionsfähigkeit aus.

- Als zentraler Nachteil eines metainformationsbasierten Ansatzes ist zu bemerken, daß die eigentliche Integration ausschließlich auf der Metadatenebene stattfindet, d.h. nur die Metadaten werden 'semantisch' integriert und nicht die eigentlichen Datensätze.

- Die Abstraktion von der Daten- zur Metadatenebene impliziert natürlich, daß wir keine direkte Manipulation der Daten vornehmen können. D.h. dieser Integrationseinsatz ist ausschließlich in Szenarien verwendbar, in denen lesend auf Daten zugegriffen werden soll.

4.3.4 Diskussion

Bezogen auf die in Kapitel 1 und 3 eingeführten Kriterien kann ein metainformationsbasierter Integrationsansatz folgendermaßen charakterisiert werden:

- Die Fragen der Autonomie, Heterogenität und Verteilung stellen sich bei der Verwendung inhaltlicher Metadaten auf der Metadatenebene genauso wie zuvor auf der Datenebene. Zur Überbrückung gerade von Heterogenität können verschiedene, von uns nicht näher betrachtete Arten von Metainformation verwendet werden.
- Die Evolutionsfähigkeit wird dadurch unterstützt, daß Metadaten in der Regel stabiler sind, so daß Änderungen seltener notwendig sind.
- Der Integrationsansatz verfolgt eine enge Kopplung – es wird ein Metadatenschema auf der Mediationsebene zur Verfügung gestellt.
- Es ist ausschließlich ein lesender Zugriff möglich.
- Das Datenmodell des Mediators kann beliebig gewählt werden; in der Regel werden strukturierte Datenmodelle (relational oder objektorientiert) verwendet, da die Metadaten strukturiert sind. Die Daten können dagegen auch semi- oder unstrukturiert sein, da diese Schemata nicht in den eigentlichen Integrationsprozeß eingehen.
- Die Frage der Anfragesprachen stellt sich bei diesem Ansatz weniger. Der Rückgriff auf die Komponenten beschränkt sich auf die Ermittlung der ausgewählten, mit den Metadatensätzen beschriebenen 'Datenpakete' (was meist das einfache Verfolgen einer Referenz, z.B. einer URL, ist).
- Bei einer metainformationsbasierten Integration treten ganz unterschiedliche Arten auf, wie mit Daten umgegangen wird:
  - Daten werden zunächste zu Paketen zusammengefaßt und mit Verweisen von der Metadatenebene versehen.
  - Durch die losgelöste Bildung von Metainformationsschemata können beliebige Abstraktionen vorgenommen werden.
  - Die Metadaten verschiedener Komponentensysteme werden von dem Mediator gesammelt ohne weitere Integrationsschritte durchzuführen.
- In der Regel werden die Metadaten auf der Mediationsebene materialisiert.
- Die Entwicklung des Mediatorschemas (des Metadatenschemas) kann sowohl bottom-up durch Integration verschiedener Metadatenschemata der Komponentensysteme erfolgen als auch top-down ausgehend von globalen Informationswünschen. In der Regel erfolgt eine kombinierte Vorgehensweise, bei der existierende Daten und Metadaten als auch existierende Referenzmodelle für Metadaten der jeweiligen Domäne Berücksichtigung finden.

Insgesamt läßt sich sagen: Ein Mediator auf der Basis inhaltlicher Metainformation ermöglicht die enge Kopplung großer heterogener Datenbestände beliebigen Grads von Strukturiertheit für den lesenden Zugriff unter besonderer Berücksichtigung der Evolutionsfähigkeit. Die Betonung großer Datenbestände ist dahingehend zu verstehen, daß durch die Abstraktion von Daten- zur Metadatenebene die Möglichkeit der semantischen Integration feingranularer Datenstrukturen verloren geht.

4.4 Semistrukturierte Daten und XML

Wie in Abschnitt 1.3 beschrieben, sind die Daten verschiedener Quellen oft strukurell, schematisch und semantisch heterogen. So könnte der Name einer Person in einem Attribut gespeichert sein, oder aber sich in Vorname und Nachname aufteilen. Die Telefonnummer einer Person kann als Integer oder als String modelliert werden. Zur Erreichung einer eng-gekoppelten Integration wird zur Überwindung dieser Heterogenität auf der Schemaebene oft eine Schemaintegration durchgeführt – ein Verfahren, das allerdinge einige Nachteile beinhaltet (siehe Abschnitt 4.1).

Eine Alternative dazu bieten die sogenannten *semistrukturierten Datenmodelle*. Die Idee ist dabei die Verwendung eines sehr einfachen Datenmodells, in das die Daten aus unterschiedlichsten Modellen relativ einfach umgewandelt werden können. Semistrukturierte Daten haben kein Schema, sondern sind selbstbeschreibend, d.h. tragen die "Metainformation" über ihren Dateninhalt ebenfalls in der Form von Daten. Die Integration von Daten aus unterschiedlichen Quellen besteht einfach in der jeweiligen Umwandlung in ein semistrukturiertes Modell; diese werden dann ohne weitere Integrationsschritte aneinandergefügt und können von da an mit einer einheitlichen Anfragesprache bearbeitet werden.

4.4.1 Das OEM Datenmodell

Ein bekanntes semistrukturiertes Modell ist das Object Exchange Modell (OEM). Es wurde ursprünglich im TSIMMIS Projekt vorgestellt (siehe 5.5), später dann aber auch beispielsweise im LORE System [30] benutzt. OEM ist ein einfaches Objektmodell. Daten werden in Form von Graphen dargestellt, wobei die Knoten mit Werten und die Kanten mit *Labels* beschriftet sind. Die Labels übernehmen dabei die Funktion von Metadaten. Es gibt im Unterschied zu klassischen objekt-orientierten Datenmodellen keine Methoden, keine Vererbung und auch keine Klassen. Man sieht leicht, daß die meisten Datenmodelle in eine OEM Repräsentation umgewandelt werden können, allerdings unter der Inkaufnahme eines Verlustes an semantischer Information.

Ein OEM-Objekt besitzt eine Objektidentität, einen Bezeichner (Label) und einen Wert eines spezifizierten Typs.

$$(oid, label, type, value)$$

Beispiel:

$$(\&123, temperatur - in - Celsius, integer, 25)$$

Es gibt zwei Kategorien von Objekten, die atomaren und die komplexen Objekte. Ein atomares Objekt zeichnet sich dadurch aus, daß es Werte speichert, die den Basistypen angehören, wie beispielsweise *Integer* oder *String*. Ein komplexes Objekt hingegen hat als Typ *set* und als Wert eine Menge von Objektreferenzen. Mit Hilfe der komplexen Objekte ist es möglich, beliebig tiefe hierarchische Strukturen aufzubauen. Das können im einfachen Fall Bäume sein, aber auch gerichtete zyklische Graphen sind möglich. Die (mögliche) Repräsentation zweier verschieden strukturierter Personendaten, die aus verschiedenen Quellen stammen, sieht man in Abb. Fig. 4.4.

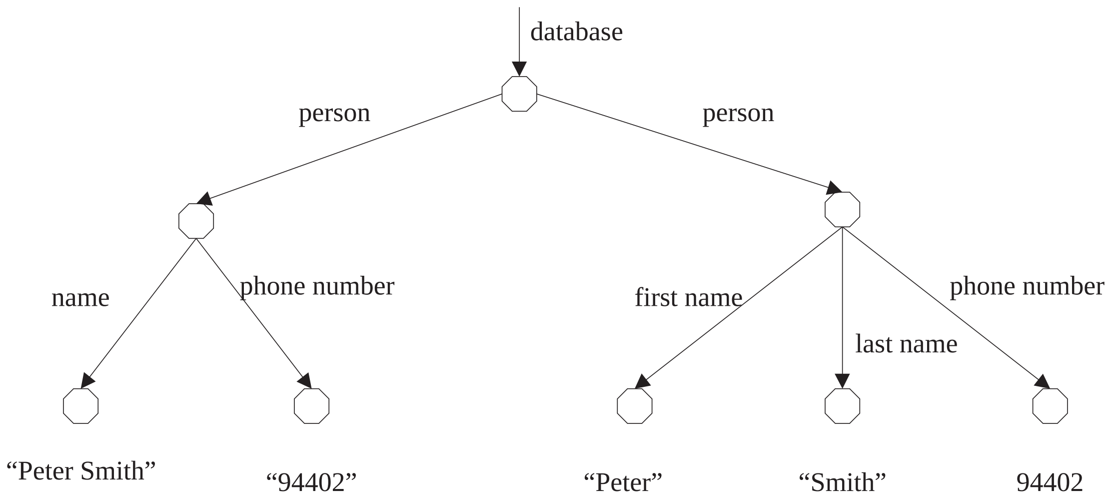

Abbildung 4.4. Darstellung zweier strukturell heterogener Datensätze als OEM Objekte

Eine interessante Eigenschaft des OEM ist an diesem Bild schon ansatzweise zu erkennen. Die Label müssen für Menschen gut verständlich sein. Der Name einer Person sollte deshalb *name* oder besser noch *person_name* heißen. Nur so kann man die Bedeutung der Daten direkt aus den Daten, also aus den Labeln, ablesen - und mehr Informationen hat man nicht zur Verfügung. Man benötigt also “gut gewählte” Label, um sinnvolle Anfragen zu stellen und um bestimmte Elemente aus einer OEM Struktur zu selektieren. Wenn man beispielsweise nach Personen und deren Namen sucht, ist man darauf angewiesen, daß die gesuchten Datenstrukturen die Label person oder name, möglicherweise auch in Abwandlungen, beinhalten. Die möglichen Abwandlungen muß man z.B. mit Hilfe von regulären Ausdrücken beschreiben. Wenn man davon ausgeht, daß ein Benutzer detailiiertes Wissen über die Daten und deren Quellen besitzt, kann das eine sehr effektive Art der Datenintegration sein. Denn es ist ja nicht nötig, vorher eine oft aufwendige Schemaintegration durchzuführen. Die einzige Vorarbeit, die nötig wird, ist die Übersetzung der Quelldaten in eine OEM-Struktur. Das ist jedoch in den meisten Fällen relativ einfach; es kommt dabei vor allem auf die gute Lesbarkeit und Aussagefähigkeit der Label an. Diese “Einfachheit” beschränkt sich allerdings auf Informationssysteme die ausschließlich lesend auf Quellen zugreifen.

Für den Fall, daß auf die Quellen auch schreibend zugegriffen werden soll, wird sich ein semistrukturiertes Datenmodell im allgemeinen schlecht eignen. Beispielsweise lassen sich keinerlei Integritätsbedingungen darstellen. Im weiteren soll jedoch nur der einfachere Fall betrachtet werden, bei dem auf die Quellen ausschließlich lesend zugegriffen wird.

Wie bereits erwähnt, gibt es eine Reihe von Ansätzen und Projekten, die sich mit dieser Art von Daten beschäftigen. Besonders populär ist sie für Projekte, die sich mit der Integration von Daten aus dem World Wide Web beschäftigen. Das hängt natürlich mit der relativen Unstrukturiertheit vieler Daten im WWW zusammen. Jede HTML Seite hat zwar meistens gewisse Strukturen, diese sind jedoch sehr flexibel und entsprechen nicht einem starren Schema im relationalen Sinne.

Entscheidend für den Einsatz eines integrierten Informationssystems ist auch die Fähigkeit, mit Hilfe einer Anfragesprache gezielt Informationen zu extrahieren. Im LORE Projekt wurde dazu eine Anfragesprache spezifiziert, die es möglich macht, semistrukturierte Daten mit der vertrauten "Select-From-Where"-Syntax zu selektieren [1].

4.4.2 XML

Einer der neusten Hypes innerhalb der IT Welt ist XML. XML ist in gewissem Sinne ein semistrukturiertes Datenmodell. Dieser Vergleich ist allerdings problematisch, da XML eigentlich eine Mark-up Language und kein Datenmodell ist [45]. Aber man kann deutlich einige Gemeinsamkeiten von semistrukturierten Datenmodellen und XML finden. Ein XML File ist eine Sequenz von Elementen, die entweder wieder Elemente oder einfachen Text beinhalten. Das ist offensichtlich ähnlich zu OEM. In XML lassen sich ebenfalls beliebig tiefe hierarchische Strukturen definieren. Diesen Vergleich könnte man auf einige andere Details ausweiten und würde weitere Parallelen entdecken. Im Gegensatz zu OEM kann ein XML Dokument auch eine optionale Grammatikbeschreibung besitzen, die sogenannte Document Type Definition (DTD). Sie beschreiben die Struktur eines XML Dokuments. Dabei gibt es "Kann"- und "Muß"-Bestimmungen. In ihnen wird beschrieben, welche Elemente auftreten und wie sie ineinander verschachtelt sein dürfen oder müssen. Man kann DTDs mit Schemata vergleichen, allerdings können sie weniger streng formuliert sein als es Schemata im Sinne eines RDBMS sind. XML stellt also eine Möglichkeit dar, semistrukturierte Daten zu repräsentieren. Aber mit XML können auch sehr streng und hierarchisch strukturierte Daten repräsentiert werden. Nach der Spezifikation des XML-Standards hat man bald erkannt, daß man Mittel benötigt mit denen man Information aus XML-Dokumenten extrahieren kann. Aus diesem Grunde gibt es beispielsweise beim World Wide Web Konsortium (W3C) ein rege Diskussion über Anfragesprachen für XML Dokumente.

4.4.3 Diskussion

Es ist offensichtlich, daß man mit Hilfe semistrukturierter Datenmodelle und entsprechenden Hilfsmitteln wie Anfragesprachen einen ganz besonderen Weg der Datenintegration gehen kann. Es ist dabei nicht notwendig, einen aufwendigen Integrationsprozeß durchzuführen, sondern man betrachtet die Daten nach einer Übersetzung in ein gemeinsames OEM-Modell als eine einzige virtuelle und semistrukturierte Quelle. In bezug auf Autonomie unterscheiden sich solche Ansätze nicht grundsätzlich von anderen Formen integrierter Informationssysteme. Offensichtlich behalten die Quellen die volle Entwurfsautonomie, da sie zur Entwurfszeit keine Wissen über eine spätere Integration besitzen müssen. Darüber hinaus ist es in "reinen", semistrukturierten Integrationssystemen auch nicht tragisch, wenn strukturierte Datenquellen ihr Schema ändern, da das Integrationssystem diese Schemata nicht explizit, sondern nur implizit behandelt. Dies trifft aber z.B nur eingeschränkt auf TSIMMIS zu, da die dort definierten, integrierenden OEM Views in der Regel explizit Bezug auf Quell-spezifische Label nehmen.

Große Vorteile, aber auch Nachteile, kann ein semistrukturierter Ansatz in bezug auf Heterogenität bieten. Viele der Probleme, die bei anderen Formen der Integration auftauchen, werden einfach ausgeklammert bzw. den Fähigkeiten der Anwender überlassen. Die Heterogenität der verwendeten Datenmodelle muß wie üblich durch eine, in der Regel sehr einfach vorzunehmende, Übersetzung in ein OEM-Modell behandelt werden. Alle weiteren logischen Heterogenitäten werden nicht weiter aufgelöst - mit ihnen muß der Anwender in jeder einelnen Anfrage selber umgehen.

In bezug auf Verteilung des Informationssystems bietet eine semistrukturiertes Datenmodell keine spezifischen Vor- oder Nachteile. Je nach Art der Implementation eines solchen Systems können verschiedenen Verteilungsstrategien verfolgt werden.

Die Verwendung eines semistrukturierten Datenmodells kann die Evolutionsfähigkeit des gesamten Systems aber auch der Einzelkomponenten positiv beeinflussen. Aufgrund der Tatsache, daß die Daten in der OEM Welt ihre Struktur selbst beschreiben, ist es offensichtlich, daß eine Veränderung der Struktur der Daten keinen wesentlichen Einfluß auf ein System haben wird, das semistrukturierte Daten speichert bzw. integriert. Denn egal wie die Struktur der Daten aussieht, ein semistrukturiertes System muß mit der Tatsache umgehen, daß es außer den Daten selbst keine andere Strukturinformationsquelle besitzt. Das bedeutet, daß selbst wenn sich die Schemata der lokalen Quellen verändern, werden diese Veränderungen ja durch die Strukturinformation innerhalb des OEM modelliert und stehen so dem Endsystem zur Verfügung. Letztlich wirkt sich eine solche Änderung also nur als eine Änderung der Daten aus.

Eine Änderung in der Konfiguration eines solchen Informationssystems (z.B. Anzahl und Standort von Komponenten) wird auch relativ leicht zu realisieren sein, wobei das im hohem Maße von der konkreten Konstellation des Informationssystems abhängig ist.

Zusammenfassend läßt sich sagen, daß die Schnittstellen zwischen dem Hauptsystem und den einzelnen Quellen sehr einfach gestaltet sein können. Werden die Daten der lokalen Quellen in das System repliziert, ist das die sicher einfachste Variante. Der einzig kritische Teil bei einer solchen Konstellation ist die Übersetzung der lokalen Quellen in ein semistrukturiertes Datenmodell (OEM). Da OEM aber eine denkbar einfache Struktur impliziert, wird dieser Schritt in den meisten Fällen keine größeren Probleme bereiten. Bleiben die Daten hingegen in den jeweils lokalen Quellen, wird eine ständige Verfügbarkeit der lokalen Quellen erforderlich sowie ein kontinuierliches und vor allem auch zweiseitiges Arbeiten der Übersetzungsalgorithmen nötig. Denn Anfragen, die gegen OEM Daten gestellt werden, müssen dann zur Laufzeit in Anfragen an die lokalen Quellen umgewandelt werden. Zur Lösung solcher Probleme gibt es verschiedene Ansätze (siehe z.B. TSIMMIS). Diese Probleme entstehen aber in ähnlicher Weise bei praktisch allen Integrationsmethoden.

Die semistrukturierten Datenmodelle sind also ein Werkzeug, das die Integration von unterschiedlichen Datenquellen erheblich erleichtern kann. Diese Erleichterung wird aber vor allem dadurch erkauft, daß der Integrationsprozeß verlagert wird und in die Kompetenz der Anwenders gelegt wird. Wenn man davon ausgeht, daß ein Anwender genügend Wissen über den Diskursbereich und über die vorhandenen Daten besitzen wird, ist das aber ein durchaus plausibler Schritt.

Ob ein semi-strukturiertes Datenmodell die beste Wahl ist, hängt von dem konkreten Einsatzfall ab. Semistrukturierte Daten sind eben nicht voll strukturierte Daten. Und der Integrationsprozeß, der ja mehr oder weniger vom Anwender geleistet werden muß, kann natürlich bei sehr komplexen und dem Anwender nicht vertrauten Problembereichen scheitern. Insbesondere erscheint ein Integrationsprozeß über die Schemata in stabilen bzw. starren Umgebungen oftmals effektiver. Handelt es sich jedoch um eine dynamische Umgebung, also verändert sich oft die Anzahl der

Quellen und verändern sich auch die Schemata der Quellen, so kann die Benutzung der semistrukturierten Datenmodelle doch einige Vorteile bieten.

4.5 Logische Anfragebearbeitung

Wir betrachten hier die Anfragebearbeitung in eng integrierten Systemen. Viele der Konzepte lassen sich aber auch auf lose integrierte Systeme anwenden - im wesentlichen fällt dort nur der Schritt der Anfrage-Modifikation (siehe unten) weg. Die Darstellung orientiert sich an [52].

Eng integrierte Systeme besitzen ein globales Schema, gegen das die Anfragen gestellt werden. Dieses Schema wird durch die Integration der Exportschemata der einzelnen lokalen Datenbanken abgeleitet. Der globale Anfrageoptimierer arbeitet hier in einer Umgebung bestehend aus dem Mediator (in der Literatur auch als Front-End-System oder Global Data Manager bezeichnet) mit seinen spezifischen Fähigkeiten zur Ausführung von Anfrageoperationen und den lokalen Datenbanksystemen mit ihren eigenen Fähigkeiten zur Ausführung von Anfragen.

Die Ausführung von Anfragen in integrierten Systemen ist ähnlich den in verteilten Datenbanksystemen. Die Autonomie und Heterogenität der einzelnen Komponenten erschweren die gegebene Aufgabe jedoch zusätzlich. Dabei sind folgende Punkte besonders zu beachten:

- Der globale Anfrageoptimierer hat keinerlei Einfluß auf die Ausführung von Anfragen in den Komponentensystemen, insbesondere auch nicht auf das Zwischenspeichern von Anfrageergebnissen.

- Die Informationen über die Kosten einer Anfrageausführung in den lokalen Systemen sind meist nur unvollständig vorhanden und können sich zum Teil stark voneinander unterscheiden.

- Die Fähigkeiten der einzelnen Systeme hinsichtlich der Ausführung von Anfragen können sich stark voneinander unterscheiden.

Das generelle Vorgehen zur Ausführung einer Anfrage in einem integrierten Informationssystem besteht zunächst in einer Zerlegung der Anfrage gegen das integrierte Schema in (normalerweise) mehrere Anfragen an die einzelnen Exportschemata, die dann von den lokalen Systemen ausgeführt werden.

Im ersten Schritt erfolgt zunächst eine Modifikation der Anfrage. Dabei werden die globalen Namen des integrierten Schemas durch die entsprechenden Namen der Exportschemata ersetzt. Dies entspricht im wesentlichen der Umkehrung des Integrationsprozesses, aus dem auch die notwendigen Informationen hierfür gewonnen wurden. Hierbei werden sogenannte Auflösungsfunktionen betrachtet, die für ein gegebenes Attribut angeben, wie sein Wert abgeleitet wird, sollte das entsprechende Objekt in mehreren Exportschemata vorkommen.

Als Ergebnis der Modifikation erhält man in aller Regel mehrere Teilanfragen (Modified Query). Diese können weiterhin Daten aus mehr als einer Datenquelle referenzieren und sind deshalb nicht direkt von einem der lokalen Datenbanksysteme ausführbar. Zur Beantwortung der Anfrage müssen sie deshalb weiter zerlegt werden. Hierbei lassen sich zwei Ansätze unterscheiden.

4.5.1 Multiple-Query Ansatz

Beim Multiple-Query-Ansatz werden zunächst sämtliche Modified Queries, sofern dies nötig ist, in sog. Single-Database Queries zerlegt, die dann jeweils direkt von einem der lokalen Datenbanksysteme ausgeführt werden können. Die Ergebnisse der Single-Database Queries müssen dann im Mediator durch eine sog. Assembling Query zum eigentlichen Ergebnis der Modified Query zusammengefügt werden.

Während der Dekomposition der Modified Queries werden zwei Typen von Prädikaten identifiziert. Beim ersten Typ handelt es sich um Prädikate, deren Attribute alle aus einer lokalen Datenbank stammen. Prädikate dieses Typs werden in die entsprechende Single-Database Query übernommen, damit sie vom lokalen Datenbanksystem ausgewertet werden. Der zweite Typ von Prädikaten enthält Attribute aus mehr als einer Datenbank. Diese Prädikate können deshalb erst innerhalb der Assembling Query vom Mediator ausgewertet werden. Aus diesem Grund müssen die Werte dieser Attribute zusätzlich als Ergebnisse der entsprechenden Single-Database Queries geliefert werden.

Durch die Zerlegung der Modified Queries in Single-Database Queries können Joins lokal in den Datenbanksystemen ausgeführt werden, unter Ausnutzung der dort vorhandenen Optimierungsmöglichkeiten, wie z.B. schnelle Zugriffspfade. Voraussetzung hierfür ist natürlich, daß die am Join beteiligten Relationen alle aus dem lokalen System stammen.

Als weiteres Ergebnis der Dekomposition können für jedes lokale Datenbanksystem mehrere Single-Database Queries generiert werden. Man kann nun versuchen, diese so zusammenzufügen, daß die Anzahl der Zugriffe auf die lokalen Relationen minimiert wird, um somit die Ausführungszeit zu verkürzen. Man bezeichnet dies als *Multi Query Optimisation*. Da die lokalen Optimierer gewöhnlich nur für die Optimierung einzelner Anfragen ausgerichtet sind, muß dieser Schritt ebenfalls im Mediator erfolgen.

4.5.2 Single Query Ansatz

Beim Single-Query-Ansatz werden die einzelnen Modified Queries isoliert voneinander betrachtet und ausgeführt. Sie müssen auch hier wieder in mehrere Teilanfragen zerlegt werden, sofern sie Daten aus mehr als einer Datenbank referenzieren. Verglichen mit dem Multiple-Query-Ansatz besteht der größte Vorteil dieser Methode darin, daß die Anzahl der zu verwaltenden Teilanfragen überschaubarer bleibt, da man zu einem Zeitpunkt ja jeweils nur eine Modified Query betrachtet. Es bleibt auch bei diesem Ansatz weiterhin möglich, die lokalen Optimierungsmöglichkeiten für Joins auszunutzen. Die einzelnen Teilergebnisse lassen sich später leicht über eine einfache Union vereinigen.

Der größte Nachteil des Single-Query-Ansatzes ist sicher die Tatsache, daß bestimmte Relationen, sofern sie in mehr als einer Modified Query vorkommen, mehrfach referenziert werden. Im Multiple-Query-Ansatz wird versucht, dies durch die Multiple Query Optimization zu verhindern. Beim Single-Query-Ansatz ist dies jedoch nicht möglich.

4.5.3 Ein Beispiel

Wir betrachten als Beispiel zunächst die Ausführung eines einfachen Joins über zwei Relationen aus unterschiedlichen Datenquellen. Dabei muß man zuerst entscheiden, an welcher Stelle der eigentliche Join ausgeführt werden soll. Man kann den Join in einem der lokalen Systeme oder direkt im Mediator ausführen lassen.

Zur Ausführung des Joins in einem der lokalen Systeme ist es notwendig, die Daten der zweiten, nicht lokalen Relation, in das ausführende System zu übertragen. Dazu wird in aller Regel die Genehmigung zur Erzeugung von temporären Relationen benötigt. Um die Menge an übertragenen Daten so gering wie möglich zu halten, können zum Beispiel die von Semi-Join her bekannten Konzepte verwendet werden. Auf die dann im lokalen System verwendete Join-Methode (Nested Loop Join, Semi-Join, Sort Merge Join, Hash Join etc.) hat man aufgrund der Autonomie der Systeme keinen Einfluß mehr.

Soll der Join direkt im Mediator ausgeführt werden, so müssen zuerst die benötigten Relationen in den Mediator geladen werden. Dies umgeht das Problem der temporären Speicherung in den lokalen Systemen und stellt sicher, daß jede Relation nur einmal von jeder Komponente referenziert wird. Man kann nun natürlich die zu verwendende Join-Methode selber bestimmen. Der Aufwand für den Mediator ist bei dieser Methode natürlich wesentlich größer als bei der vorherigen.

Einige der Modified Queries können mehr als einen Join enthalten. Es ist deshalb wichtig Techniken zu finden, die diese Art von Anfragen optimieren können. Eine mögliche solcher Techniken besteht im wesentlichen aus zwei Schritten.

Im ersten Schritt wird ein kostenbasierter Optimierungsalgorithmus verwendet, um einen optimalen sog. Left Deep Join Tree zu erzeugen. Ein solcher Algorithmus existiert zum Beispiel in System R. In einem solchen Left Deep Join Tree erfolgt die Ausführung der Joins sequentiell (bottom-up), so daß jeweils auf die Beendigung jedes einzelnen Joins gewartet werden muß.

In einem zweiten Schritt wird nun versucht, den Left Deep Join Tree so zu verändern, daß einzelne Joins parallel zueinander ausgeführt werden können, ohne jedoch die absoluten Kosten der Ausführung zu erhöhen. Somit kann die Antwortzeit bei gleichbleibenden Kosten verringert werden. Dabei muß jedoch sichergestellt sein, daß das Gesamtergebnis durch die Umstellung der Joins nicht verändert wird.

Kapitel 5

Konkrete Systeme

In diesem Kapitel wird nun eine Reihe von konkreten Systemen kurz dargestellt, die alle von uns als "Mediator-basiert" klassifiziert wurden. Die einzelnen Projekte haben dabei unterschiedliche Schwerpunkte:

- *Garlic* konzentriert sich auf Anfrageoptimierung in Umgebungen mit nicht - DBMS Quellen
- Das *Information Manifold* verwendet einen unorthordoxen, aber mächtigen Mechanismus zur semantischen Beschreibung des Inhalts von Quellen.
- *MeBro* integriert keinerlei Daten, sondern ist ein flexibles System zur Vermittlung von Verweisen, basierend auf Metadaten in verschiedenen Dimensionen.
- *SIMS* zeichnet sich insbesondere durch die Verwendung eines sehr ausdruckstarken Datenmodells, einer *Description Logic*, sowohl für Anfragen als auch für das globale Schema aus.
- *TSIMMIS* dagegen verwendet ein semantisch sehr armes, semi-strukturiertes Datenmodell. Schwerpunkt ist hier die flexible Integration von Daten.
- *Mariposa* schließlich unterscheidet sich durch seinen ökonomisch motiverten Ansatz gänzlich von allen andern Systemen.

Natürlich konnten in dem Seminar nicht annähernd alle Projekte im Bereich Informationsintegration bzw. Mediation behandelt werden. Es fehlen beispielsweise DISCO ([46]), InfoMaster ([11]), DIOM ([29]), COIN ([12]), Araneus ([4]) etc. Eine (ebenfalls unvollständige) Liste kann man im WWW finden (http://salmon.cs.tu-berlin.de:8888/leser/research/cs_int_projects.html).

Im folgenden werden die Projekte jeweils kurz charakterisiert und dabei jeweils unter den in den vorherigen Kapiteln beschriebene Aspekten betrachtet. Insbesondere erfolgt dabei:

- eine architekturelle Einordnung im Vergleich zur klassischen FDBMS Architektur (Kapitel 2.1) und zur generischeren "Mediator" Architektur (Kapitel 2.2)
- eine Betrachtung, inwieweit die Systeme mit Verteilung, Autonomie und Heterogenität der Komponenten umgehen können.

- eine kritische Analyse der Evolutionsfähigkeit der Ansätze in Szenarien wie in Kapitel 1.4 beschrieben.

Um die erwünschte Klärung herbeizufüren werden am Ende jedes Abschnitts **in jeweils einem Satz** die Begriffe **Mediator** und **Wrapper** im Kontext der Projekte definiert.

5.1 Garlic

Garlic ist ein von IBM am Almaden Research Center durchgeführtes Projekt, dessen Ziel die Schaffung eines multimedialen Informationssystems zur Integration unterschiedlicher Datenquellen ist. Der Zugriff auf die Daten kann hier ausschließlich lesend erfolgen. Dabei sollen die Fähigkeiten der einzelnen Quellen hinsichtlich der Abfrage und Bearbeitung der von ihnen gespeicherten Objekte so gut wie möglich unterstützt werden. Es ist deshalb ein universeller Anfrageoptimierer entwickelt worden, dem die Fähigkeiten der einzelnen Systeme mitgeteilt werden können. Diese Informationen werden dann vom Optimierer zur Erstellung eines optimalen Ausführungsplanes für Anfragen über mehrere Datenquellen mit unterschiedlichen Fähigkeiten ausgenutzt. Da die Fähigkeiten der einzelnen Systeme im Optimierer nicht fest verdrahtet sind, bleibt er unanfällig gegen Änderungen, wie das Hinzufügen neuer Datenquellen mit erweiterten Anfragefunktionalitäten. Dies wird durch die Verwendung eines regelbasierten Systems zur Generierung von Ausführungsplänen ermöglicht.

Die Daten sollen zur Integration nicht an eine zentrale Stelle kopiert werden, sondern einzig und allein über die eingebundenen Datenquellen zugänglich sein. Auch die Autonomie der einzelnen Systeme soll vollständig erhalten bleiben. Ein weiteres Ziel ist die leichte Erweiterbarkeit des Systems um zusätzliche Datenquellen (Skalierbarkeit).

Als Datenquellen kommen sowohl relationale als auch objektorientierte Datenbanken, sowie Bild- und Videodatenbanken, textbasierte Systeme und WWW-Suchmaschinen in Frage. Da viele der Daten von Natur aus als Objekte modelliert sind, wird von Garlic ein objektorientiertes Datenmodell zusammen mit einer objektorientierten Anfragesprache verwendet. Bei dieser Sprache handelt es sich um eine objektorientierte Erweiterung von SQL. Das Datenmodell ist an den ODMG-93-Standard angelehnt.

Die Exportschemata der einzubindenden Datenquellen werden mit Hilfe des verwendeten Datenmodells beschrieben, um dann zum globalen Schema integriert zu werden. Es handelt sich also um ein eng integriertes System mit integriertem Schema. Diese Informationen werden im Systemkatalog gespeichert. Man löst so die Probleme der logischen Heterogenität bezüglich der verwendeten Datenmodelle. Wie das genaue Vorgehen bei der Integration der Exportschemata aussieht und wie semantische Heterogenitäten aufgelöst werden, wird von den Autoren in ihren Publikationen nicht genauer beschrieben. Auch findet man keine expliziten Angaben darüber, ob Systeme Top-Down oder Bottom-Up entwickelt werden, wobei das beschriebene Vorgehen eher einer Bottom-Up-Entwicklung gleicht.

Relevante Änderungen an einem lokalen Schema, die zur Änderung des Exportschemas führen, ziehen dann eine erneute Integration zur Bildung eines globalen Schema nach sich. Gleiches gilt für das Einbinden neuer Datenquellen.

Der Query Language Processor erhält als Eingabe eine globale Anfrage. Seine Aufgabe ist es nun, einen bezüglich des verwendeten Kostenmodells optimalen Ausführungsplan zu generieren.

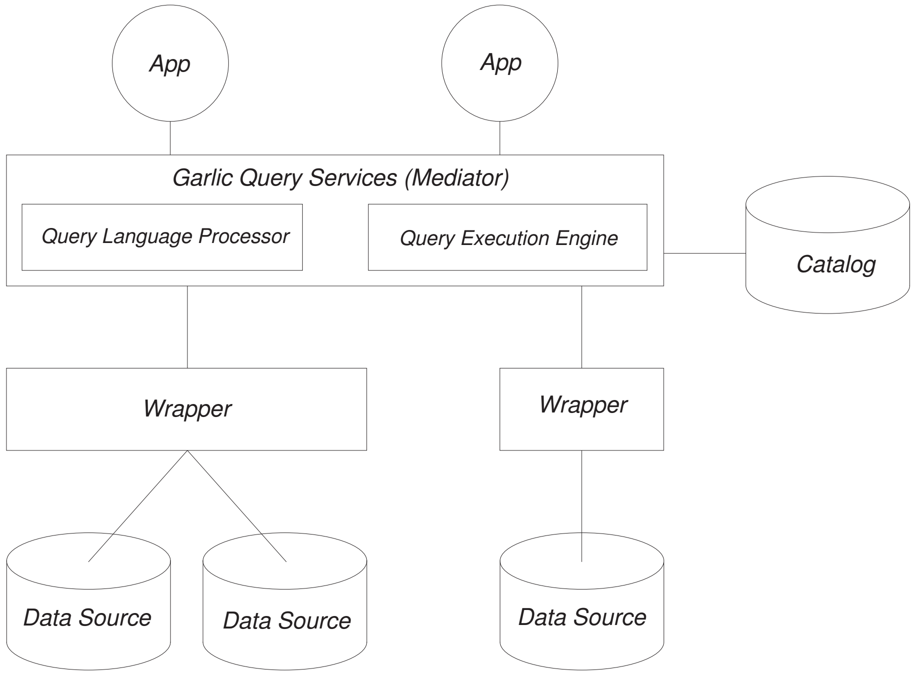

Abbildung 5.1. Garlic Mediator-Wrapper Architektur

Nachdem der optimale Ausführungsplan bestimmt worden ist, wird seine Ausführung durch die Query Execution Engine koordiniert.

Oberhalb jeder Datenquelle befindet sich ein Wrapper, dessen Aufgabe es ist, die Datenquelle in das System einzubinden. Er ermöglicht den Zugriff auf die lokalen Daten in Form von Objekten, den Zugriff auf deren Attribute und den Aufruf von Methoden auf den Objekten. Zusätzlich wird er in die Planung eines optimalen Ausführungsplans mit einbezogen und muß dann während der Ausführungsphase die von ihm zuvor in der Planungsphase übernommenen Teilaufgaben ausführen. Da der Wrapper sämtliche Zugriffe des Systems auf die Daten seiner Datenquellen und die Ausführung von Anfragen koordiniert, werden so die Heterogenitäten bezüglich verschiedener Anfragesprachen gelöst.

Eine der weiteren wesentlichen Aufgaben eines Wrappers ist die Beschreibung der Fähigkeiten der von ihm verwalteten Datenquellen hinsichtlich der Ausführung von Anfragen. Der Wrapper muß dabei nicht die volle Funktionalität seiner Datenquelle zur Verfügung stellen, um deren Ausführungsautonomie erhalten zu können. Die Quelle bzw. der Wrapper kann selbst entscheiden, welche Anfragen ausgeführt werden können oder sollen. Diese Entscheidung kann dynamisch während der Planungsphase des Ausführungsplans erfolgen.

Der Wrapper muß jedoch eine gewisse Grundfunktionalität anbieten, die zumindest den Zugriff auf die von der Quelle verwalteten Objekte bzw. deren Attribute ermöglicht. Dies soll die schnelle Entwicklung einfacher Wrapper ermöglichen, deren Funktionsumfang dann im Laufe der Zeit einfach erweitert werden kann.

Teilaufgaben, die von einem Wrapper nicht ausgeführt werden, wie z.B. Sortierung der Daten, müssen von der Query Execution Engine übernommen werden. Gleiches gilt für Joins, die von keinem der lokalen Systeme ausgeführt werden können.

Wie bereits oben erwähnt, ermöglicht die Verwendung von *Wrappern* eine leichte Integration neuer Datenquellen, da alle Informationen in einem dynamisch verwendeten Repository, dem System Katalog, gespeichert werden. Dies erfordert dann allerdings in aller Regel eine neue Integration des globalen Schemas.

Der Query Language Processor und die Query Execution Engine bilden in Garlic gemeinsam einen *Mediator*, der die Ausführungsplanung und die Ausführung von Teilaufgaben, die von den Quellen nicht erfüllt werden können, übernimmt.

5.2 Information Manifold

Das Information Manifold (IM) wurde in einem Forschungsprojekt bei AT&T in den Jahren 1995 und 1996 entwickelt und dabei auch als Prototyp implementiert. Es ging darum, einem Benutzer im World Wide Web eine Oberfläche zu bieten, über die er auf eine große Anzahl (im Prototyp über 100) heterogener Datenquellen mit ähnlichen Informationen zugreifen kann. Dabei braucht der Benutzer nur angeben, welche Informationen er gern hätte und nicht wo diese liegen.

Das globale Schema des IM bilden die Klassen und Relationen des sogenannten *World Views*. Über das World View Interface kann der Benutzer auf das IM zugreifen. Dabei waren die eigentlichen Forschungsschwerpunkte im IM einerseits die Formulierung von äußerst differenzierten deklarativen Beschreibungen der zur Verfügung stehenden Datenquellen, die die einzubindenden Informationsquellen in Beziehung setzen zu den Klassen und Relationen des World Views, und andererseits der Query Planning Algorithmus, der eben diese Beschreibungen benutzt, um möglichst effektiv Anfragen an überflüssige Datenquellen und Übertragung nicht relevanter Daten zu vermeiden.

5.2.1 Die Architektur des Information Manifold

Wie in Fig. 5.2 (nach [26]) zu sehen, greift ein Benutzer des IM über die WWW basierte Benutzerschnittstelle (*User Interface*) unter Zuhilfenahme der Informationen des *World Views* auf das IM zu. Der *Plan Generator* erstellt mit Hilfe der Quellenbeschreibungen (*Source Descriptions*) einen Ausführungsplan (*Execution Plan*), der von der *Execution Engine* ausgeführt wird. Jede Quellenart hat ihr Zugriffsprogramm, welches die Verarbeitung dieses Datenformates ermöglicht. Der eigentliche Zugriff auf die Datenquellen erfolgt schließlich über das Internet. Im folgenden wird kurz auf die Forschungsschwerpunkte, also den World View, die Source Descriptions und den Plan Generator eingegangen. Alle restlichen Komponenten des IM sind den in anderen Systemen sehr ähnlich.

5.2.2 Die Komponenten des Information Manifold

Der "World View"

Der World View baut auf ein relationales Datenmodell auf, das mit objektorientierten Merkmalen erweitert wurde, die das Beschreiben und Treffen von Entscheidungen erleichtern sollen.

Im World View sind keinerlei Daten enthalten, er ist rein konzeptuell und besteht aus einer Anzahl virtueller Relationen beliebiger Ordnung über einer Anzahl virtueller Klassen, die in einer

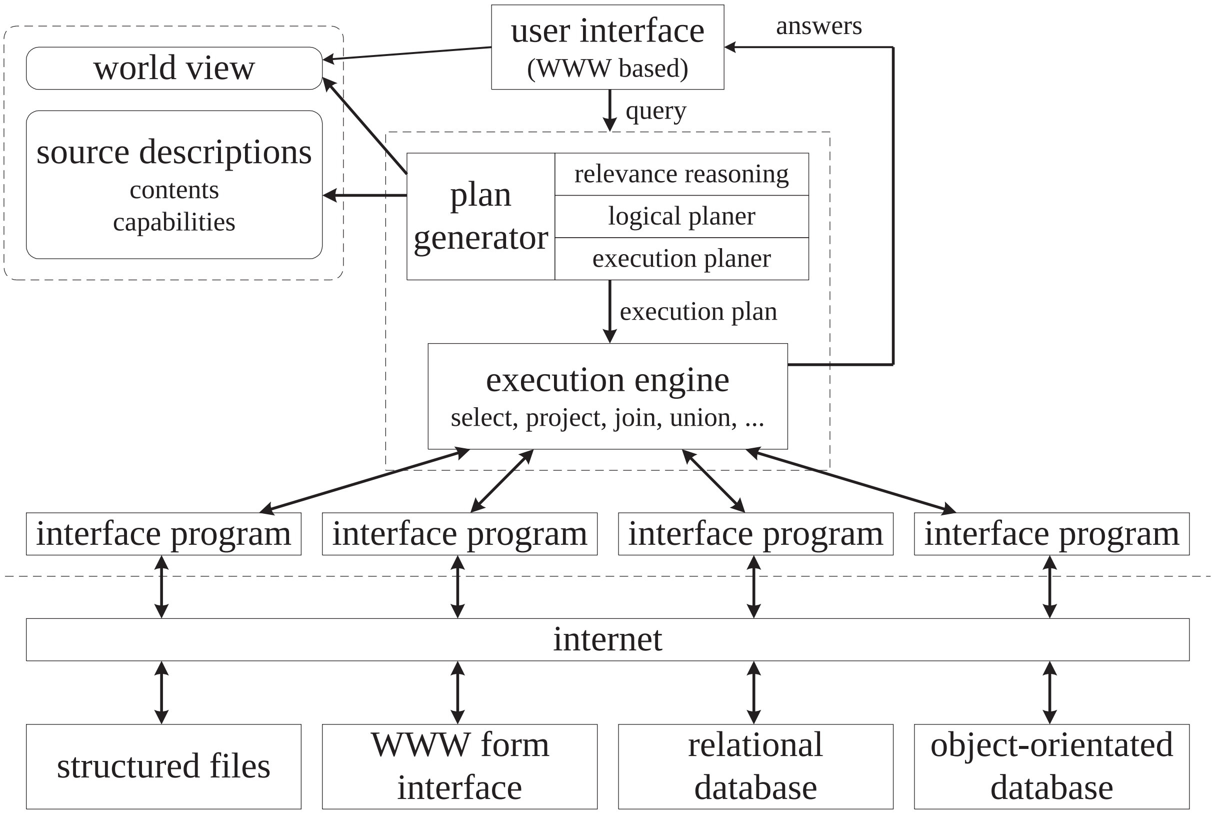

Abbildung 5.2. Die Architektur des Information Manifold

Klassenhierarchie geordnet sind. Zu jeder Klasse gibt es einen Satz von Attributen.

Als Datenmodell für die Klassen, Relationen und Attribute des World View wird die ausdrucksstarke Beschreibungslogik CLASSIC verwendet. Wäre zum Beispiel Car eine Klasse mit den Attributen $model$, $year$, $price$ und $owner$, so könnte auf diese Klasse mit der Relation

$$Car(model, year, price, owner)$$

zugegriffen werden, außerdem könnte mit $Car(mo)$ geprüft werden, ob das eingegebene Objekt mo überhaupt ein Auto ist. Das Konzept der Klasse wird dabei eigentlich nur benutzt, um durch in Beziehung setzen der Klassen in *"is a"*-Beziehungen eine Art Klassenhierarchie aufzubauen. So hat man zum Beispiel die Möglichkeit klar zu machen, daß eine Quelle, die Informationen über japanische Autos enthält, natürlich Informationen über ausländische Autos enthält, indem man im World View festlegt, daß japanische Autos ausländische Autos sind.

Der Benutzer kann mit Hilfe der im World View vorgegebenen Klassen und Relationen konjunktive Anfragen (also mit UND verknüpfte Relationen und Constraints) an das IM stellen. Durch die Angabe von Constraints hat er die Möglichkeit, die Ergebnismenge einzuschränken (zum Beispiel: $price \leq 5.000$).

Die Quellenbeschreibungen

Die Beschreibungen der Datenquellen setzen die Semantik der Quellen in Beziehung zu den Klassen und Relationen des World Views. Dabei sollen auch feinste Unterschiede zwischen den verschiedenen Informationsquellen ausgedrückt werden, da dies zur Entscheidungsfindung bei der Anfrageoptimierung benötigt wird. Wie im World View wird auch hier mit CLASSIC gearbeitet. Durch diese Quellenbeschreibungen wird es ermöglicht, die Anfragen an den World View in Anfragen an die eigentlichen Quellen zu übersetzen. Beschrieben werden dabei, neben den Inhalten der Quellen, auch ihre Fähigkeiten, Anfragen zu beantworten. Eine solche Beschreibung der Fähigkeiten besteht aus dem Tupel

$$(\{Eingabe\}, \{Ausgabe\}, \{Selektion\}, Min, Max)$$

wobei $\{Eingabe\}$ die Menge der Eingabeparameter ist, $\{Ausgabe\}$ die Menge der Ausgabeparameter, $\{Selektion\}$ die Menge der Parameter auf die die Informationsquelle in der Lage ist, Selektionen auszuführen, Min die minimale Anzahl von Eingabeparametern und Max die maximale Anzahl von Eingabeparametern. Eingabeparameter müssen den Daten in der Informationsquelle exakt entsprechen, während Selektionen mit Vergleichsoperatoren wie $<$, $\leq$, $\geq$ oder $>$ arbeiten können.

Beispielsweise würde die Quellenbeschreibung einer Informationsquelle, die Daten über Autos enthält, die ab 1992 gebaut wurden und neben der Eingabe eines Modells auch eine optionale Einschränkung nach Herstellungsjahr akzeptiert, folgendermaßen aussehen:

$$q1(model, year, price, owner) : Cars(model, year, price, owner) \wedge Car(model) \wedge year \geq 1992;$$
$$(\{model\}, \{model, year, price, owner\}, \{year\}, 1, 2)$$

model müßte also explizit angegeben werden ($Min = 1$), während die Angabe des Herstellungsjahres nicht zwingend notwendig ist.

Die Anfrageplanung und -ausführung

Im IM wurde eine neue Optimierungstechnik für Anfragen erforscht. Wie bei allen Optimierungen wird versucht, die Kosten der Anfragebearbeitung so gering wie nur irgend möglich zu halten. Im IM wird probiert, dies dadurch zu erreichen, daß bei der Optimierung Anfragen an möglichst viele der für die Beantwortung der Anfrage prinzipiell zur Verfügung stehenden Informationsquellen vermieden werden, so daß nur die Quellen mit wirklich relevanten Informationen angesprochen werden.

Bei der Anfragebearbeitung wird durch *Containment Mapping* (siehe unten) gezeigt, ob bestimmte, zur Beantwortung einer Benutzeranfrage aus den Datenquellenbeschreibungen generierte konjunktive Anfragen in der eigentlichen Benutzeranfrage semantisch enthalten sind (*query containment*). Dies ist nur für konjunktive Anfragen, d.h. Anfragen ohne Negation oder Rekursion, entscheidbar, weshalb im IM auch nur diese Art von Anfragen erlaubt sind.

Query Containment stammt aus der Welt der Optimierung von Anfragen. Dabei werden redundante Teile in einer Anfrage gesucht, indem man zeigt, daß eine Anfrage, auch wenn bestimmte Teile aus ihr entfernt werden, trotzdem semantisch äquivalent bleibt. Äquivalenz ist dabei das wechselseitige Containment. Hier einige Beispiele:

- Jede Anfrage ist natürlich in sich selber enthalten.
- Enthält eine Anfrage im Unterschied zu einer anderen nur zusätzliche Bedingungen an Attributwerte, so wird die Ergebnismenge ja höchstens eingeschränkt. Es kann aber nicht passieren, daß Datensätze geliefert werden, die nicht in der Ergebnismenge der nicht eingeschränkten Anfrage enthalten sind.
- Enthält eine Anfrage im Unterschied zu einer anderen nur zusätzliche Relationen, die mit Joins verbunden sind[1], so kann dies zwar die Anzahl der gelieferten Attribute erhöhen, schränkt aber die Ergebnismenge in bezug auf die gelieferten Datensätze eher ein; es können keine Datensätze geliefert werden, die nicht auch in der Ergebnismenge der ursprünglichen Anfrage enthalten sind.

Containment Mapping ist ein Beweisverfahren für Query Containment: eine Anfrage ist in einer anderen enthalten, wenn es ein Containment Mapping gibt. d.h. wenn es eine Funktion gibt, die durch konsistente Substitution von Konstanten und Variablen alle Relationen der einen Anfrage auf Relationen der anderen abbildet.

Der "Query Evaluation Plan"

Die Auswertung einer Anfrage erfolgt im IM in mehreren Schritten. Im ersten Schritt wird versucht, semantisch korrekte konjunktive Pläne zu erzeugen, die im zweiten Schritt in die optimale Ausführungsreihenfolge gebracht werden. Außerdem werden soviel wie möglich Selektionen an die Quellen übergeben, um die zu übertragende Datenmenge zu verringern.

Bei der Erzeugung der Pläne wird folgendermaßen vorgegangen: Prinzipiell werden alle Kombinationen der Source Descriptions aufgezählt und jeweils versucht, Query Containment zu beweisen. Die Kombinationen, die nicht einmal alle Relationen der Anfrage enthalten oder direkt widersprüchlich sind, können dabei weggelassen werden. Dies erfolgt durch eine erste, relativ einfache Optimierung, bei der für jede Relation der konjunktiven Anfrage ein Töpfchen (bucket) gebildet wird, daß mit für diesen Teil relevanten Informationsquellen gefüllt wird. Danach werden alle Kombinationen aus den verschiedenen Töpfchen auf ihre Brauchbarkeit hin geprüft und nur die relevanten und minimalen Pläne ausgeführt. Bei der Erstellung der Pläne sowie der optimalen Ausführungsreihenfolge werden Einschränkungen durch eventuelle Pflichteingaben (Tupel $\{Eingabe\}$) der Datenquellen beachtet.

Ein weiteres zentrales Forschungsthema im IM war die Verzahnung von Anfrageplanung und -ausführung. Dabei werden ständig die bei der Ausführung der Anfrage gewonnenen Informationen zur weiteren Optimierung verwendet. Es werden also während der Laufzeit weitere Bedingungen zur Optimierung eingearbeitet, im Gegensatz zu herkömmlicher Optimierung, bei der erst der Plan komplett erstellt und dann konsequent abgearbeitet wird.

5.2.3 Einordnung in das Gebiet der Datenintegration

Eine Einordnung des IM in die 3- beziehungsweise 5-Ebenen-Architektur ist nicht sinnvoll, da in diesem Projekt eigentlich nur die Mediationsebene erforscht wurde. Es wurde dabei weder

[1] Mit Join ist hier der Inner Join gemeint.

auf den eigentlichen Zugriff auf die Informationsquellen noch auf die Benutzerschnittstelle näher eingegangen.

Das IM arbeitet mit enger Integration, da der Benutzer die Quellen nicht einmal kennt, sondern nur auf die im World View definierten Relationen und Klassen zugreifen kann. Dabei wird nur lesender Zugriff gestattet. Das verwendete Datenmodell ist die mächtige Beschreibungslogik CLASSIC, die mehr leisten dürfte als darunterliegende WWW-Quellen, weshalb es keine Probleme mit der Übertragung von Informationen in das Datenmodell des Mediators geben dürfte. Die Anfragesprachen in den Wrappern waren nicht Forschungsgebiet, darüber kann also keine Aussage gemacht werden. Im IM werden Daten nur gesammelt und keine Ergebniszusammenführung oder Eliminierung von gleichen Objekten gemacht, was aber auch nicht Ziel der Forschung war. Das IM arbeitet mit virtueller Anfragenbearbeitung und ist ein "top down"-Ansatz.

Wie das IM mit Autonomie und Heterogenität umgeht, ist schwer zu sagen, da auch diese Themen keine Forschungsschwerpunkte waren. Die Autonomie der Komponenten wird kaum eingeschränkt, denn da das IM viele ähnliche Datenquellen in sich vereint, ist die einzelne Quelle zum Funktionieren des Gesamtsystems nicht mehr so entscheidend. Außerdem sind die World Views von IMs wahrscheinlich eher simple gestrickt, da es im IM mehr darum geht, durch eine Oberfläche viele ähnliche Quellen anzufragen, als spezielle Informationen, die nur von wenige Quellen angeboten werden, darzubieten. Änderungen an solch grundlegenden Informationen dürften also eher selten sein. Über die Behandlung der verschiedenen Arten von Heterogenität kann nichts gesagt werden, da diese ja eher durch die Wrapper bearbeiten werden, und Wrapper im IM nicht weiter erforscht wurden.

Verteilung der eingebundenen Informationsquellen wird in großem Maße unterstützt, da das IM in erster Linie WWW-Quellen einbindet und sich mit ihnen über das Internet verbindet.

Das IM ist sehr evolutionsfähig, da es sehr einfach ist, neue Datenquellen hinzuzufügen beziehungsweise alte Datenquellen zu löschen. Einerseits war es ein Ziel, möglichst viele Datenquellen zugänglich zu machen, so daß die einzelne Quelle nicht mehr so stark ins Gewicht fällt: Sollte eine Quelle ausfallen, werden immer noch die Daten von anderen Quellen zu dieser Anfrage zurückgeliefert. Andererseits könnte eine Änderung des World Views schrittweise vollzogen werden, wobei dann Anfragen, die sich auf die neuen Felder beziehen, allerdings nur aus den Quellen mit schon aktualisierten Quellenbeschreibungen kommen würden. Alle anderen Anfragen würden aber problemlos weiterarbeiten.

Im Sinne des IM ist ein *Mediator* ein System, das in der Lage ist zu entscheiden, für welche Benutzeranfrage welche Datenquelle befragt werden muß. Dinge wie das Erkennen von gleichen Objekten oder ähnlichem, werden in diesem Projekt ebenso wenig betrachtet, wie der Zugriff auf die eigentlichen Datenquellen in den *Wrappern*, über die hier somit auch keine Definition angegeben werden kann.

Für weitere Informationen sei auf das Literaturverzeichnis verwiesen, im speziellen auf die Quellen [24], [25], [26] und [27].

5.3 MeBro

MeBro [2] [23] ist ein am Fraunhofer-Institut für Software- und Systemtechnik ISST entwickeltes Framework, das den Aufbau und die Evolution von Informations-Infrastrukturen vereinfachen

[2]MeBro: *Me*tadata-based *Bro*kerage of Information

soll. Dazu werden Methoden, Werkzeuge und Softwarekomponenten bereitgestellt, die für den Entwurf, die Entwicklung, Einführung und kontinuierliche Modernisierung von Informationsvermittlungssystemen notwendig sind.

Wir wollen im folgenden den Aufbau von Systemen betrachten, die mit dem MeBro-Framework entwickelt wurden – sogenannte MeBro-Instanzen. Obwohl eine präzise Trennung zwischen Framework und Instanz notwendig wäre, werden wir zur Vereinfachung die Begriffe MeBro und MeBro-Instanz im folgenden synonym verwenden. Grob läßt sich MeBro wie folgt charakterisieren (vgl. Fig. 5.3):

- Als technische Infrastruktur setzt MeBro auf Internet-Technologien auf, die als Basis für die Informationsvermittlung breite Akzeptanz gefunden haben.
- Inhaltlich setzt MeBro auf eine metadaten-basierte Vermittlung als zentrales Konzept (vgl. Abschnitt 4.3).
- Die organisatorische Einbettung wird durch ein festes Rollenkonzept unterstützt, das die einzelnen Aufgabenbereiche klar definiert. Das MeBro-Modell geht hierbei von einem 'Clearing House'-Ansatz aus: Die Vermittlung von Informationen verschiedener Anbieter erfolgt über eine zentrale Vermittlungsinstanz, die die Metadaten der Anbieter sammelt und an den Nutzern zur Verfügung stellt.

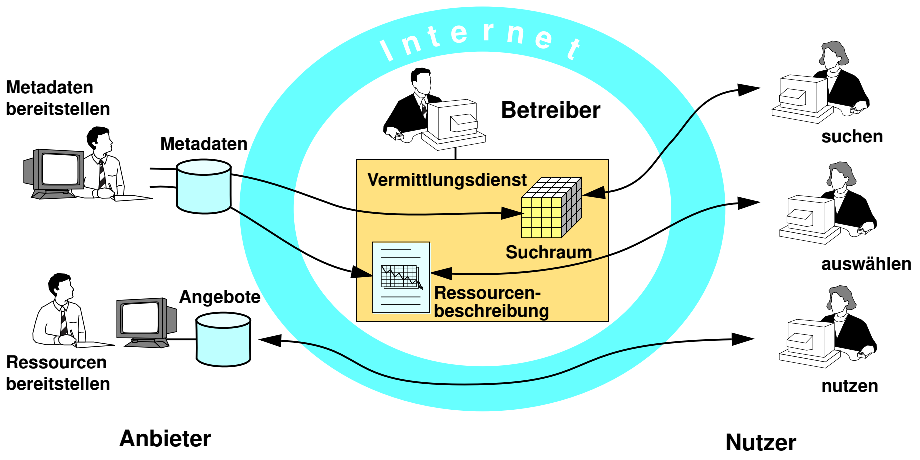

Abbildung 5.3. MeBro – Übersicht

5.3.1 Integration von Metadaten

Die verwendeten Metadaten unterstützen genau die in Abschnitt 4.3.3 kennengelernten Prozesse der Suche nach relevanten Daten, der Entscheidung über ihre Relevanz und Verwendbarkeit sowie der Beschaffung der Daten. Da jedoch lediglich die Suche ein integriertes Schema erfordert, erfolgt in MeBro nur ein partielle Integration der Metadaten einer Komponente.

Die Integration von Metadaten und damit die Bereitsctellung von Information in MeBro untergliedert sich in drei Schritte (vgl. Fig. 5.4):

- *Erfassen von Metadaten*: Jeder Anbieter kann autonom die für seine Datenbestände (Ressourcen) passenden Metadaten definieren. Das Schema dierser speziellen Metadaten entspricht einem lokalen Schema in der 5-Ebenen-Architektur (s. Abschnitt 2.2.3).

- *Syntaktische Homogenisierung*: Die lokalen Metadatenformate werden in ein einheitliches syntaktisches Format überführt. MeBro baut hier auf SGML (in Zukunft XML) auf, d.h. jedes lokale Metadatenschema wird in Form einer SGML-DTD formuliert. Eine solche DTD entspricht dem Komponentenschema in der 5-Ebenen-Architektur.

  Die syntaktisch homogenisierten Metadaten bilden die Ressourcenbeschreibungen, die vom Nutzer für die Entscheidung, ob die beschriebenen Ressourcen tatsächlich relevant sind, verwendet werden können.

- *Semantische Homogenisierung*: Um eine Suche über alle Metadaten durchführen zu können, erfolgt die Abbildung der für die Suche relevanten Ausschnitte (Export-Schema!) der Komponentenschemata in ein globales Schema. Die Unterscheidung des Such- und Entscheidungsprozesses sowie entsprechend die nur partielle Integration der Metadaten ermöglicht es, die Autonomie der Anbieter in der Bereitstellung von Metadaten auf ein notwendiges Maß zu beschränken – zur Schaffung einer komponentenübergreifenden Suche.

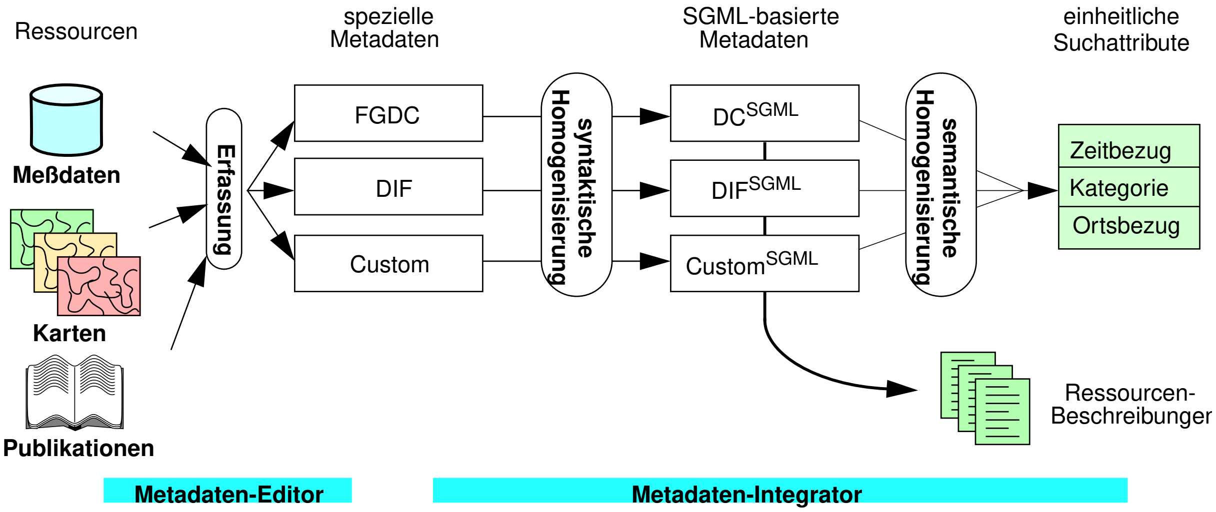

Abbildung 5.4. MeBro-Integrationsansatz ([32])

5.3.2 Das Datenmodell von MeBro

Das globale Schema eines MeBro-Systems baut auf einem multidimensionalen Datenmodell auf, wie wir sie von OLAP-Systemen kennen (vgl. dazu den Abschnitt 6.1): Relevante Kriterien für die Suche bilden die Dimensionen des Suchraums. In dem in Fig. 5.5 dargestellten Beispiel sind dies Schlagworte, Ort- und Zeitbezug von Ressourcen.

In den entstehenden Suchraum werden nun die Verweise auf Ressourcen entsprechend ihrer Metadaten eingeordnet. Eine Ressource kann dabei an mehreren Punkten im Raum auftreten, z.B. wenn mehrere Schlagworte vergeben wurden. Auf der anderen Seite können natürlich auch mehrere Ressourcen auf einem Punkt liegen, z.B. liegen mehrere Ressourcen (ggf. verschiedener Komponenten) vor, die Aussagen über die Temperatur in München 1997 machen.

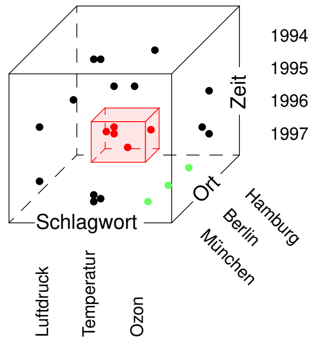

Abbildung 5.5. MeBro – Datenmodell ([32])

Das erstellte globale Schema wird in MeBro in einem relationalen Datenbanksystem abgelegt. Die Abbildung des multidimensionalen in das relationale Datenmodell berücksichtigt dabei insbesondere die Kardinalitäten zwischen Ressource und Dimension, d.h. an wievielen Stellen ein Verweis auf eine Ressource auftauchen darf.

5.3.3 Die Informationssuche in MeBro

Eine Suche erfolgt auf dem oben skizzierten Suchraum. Sie beinhaltet die iterative Einschränkung des Suchraums entlang der Dimensionen. Es ergibt sich letztlich eine konjunktive Anfrage über Dimensionseinschränkungen. Da eine Dimension wiederum sehr komplex strukturiert sein kann und spezifische Benutzerinteraktionen benötigt (man denke an eine geographische Suche), können dimensionsspezifische Suchapplikationen in das System eingebunden werden. Die vorgenommenen Einschränkungen werden über eine definierte Schnittstelle an den globalen Applikationsrahmen übergeben, der letztlich die konjunktive Anfrage an die Datenbank formuliert. Das Ergebnis einer Anfrage ist eine Menge von Verweisen auf Ressourcen und ihren ausführlichen Ressourcenbeschreibungen, die in dem eingeschränkten Suchraum liegen.

Ein besonderes Feature von MeBro bildet die Vorschau-Funktionalität. Sie zeigt bereits vor einer Einschränkung, die damit verbundene Verringerung der Trefferzahl. So ist es möglich, die Anfrageformulierung bestmöglich anzupassen (weder zu restriktiv noch zu allgemein).

5.3.4 Diskussion

Da MeBro genau dem in Abschnitt 4.3 beschriebenen metadaten-basierten Integrationsansatz folgt, treffen genau die dort diskutierten Vor- und Nachteile auch auf MeBro zu: Die Integration auf Metadatenebene verhindert eine Integration mit möglicher Objektfusion, d.h. wir sammeln lediglich Verweise auf Daten und klassifizieren sie nach einem globalen Metadatenschema.

Auf der anderen Seite zeigt der metadaten-basierte Ansatz Vorteile hinsichtlich der Autonomie und Evolutionsfähigkeit. So kann die Integration eines weiteren Anbieters bei MeBro zur Laufzeit ohne Beeinflussung anderer Komponenten erfolgen. Sogar das Hinzufügen von Dimensionen zum Suchraum ist mit vertretbarem Aufwand möglich.

Unterstützt das MeBro-Framework zwar die effiziente Entwicklung metadaten-basierter Informationsvermittlungssysteme, so zeigt es jedoch einige Mängel in der technischen Ausgestaltung. So sind derzeit die Suchapplikationen eng verwoben mit dem Suchraum. Eine zusätzliche Schnittstelle zum Suchraum ist nicht gegeben, so daß eine Integration eines MeBro-Systems als Komponente in ein übergeordnetes Informationssystem kaum möglich ist. Der MeBro-Mediator kann also nicht als eine Komponente eines anderen Mediators eingesetzt werden.

Ein *Mediator* im Kontext von MeBro bezeichnet die Systemkomponente, die dem Nutzer eine metadatengestützte Suche, Auswahl und letztlich Ermittlung relevanter Daten (Ressourcen) aus verschiedenen Quellen ermöglicht und damit einen einheitlichen Zugang zu heterogenen Datenbeständen erlaubt. *Wrapper* können in MeBro zur Kapselung der Ressourcen eingesetzt werden, wenn diese nicht direkt (per URL) referenzierbar sind; sie werden in MeBro jedoch nicht detaillierter betrachtet.

5.4 SIMS

5.4.1 Überblick

SIMS ("Search in Multiple Sources") ist ein System, das die Integration und den Zugriff auf verschiedene Informationsquellen mittels einer Ontologie realisiert ([2, 3]). Anfragen werden mit einer globalen einheitlichen Sprache an das System gestellt und sind von der Art und dem Ort der Informationsquellen und von der Verteilung der gesuchten Information auf diese unabhängig. SIMS entscheidet, welche Informationsquellen genutzt werden, wie Daten temporär gespeichert und manipuliert werden, und auf welche möglichst effiziente Weise diese Daten abgerufen werden. Ein wichtiger Aspekt von SIMS ist die Fähigkeit, dynamisch auf Informationsquellen zugreifen zu können. Steht zum Beispiel eine Quelle nicht zur Verfügung, dann werden alternative Möglichkeiten gesucht, um die Daten abzurufen. SIMS wählt bei der Behandlung einer Anfrage zunächst eine Gruppe von Informationsquellen aus, generiert einen Abfrage-Plan (Query Plan) und optimiert diesen, um die Ausführungskosten minimal zu halten.

Zur Spezifizierung der zugrundeliegenden Ontologie wurde Loom gewählt. Die Elemente, die mit Loom beschrieben werden, sind Konzepte und Rollen, die zum einen die Attribute einer Klasse darstellen, als auch Relationen zu anderen Klassen.

Das SIMS System wurde in mehreren Anwendungsgebieten getestet, unter anderem ein "Trauma Care"-Informationssystem und ein Logistik-Informationssystem, das den zur Zeit größten Diskursbereich beinhaltet, der mit dieser Technologie umgesetzt wurde. Es enthält acht relationale Datenbanken über Flughäfen, Seehäfen und geographische Daten. Diese Datenbanken wurden repliziert und ein kommerzieller Zugang über das Internet realisiert. Es bietet allerdings nur Datenselektion an, keine Manipulation, so daß die Autonomie der Quellen komplett gewahrt bleibt. Die beteiligten Datenbanken sind groß, die GEO-Datenbank enthält zum Beispiel 23 Tabellen mit Graden zwischen 2 und 33, von denen die mächtigste 52.687 Tupel enthält. Das Domain Model dieses Informationssystems enthält ungefähr 170 Konzepte. Das System wurde mit 23 komplexen

Abfragen getestet, die auf einer Sparc 20 alle jeweils innerhalb einer Sekunde bearbeitet wurden.

SIMS enthält in seiner jetzigen Version Wrapper für relationale Datenbanken (z.B. ORACLE), Loom Knowledge Bases und für Programme zur Datengeneration, die innerhalb von SIMS aufgerufen werden können. Wrapper für andere Arten von Quellen sind denkbar, wurden jedoch bisher nicht implementiert. Das System kann zumindest durch die hohe semantische Ausdruckskraft seiner Sprache verschiedenste Datenmodelle repräsentieren. Diese Ausdruckskraft führt dazu, daß die Ontologie ihren Quellen keine Anfragebeschränkungen auferlegt, Loom kann alle Anfragemöglichkeiten ausschöpfen, die die Quellen bieten. Es ist jedoch durchaus möglich, daß Anfragen gestellt werden, die von den Quellen nicht beantwortet werden können.

Die technische Heterogenität ist bei SIMS nicht sehr hoch. Es handelt sich wie erwähnt um replizierte relationale Datenbanken, deren Schnittstellen sehr ähnlich sind. Die logische Heterogenität wird mit Hilfe der Ontologie bewältigt, wobei SIMS nicht in der Lage ist, die Extensionsmengen intensional gleicher Daten aus unterschiedlichen Quellen auf Subsumption zu testen, um qualitative Aussagen über die Quellen machen zu können.

5.4.2 Anfragebearbeitung

Eine Anfrage wird vom Mediator zunächst umformuliert, da in ihr zu Anfang nur Domain Model Konzepte enthalten sind (siehe auch Abschnitt 4.2). Der Mediator wendet dazu verschiedene Operatoren an, die dazu dienen, die passenden Source Model Konzepte zu finden, Generalisierungen / Spezialisierungen und virtuelle Konzepte aufzulösen, Äquivalenzen zu anderen Konzepten festzustellen, und Domain Model Konzepte aufzuspalten, wenn sie aus mehreren Source Models zusammengesetzt sind. Ein Schwachpunkt bei der Zuweisung zu Source Model Konzepten ist, daß verschiedene Informationsquellen, die Daten der gleichen Intension liefern können, in ihrer Extension nicht näher betrachtet werden. Es wird nicht überprüft, ob eine Extensionsmenge die andere enthält, ob sie disjunkt sind oder sich überlappen. Der Mediator würde sich anhand der Mächtigkeit dieser Extensionsmengen für eine entscheiden. Dieser Schwachpunkt ist allerdings keine Fehler des Ontologieansatzes, da der Mediator sehr wohl in der Lage ist, Daten zu vergleichen. Der Mediator bietet auch die Möglichkeit, Daten zu aggregieren und zu abstrahieren.

Jede Abfrage wird mittels dieser Operatoren so lange umformuliert, bis sie nur noch Source Model Konzepte enthält und damit ausführbar ist. Alle diese Operatoren erhalten die Semantik der Abfrage. Wenn diese Operatoren auf eine Abfrage angewandt werden, wird eine Vielzahl von möglichen Abfragen auf Source Ebene erzeugt, der Suchraum kann sehr groß werden. Da einige davon performanter als andere sind, muß der Suchraum vom Mediator betrachtet werden. Sollte der Suchraum vollständig erzeugt worden sein, aber keine Abfrage ausschließlich aus Source Model Konzepten bestehen, dann ist die originäre Abfrage vom Mediator nicht durchführbar, das System bricht ab. Um den Suchraum zu beschränken, werden Operatoren nur auf Domain Model Konzepte angewandt, die nicht direkt auf Source Model Konzepte gemappt werden können.

Zusätzlich zu den Umformungsoperatoren benötigt der Mediator noch Operatoren zur Manipulation der Daten, um sie von einer Informationsquelle zu einer anderen zu verschieben, Joins auszuführen, Daten abzurufen, neue Attribute zu erzeugen oder zu berechnen. Diese Manipulationsoperatoren werden während der Abfragebearbeitung ausgeführt, das heißt, es findet ein reger Datenfluß zwischen den Quellen statt. Trotzdem sollte man hier noch von einer virtuellen Anfragebearbeitung sprechen, da der Mediator die Daten nicht selber hält. Beide Operatorarten sind mittels einer generellen Operatorsprache spezifiziert. Der Planungsprozeß für die Abfrage wird in die Suche integriert. Der Vorteil liegt in der dynamischen Abschätzung der Abfragekosten der erzeugten Teilpläne. Diese Abschätzung wird mit einer branch-and-bound Suche kombiniert, um den besten Plan zu finden. Die Kostenabschätzung basiert auf Statistiken über die Kardinalität der Rollen und die Mächtigkeit der Konzepte.

5.4.3 Bewertung

Die Evolutionsfähigkeit von SIMS ist vergleichsweise gut. Die Ontologie kann auf Domain Model Ebene beliebig erweitert werden, ohne daß Änderungen in den Source Models nötig sind. Umstrukturierungen des Domain Models führen hingegen zu einem erneuten Zuordnen der Source Model Konzepte und Rollen zu denen des Domain Models. Der Aufwand richtet sich hier nach der Art der Umstrukturierung. Die Erweiterung oder Änderung eines Komponentenschemas kann zu einer Remodellierung seines Source Models und dessen Assoziationen zum Domain Model führen, hat aber weder Auswirkungen darauf noch auf andere Source Models. Das Hinzufügen neuer Quellen kann zu Änderungen oder Erweiterungen des Domain Models führen, wenn die Konzepte noch nicht von der Ontologie abgedeckt wurden, ansonsten findet lediglich die Definition des Source Models sowie dessen Assoziationen statt. Das Entfernen von Quellen und deren Source Models hat keine Auswirkungen auf die Ontologie. Die geographische Verteilung der Quellen wird mit Hilfe des Replikationsmechanismuses überbrückt, eine Änderung hätte keine weitergehenden Komplikationen. Eine Veränderung einer Komponente hinsichtlich des DBMS kann durch eine neue Zuteilung eines Wrappers abgefangen werden. Ist ein solcher für das neue System nicht vorhanden, führt dies zu sehr hohem Aufwand, da ein solcher entwickelt werden müßte.

Wie alle ontologie-basierten Systeme (eine Ontologie ähnelt einem föderierten Schema) handelt es sich bei SIMS um eine enge Integration. Da Ontologien den Anspruch haben, den gesamten Diskursbereich abzudecken, wird meist ein Top-Down Ansatz zur Integration verwendet.

Ein *Mediator* kann innerhalb dieses Projekte als der Mechanismus definiert werden, der eine globale Abfrage entgegennimmt, sie auf Source Model Konzepte überführt, einen Query Plan erstellt und durchführt, dabei den Datenfluß zwischen den Komponenten koordiniert, dann die Daten aufbereitet und dem Nutzer letztendlich präsentiert.

Ein *Wrapper* ist hier der Mechanismus, der ein Source Model mit der dazugehörigen Komponente verbindet, und Abfragen des Mediators auf dieses Source Model ausführt.

5.5 TSIMMIS – Mediatoren als Viewspezifikation

Das Projekt TSIMMIS[3] der Universität Stanford und des IBM Almaden Research Centers führte zur bisher umfangreichsten Referenzimplementation der Mediator-Architektur. Das Ziel ist die flexible Integration heterogener Informationsquellen, welche sowohl strukturierte als auch unstrukturierte Daten enthalten.

In einem traditionellen Integrationsszenario gibt es zwei Phasen: eine Integrationsphase, in der die Datenmodelle und Schemata angeglichen werden, und eine Zugriffsphase auf die Daten. In der heterogenen Umgebung von TSIMMIS ist stattdessen eine ständige, fließende Anpassung nötig,

[3]TSIMMIS steht als Abkürzung für *The Stanford-IBM Manager of Multiple Information Sources*. Zusätzlich ist Tsimmis die Bezeichnung für eine jüdische Speise, die aus den heterogenen Zutaten Früchten und Gemüse besteht.

bei der es erst beim Auftreten bestimmter Informationen klar werden kann, daß diese verknüpft werden.

Die Promotion [35] vermittelt ausführlich die zu TSIMMIS veröffentlichten Themengebiete. Ein deutschsprachiger Überblick wird in [53] geboten. Im folgenden werden die wichtigsten Begriffe vorgestellt, Vergleiche mit verwandten Forschungsgebieten durchgeführt und eine Klassifizierung von TSIMMIS vorgenommen.

5.5.1 Die Architektur von TSIMMIS

Um integrierten Zugriff auf heterogene Datenquellen zu ermöglichen, wurden das allgemeine Objektmodell OEM und die Anfragesprachen MSL bzw. LOREL für TSIMMIS entwickelt.

OEM ist ein selbstbeschreibendes, auf Objektidentität und auf hierarchisches Objekt-Nesting beschränktes *Objektmodell.* Diese Beschränkung vereinfacht mit ihrer geringen Komplexität das Ziel der Integration. Die zurückgegebenen OEM-Objekte sind entsprechend einem hierarchischen Datenmodell verschachtelt – oft enthalten sie baumartige Strukturen. OEM wird im Kapitel 4.4 näher beschrieben. Sowohl Wrapper als auch Mediatoren liefern ihre Ergebnisse in OEM zurück. Daher können Mediatoren nicht nur auf Wrapper, sondern auch auf andere Mediatoren zugreifen.

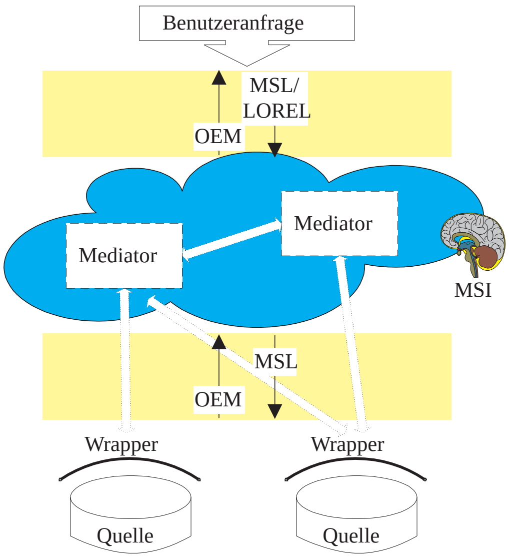

Abbildung 5.6. Die Mediator-Architektur von TSIMMIS

Die *Anfragesprachen MSL und LOREL* beschreiben die zu findenden OEM-Objekte. Das in Kapitel 4.4 näher beschriebene LOREL orientiert sich am SQL-Syntax und wird als Endbenutzersprache propagiert — es ist als rein *deskriptive* Anfragesprache auch nur dazu fähig. Im Unterschied ist dazu enthält die "Mediator Specification Language" MSL *zusätzliche prozedurale Elemente* zur Mediatorspezifikation und ist in erster Linie nicht als Anfragesprache konzipiert, obwohl sie dafür verwendet werden kann. MSL benutzt eine Prolog-ähnliche Syntax (genauer:

DATALOG-ähnlich) und wird zur Spezifikation von Mediatoren und ihrer Integrationskonzepte benutzt.

In Fig. 5.6 ist der Ablauf einer Anfragebearbeitung in TSIMMIS dargestellt:

- Der Benutzer stellt eine Anfrage in MSL oder LOREL an den Mediator Specification Interpreter MSI.
- Der MSI verarbeitet die Anfrage. Seine Komponenten werden noch vorgestellt.
- Mediatoren sind in der Graphik gestrichelt dargestellt, da sie keine eigenständig laufenden Prozesse sind, sondern View-Spezifikationen innerhalb des MSI.
- Die Einzelanfragen werden in MSL an die Wrapper der betroffenen Quellen gestellt. Das Ergebnis wird an den Anfragenden in OEM zurückgegeben.

Die Begriffe "Mediator" und "Wrapper" im Kontext von TSIMMIS

Ein *Mediator* ist ein View[4] mit Integrationskonzepten, also mit Regeln zur Fusion von gefundenen Objekten. Er wird durch eine Menge deklarativer Regeln in der Sprache MSL spezifiziert. Mediatoren arbeiten unabhängig voneinander. Ein Mediator muß die von ihm verarbeitete Information nicht vollständig verstehen - er kann sie auch unverarbeitet weiterreichen.

Die Daten von einer Quelle werden von einem *Wrapper* in das allgemeine Objektmodell OEM konvertiert, nachdem die MSL-Anfragen vom Wrapper für die Quelle übersetzt wurden. Ein Wrapper ist wie schon ein Mediator kein eigenständiger Prozeß, sondern eine Spezifikation der Fähigkeiten bzw. Anfragebeschränkungen der Quelle.

Anfragebearbeitung durch Views im MSI

Der Mediator Specification Interpreter MSI analysiert die Anfrage, sammelt die angeforderten Informationen und integriert sie entsprechend den Mediatorspezifikationen, um die entdeckten OEM-Objekte zurückzuliefern. Folgende Komponenten[5] des MSI verarbeiten die Anfrage:

- Der *View Expander And Algebraic Optimizer* erstellt einen logischen Plan. Dafür versucht er, die Anfrage mit den zur Verfügung stehenden Mediatorspezifikationen, also Views, zu matchen und entscheidet daraufhin, welche Quellen befragt werden.
- Der logische Plan wird daraufhin an den *Cost Based Optimizer* übergeben. Dieser erstellt daraus einen physischen Plan, der festschreibt, in welcher Reihenfolge die Quellen befragt werden.
- Der physische Plan wird nun von der *Datamerge Engine* ausgeführt. Die hier nötige Objektfusion zur korrekten Zuordnung von Informationen zu gleichen, globalen Objekten, wird detailiert in [36] beschrieben.

[4] Der Begriff "View" wird in den TSIMMIS Papers nicht verwendet. Stattdessen sprechen diese gleichbedeutend von "virtuellen Objekten".

[5] In den TSIMMIS Papers werden sie teilweise unterschiedlich benannt.

Capability Based Rewriting der Mediatoren

Das komplexes Forschungsgebiet des Capability Based Rewriting beschäftigt sich mit der Frage, ob eine von den Quellen "naiv" einzeln nicht-beantwortbare Anfrage durch eine *Kombination von Einzelanfragen* beantwortet werden kann. Dies geschieht sowohl im Mediator als auch im Wrapper. Im Mediator wird ein Anfrage basierend auf den Beschreibungen der Wrapperfähigkeiten in der *Relational (!) Query Description Language* RQDL zerlegt in eine Kombination von Einzelanfragen an den Wrapper, und im Wrapper wird sie basierend auf den Beschreibungen der Fähigkeiten der Quelle in QDTL (Query Description and Translation Language) in eine Kombination von Einzelanfragen an die Quelle zerlegt. Die in QDTL beschriebenen Fähigkeiten und Anfragebeschänkungen einer Quelle werden Query Templates genannt. Von den hierzu zahlreich veröffentlichen Papieren bietet sich u.a. [37] an.

Falls beispielsweise eine Quelle "naiv" nur die Eltern einer Person nennen kann, können durch eine Kombination von Einzelanfragen die Eltern der Eltern, also die Großeltern ermittelt werden. Die Antwortfähigkeit nimmt weiter zu, wenn mehrere Quellen kombiniert werden. Falls beispielsweise obige Eltern-Quelle "naiv" nicht beantworten kann, ob eine Person denselben Beruf ausübt wie ihre Großeltern, könnte dies durch automatisches Kombinieren von Einzelanfragen beantwortet werden.

Hier soll nur auf drei nicht-triviale Problemfelder dieser Forschungsrichtung hingewiesen werden:

- Die Beschreibung der Wrapperfähigkeiten einer Quelle erfolgt in einer relationalen Logik, die eng an Datalog angelehnt ist. Dies steht im Widerspruch zu der eigentlich vollständig auf das nicht-relationale OEM ausgerichteten Architektur. Dieser Widerspruch wird auch nicht hinreichend aufgeklärt. Insbesondere ist es eine offene Frage, ob man alle OEM Views immer in eine relationale Anfrage gegen ein ebenfalls zu erstellendes Schema übersetzen kann.

- Die Frage, ob eine MSL-Anfrage einem QDTL-Template oder einer RQDL-Spezifikation entspricht, ist ein NP-vollständiges Problem; es verwendet ähnliche Techniken wie die Anfragebeantwortung des Information Manifold Projektes (siehe Abschnitt 5.2)

- TSIMMIS-Mediatoren können nur Quellen durch Capability Based Rewriting in ihren Antwortfähigkeiten erweitern, die in RQDL beschrieben sind. RQDLs wurden nur für Wrapper untersucht. Was passiert, wenn ein Mediator zuerst auf andere Mediatoren zugreift? Wie entsteht die RQDL für den Mediator, auf den zugegriffen wird?

5.5.2 Vergleich von TSIMMIS mit anderen Architekturen

In diesem Kapitel wird TSIMMIS gegenüber anderen Forschungsgebieten eingeordnet und verglichen.

TSIMMIS im Vergleich zu klassischen FDBMS

Der Hauptunterschied zu einem eng integrierten FDBMS ist das fehlende globale Datenschema. Das "Schema" eines TSIMMIS-Diskursbereiches sind stattdessen OEM-Labels, also einfach alle Bezeichner aus den Schemata der Quellen. Der Benutzer hat nur geringes Vorwissen über die Struktur der Daten. Seine Anfragen gleichen eher der Stichwortsuche in einer Internet-Suchmaschine, wobei die Stichwörter sowohl in Quellschemata (Labels in OEM) als auch in deren Extensionen (Values in OEM) gesucht werden können.

TSIMMIS im Vergleich zu Metasuchmaschinen

Die Mediator-Architektur weist einige Gemeinsamkeiten zu Internet-Metasuchmaschinen auf. Bei beiden soll eine einheitliche Benutzerschnittstelle mit Orts-, Modell- und Schnittstellentransparenz für eine Vielzahl von Quellen angeboten werden. Die von den beiden Forschungsrichtungen verwendeten Architekturen gleichen sich:

- Zuerst erfolgt die Quellenselektion,
- danach die Anfrageübersetzung mit dem Bestimmen der Queryfragmente für die ausgesuchten Datenquellen und dem Übersetzen der Fragmente für die jeweilige Datenquelle.
- Schließlich wird Result Merging durchgeführt.

Im Vergleich ist die View-Komplexität bei Mediatoren weitaus höher als bei Metasuchmaschinen. Auch in der Benutzererwartung unterschieden sich beide Herangehensweisen. Von Mediatoren werden vollständige Antworten erwartet und geliefert, von Metasuchmaschinen nur die durch Ranking bestimmten besten Treffer. Die Mediator-Architektur ähnelt in ihren Fragestellungen also eher den Anfragesprachen für das Web als den in [13] zum Vergleich bemühten Metasuchmaschinen.

TSIMMIS im Vergleich zu anderen Mediator–Architekturen

Die TSIMMIS-Wrapper sind derart komplex, daß sie sich beinahe nur noch durch ihre Kopplung an Quellen von Mediatoren unterscheiden. Hier wurde auf schlanke Wrapper zugunsten von einer höheren Antwortmächtigkeit verzichtet. Die Anfragebeschränkungen einer Quelle werden in sogenannte Query Templates beschrieben. Sowohl die Wrapper als auch die Mediatoren benutzen diese Query Templates wie in 5.5.1 erläutert, um die naiven Anfragebeschränkungen der Quellen automatisch zu erweitern. Das parallel zu TSIMMIS entwickelte, in 5.1 skizzierte Projekt Garlic geht den umgekehrten Ansatz der schlanken Wrapper.

Mediatoren und Wrapper sind in TSIMMIS keine eigenständigen Prozesse, sondern View-Definitionen. Zur Integration der Daten wird in MSL das Erkennen gleicher Objekte, das Erkennen redundanter Daten und die Auflösung von Datenkonflikten unterstützt. Referenzen auf in der Antwort enthaltene Objekte werden korrekt an den Anfrager weitergegen. Auch Referenzen auf in der Quelle vorhandene, aber nicht in der Antwort enthaltene Objekte (z.B. WWW-Links) sollen möglich sein.

5.5.3 Verteilung, Autonomie und Heterogenität der Komponenten in TSIMMIS

TSIMMIS soll nun anhand der in diesem Seminar benutzen Nomenklatur klassifiziert werden.

Die Architektur ist eine direkte Umsetzung der 3-Ebenen Architektur mit Wrappern und Mediatoren wie in Kapitel 2 beschrieben. Es wird nur lesender Zugriff erlaubt.

Berücksichtigung von Verteiltheit in TSIMMIS

Verteiltheit spielt hier keine Rolle. Die Mediatoren selbst sind nicht verteilt, da sie ja nur Views innerhalb eines MSI-Prozesses sind. Die dazu nötige Erweiterung wäre eine Verteilung von MSI-Prozessen auf mehrere Knoten, deren Mediatorspezifikationen miteinander verknüpft sind.

Auf die Frage, ob durch die Wrapper verteilte Quellen angesprochen werden können, wird nicht eingegangen.

Berücksichtigung von Autonomie in TSIMMIS

Die Autonomie von Quellen wird dahingehend eingeschraenkt, daß sie zur Anfragezeit verfügbar sein müssen.

Eine unbemerkte Veränderung eines Quellschemas wird nicht zu einem Laufzeitfehler führen (wie das bei relationalen Anfragen der Fall ist), sondern einfach dazu, daß die MSL View-Spezifikation nicht mehr von der Quelle gematched werden kann und deshalb vorhandene Information nicht mehr gefunden wird. MSL ist flexibel und kann, wie in den nächsten beiden Abschnitten geschildert, so formuliert werden, daß Quelländerungen gehandbabt werden können - allerdings nur auf Kosten der Integrationstiefe.

MSL's Beschreibungsfähigkeit von Heterogenität

Unterschiedliche Modellierungskonzepte in verschiedenen Quellen werden durch die Wrapper angeglichen. Sie übersetzen Daten in das einheitliche Objektmodell OEM (*Datenmodellheterogenität*) und MSL-Anfragen an die Quellen in die quellspezifischen Anfrageformate (*syntaktische Heterogenität*). Auch *plattformbezogene Heterogenität* wird durch die jeweilige Spezifikation von Wrappern modellierbar. *Strukturelle Heterogeneität* kann man dabei sehr gut behandeln, da OEM (und MSL) sehr flexibel mit der Struktur von Daten umgehen, weil sie auf ein relationales Schema verzichten.

Schematische Heterogenität ist in MSL insofern beschreibbar, daß ein und dieselbe MSL-Variable sowohl für Labels als auch für Values stehen kann, also sowohl für Schema-Bezeichner als auch für extensionale Daten.

Durch unkorrekte oder veraltete Daten auftretende *Datenkonflikte* müssen beim Betrachten der Antwort erkannt werden und können dann zur Um- oder Neuformulierung von Mediatorspezifikationen führen.

Zur automatischen Überbrückung von *semantischer Heterogenität* wird in den TSIMMIS-Papers ein Lexikon-Service vorgeschlagen, der zu jeder Quelle über deren Inhalte Buch führt und zu einem Thesaurus erweitert werden kann. Dieser Vorschlag wurde aber nicht implementiert. Die zur Objektfusion wichtige Beschreibbarkeit der Gleichheit von gefundenen OEM-Objekten wird durch zahlreiche Möglichkeiten der Vergabe von *semantischen Objekt-ID's* Rechnung getragen.

Analyse der Evolutionsfähigkeit von TSIMMIS

Schema-Evolution wird in den *"Rest-Variablen"* der MSL Spezifikation aufgefangen — dem Betrachter werden diese Rest-Variablen in der Antwort weitergereicht und ihm ist es dann überlassen, neue oder fehlende Information zu bemerken. Durch die Verwendung von *Variablen für OEM-Labels* in Mediatorspezifikationen ist es auch möglich, Bedingungen an unbekannte Schema-Information zu knüpfen. So kann beispielsweise an ein aus einer Quelle gefundenes Datum "Hausmeister" die Bedingung geknüpft sein, daß "Hausmeister" der Name eines Attributs im Schema einer anderen Quelle ist, welches das Datum "Horst Maier" enthalten muß. Sobald der das Datum "Hausmeister" durch "Gebäudepfleger" ersetzt wird, gilt die "Horst-Maier" Bedingung im Vergleichsschema für das Attribut "Gebäudepfleger".

Zusätzlich ermöglicht MSL *unvollständige Pfadangaben* mit Wildcards. Eine nicht ungefährlicher aber mächtiger Unterschied von MSL zu LOREL sind die in MSL möglichen *rekursiven Views*.

Das globale OEM-Modell bleibt aufgrund seiner Trivialität konstant, Modelländerungen sind nicht zu erwarten.

Zusätzliche Quellen oder der Wegfall von Quellen führen nur zu mehr oder weniger ausführlichen Antworten als vorher, behindern aber das System nicht. Bei Änderung der Schnittstelle einer Quelle müssen die Query Templates in QDTL für die Quelle und in RQDL für den Wrapper angepaßt werden. Inwieweit sich QDTL in RQDL oder umgekehrt umformen läßt, ist nicht erforscht.

Mediatoren können auf andere Mediatoren zugreifen. Allerdings werden die dabei entstehenden Komplikation im Rahmen von TSIMMIS nicht untersucht. So könnten in einer vielschichtigen ("n-tier") Architektur beispielsweise Schleifen oder unbehandelte Überlappungen auftreten. Die sowieso schon kaum durchschaubare Komplexität eines TSIMMIS-Systems bietet schon genügend Fehlerquellen, von der Kombination mehrerer Mediatoren ist beim derzeitigen Stand der Forschung abzuraten.

Inwieweit integrationsbedürftige Anwender sich auf die Diskrepanz zwischen der schwer vermittelbaren Anfragelogik und dem simplen Antwortformat OEM einlassen ist abzuwarten.

5.6 Mariposa

Das Mariposa Projekt wurde mit dem Ziel gegründet, "eine Plattform zur Entwicklung eines neuartigen Algorithmus für die verteilte Anfrageoptimierung bereitzustellen" [43, 44]. Ausgangspunkt war also die Idee eines neuartigen Optimierungsalgorithmus, gar die eines neuen Optimierungsparadigmas - das eines mikroökonomischen Systems verteilter Server, die auf einem gemeinsamen Markt, dem Netzwerk, um die Ressource Geld konkurrieren.

Das Herzstück von Mariposa ist der mikroökonomische Optimierer. Das ganze Mariposa System wird als virtueller (oder wahrer) Marktplatz mit den Sites als Anbieter und den Anfragen als Abnehmer beschrieben. Dadurch wird nicht nur eine Anfrageoptimierung und -bearbeitung geleistet, als Nebeneffekt wird außerdem das Speichermanagement übernommen, das entscheidet, welche Daten oder Replikate wo gespeichert werden. Das Marktplatz-Paradigma hat den Vorteil, daß keine zentrale Koordinationsstelle des DBMS alle Vorgänge verwalten muß, und daß eine dynamische Anpassung der Anfragebearbeitung und Speicherverwaltung möglich wird. Zum Betrieb des Systems wird ein Regelsystem verwendet.

Die Kernidee des Marktplatzes ist die Zuteilung eines Budgets in einer fiktiven Währung für jede Anfrage. Die Anfragen verwenden dieses Budget, um sich bearbeiten zu lassen, während die Sites Geldeinheiten verlangen, um die Anfrage oder einzelne Operatoren zu bearbeiten oder die entsprechenden Daten zu liefern. Da die einzelnen Sites in Konkurenz zueinander stehen und gleichzeitig ihre Einnahmen maximieren wollen, wird das gesamte System ohne weiteres Zutun nach dem Prinzip der freien Marktwirtschaft optimiert.

Mariposa ist also eine verteilte Datenbank und entspricht in den meisten Eigenschaften denen in Abschnitt 6.2. Technisch dürfen die einzelnen Komponenten *heterogen* sein, sie müssen lediglich ein sogenanntes kanonisches Schema im- und exportieren können. Eines der wichtigsten Ziele des Mariposa Systems war es, eine starke *Verteilung* der Daten zu unterstützen. Nicht nur Klassen, sondern auch einzelnen Objekte können nach Belieben verteilt und repliziert werden. Einzig das marktwirtschaftliche Prinzip gebietet Einhalt, etwa dann, wenn sich eine weitere Verteilung finanziell nicht mehr lohnt.

Das Regelsystem bietet eine starke *Evolutionsfähigkeit* der Konfiguration und Technik. Eine Komponente kann z.B. leicht ausscheiden, indem sie alle Daten verkauft und keine weiteren Dienste mehr anbietet. Das Schema hingegen ist nur sehr schwer zu ändern, da jede beteiligte Komponente informiert werden muß. Es wird nur vage die Idee einer Schema ID angesprochen, jedoch nicht weiter verfolgt.

Das Regelsystem erlaubt außerdem einen flexiblen Umgang mit der *Autonomie*. Je nach Definition der Regeln einer Komponente ist die Komponente sehr autonom, und nimmt zum Beispiel keine fremde Regel entgegen, reagiert nicht auf Anfragen etc., oder aber weniger autonom, so daß z.B. fremde Komponenten die Regeln verändern können.

Auch wenn dies nicht explizit erwähnt wird, gibt es in einem Mariposa System ein globales Schema. Mariposa verlangt von einer teilnehmenden Komponente die Beherrschung einer globalen kanonischen Datenrepräsentation der Daten des gesamten Schemas. Das *Datenmodell* basiert auf POSTGRES und ist somit objektorientiert. Im Unterschied zu anderen Systemen sind Mariposa-Daten mobil. D.h. weder haben sie eine feste Heimat, noch gibt es eine Komponente, die etwa für ein bestimmtes Objekt "verantwortlich" ist.

Kapitel 6

Verwandte Systeme und Begriffe

Dieses Kapitel ist verwandten Ansätzen zur Datenintegration gewidmet, die wir aber nicht unter die "Mediator-basierten" Systeme zählen aus folgenden Gründen:

- Klassische *Data Warehouses* materialisieren per Definition alle Daten und betreiben damit keine virtuelle Integration. Selbstverständlich treten dadurch wieder spezifische Probleme auf, z.B. die Zusicherung der Aktualität der Daten. Außerdem befaßt sich die Forschung im Bereich Warehouses größtenteils mit thematisch entfernt liegenden Fragen, z.B. der Optimierung multi-dimensionaler Anfragen.

- *Verteilte Datenbanken* verbieten die meisten Formen von Heterogenität in ihren Komponentensystemen. Ihre Autonomie wird damit quasi aufgehoben. Durch den hohen Grad an Konformität sind damit aber hoch-performante Systeme möglich.

6.1 Data Warehouses

Ein Data Warehouse ist ein Informationssystem zur Entscheidungsunterstützung für das Management im Unternehmen. Es zeichnet sich darin aus, daß es ein von den operativen Systemen im Unternehmen physisch getrenntes System ist, auf das nur lesend zugegriffen wird. Typischerweise enthält ein Data Warehouse Daten, die über einen längeren Zeitbezug verfügen und zusätzlich in verdichteter Form vorliegen. Diese Daten stammen häufig aus einer Vielzahl unterschiedlicher Datenbanksysteme, da in den meisten Unternehmen eine historisch gewachsene heterogene Systemwelt vorherrscht. Es existieren mittlerweile unzählige Definitionen zum Thema Data Warehousing (vgl.[14]). Eine weit verbreitete und akzeptierte Definition ist die von Inmon (1992) (aus [14]):
Ein Data Warehouse ist eine Ansammlung von subjektorientierten, integrierten, zeitvarianten und nicht volatilen Daten, die der Unterstützung des Managements in Entscheidungsprozessen dienen.
Unter der Subjektorientierung von Daten wird eine an die Unternehmensprozesse angepaßte Datenhaltung verstanden. Datenintegration bedeutet das Beheben von Dateninkonsistenzen, welche beim Zusammenfügen von Daten aus unterschiedlichen Systemen auftreten können, sowie deren Aggregation. Daten müssen zeitvariant gespeichert werden, d.h. es erfolgt eine Datenarchivierung. Nicht-Volatilität beschreibt den nur lesenden Zugriff auf ein Data Warehouse.

6.1.1 Schemakonzept

Ein Data Warehouse besteht im wesentlichen aus einer Datenbasis, welche sowohl aktuelle als auch historische Daten und Informationen aus allen Unternehmensbereichen enthält. Diese Daten liegen in atomarer Form und in unterschiedlichen Verdichtungsstufen vor (s. Fig. 6.1 und vgl. [33]). Die Integration unternehmensinterner und -externer Daten in ein Data Warehouse setzt umfangreiche Transformationsprozesse voraus. Es müssen Funktionen zur Datenextraktion aus den unterschiedlichen operativen Systemen vorhanden sein. Dabei treten häufig Probleme bei der Zusammenfügung der Daten aus Systemen unterschiedlicher Schemata auf. Das Konzept eines Cleansing Houses als eine Vorstufe des Data Warehouse bietet die Möglichkeit, Homonyme und Synonyme zu erkennen und durch entsprechende Transformationsprozesse semantische Integrität zu schaffen.

Updates eines Data Warehouses erfolgen in vom Unternehmen abhängigen Zeitintervallen. Jedoch sollten diese Intervalle nicht untertägig sein, damit eine unnötige Belastung der operativen Systeme vermieden wird. In der Regel erfolgen die Updates als sog. Batch-Läufe, also zu Zeitpunkten niedriger Auslastung der Systeme (z.B. in den Nachtstunden).

Das Ziel eines Data Warehouses ist es, flexible und individuelle Ad-hoc Anfragen für unterschiedliche Zielgruppen zu ermöglichen. Die Schaffung eines eigenständigen Systems wird damit begründet, daß operationale DV-Systeme auf eine möglichst effiziente Verarbeitung des relativ statischen Tagesgeschäftes ausgerichtet sind. Dieses zeichnet sich darin aus, daß eine hohe Anzahl von Transaktionen auf relativ kleinen Datensätzen erfolgt. Dagegen sind dispositive DV-Systeme durch eine effiziente Verarbeitung einer hohen Anzahl sequentiell gelesener Datensätze sowie einer flexiblen Anpassung an die dynamischen Informationsbedürfnisse der Entscheidungsträger geprägt.

6.1.2 Drei-Ebenen-Architektur

Ein Data Warehouse ist eingebettet in eine Drei-Ebenen-Architektur (s. Fig. 6.2). Auf der Applikationen-Ebene erfolgt der Zugriff des Endanwenders auf das System. Dazu stehen unterschiedliche Produkte als sog. Front-Ends zur Verfügung. Dieses können einfache Statistik-Programme, aber auch komplexe Data Mining-Tools sein. Das eigentliche Data Warehouse, welches auf der mittleren Ebene angesiedelt ist, wird durch OLAP-Systeme getragen. Die operativen Systeme bilden die dritte Ebene und werden als OLTP-Systeme bezeichnet.

Online Transaction Processing Systeme (OLTP-Systeme) zeichnen sich darin aus, daß sie ausschließlich detaillierte und aktuelle Daten vorhalten. Sie sind voll in die unternehmensinternen Geschäftsprozesse integriert und verfügen daher über einen ständig aktuellen Datenbestand. Die Systemlast ist aufgrund des fest definierten und somit statischen Geschäftsprozesses vorhersehbar. Die Datenverarbeitung erfolgt transaktionsgesteuert. An das System werden hohe Anforderungen bezüglich der Verfügbarkeit (Ausfallsicherheit) gestellt. Pro abzuarbeitenden Prozeß werden nur auf geringe Datenmengen zugegriffen.

Online Analytical Processing Systeme (OLAP-Systeme) zeichnen sich hingegen darin aus, neben atomaren, aktuellen Daten, auch verdichtete und historische Daten vorzuhalten. Die Aktualität der Daten hängt von der Häufigkeit der Updates des Systems ab. Die Systemlast ist nicht vorhersehbar und kann stark schwanken, da die Benutzer (Management) nicht permanent Anfragen stellen. Die Verfügbarkeit eines OLAP-Systems ist nicht so entscheidend, da ein Systemausfall

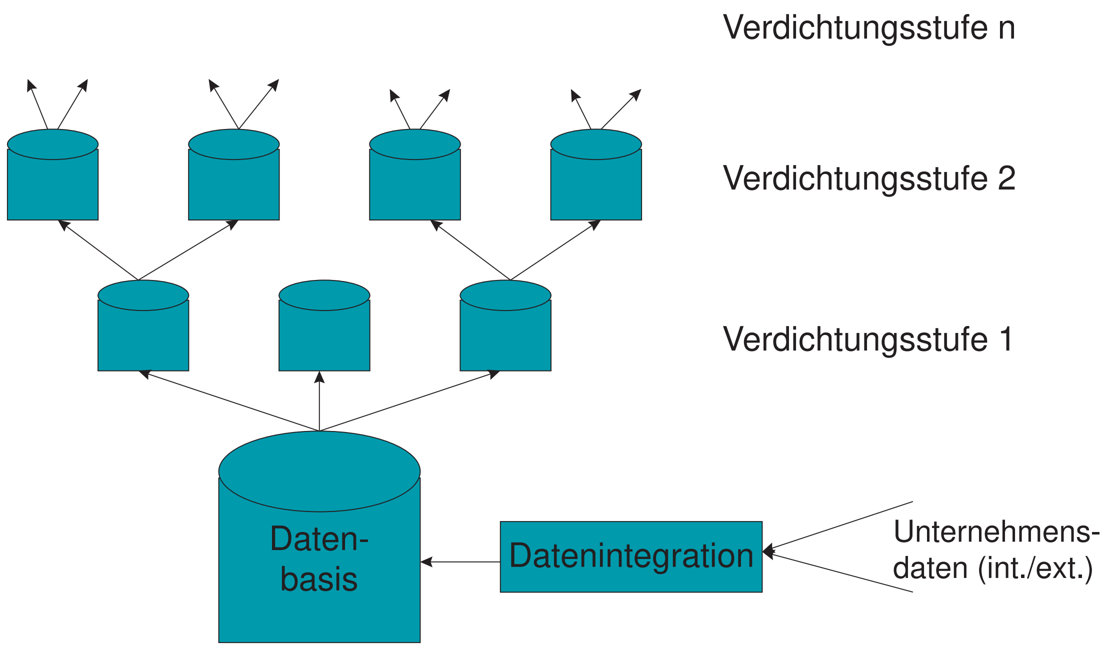

Abbildung 6.1. Architekturkonzept eines Data Warehouses

weit geringere finanzielle Folgen hätte als bei OLTP-Systemen. OLAP-Systeme müssen dagegen sehr flexibel bezüglich der Anfragen sein. In der Regel werden sehr komplexe Anfragen gestellt, die auf eine relativ große, multidimensionale Datenmenge zurückgreifen.

6.1.3 Multidimensionales Datenmodell

Die Konzeption managementunterstützender Informationssysteme erfordert eine besondere Art der Modellierung. Während bei der Modellierung operationaler Informationssysteme Datenobjekte des Tagesgeschäftes im Vordergrund stehen, müssen in managementunterstützenden Systemen entscheidungs- und führungsrelevante Fragestellungen Schwerpunkte der Modellierung sein. Primäres Ziel bei der Modellierung ist die Schaffung analysefähiger und relevanter Datenstrukturen. Daher enthält das Datenmodell insbesondere verdichtete und zeitbezogene Daten. Häufig werden die Daten in einer Zeithierarchie verdichtet. Diese kann ausgehend vom Kalenderjahr als höchste Verdichtungsstufe, über Halbjahre, Quartale, Monate bis hin zu den Kalendertagen als niedrigste Verdichtungsstufe aufgespannt werden.

Multidimensionale Datenmodelle verwenden häufig eine Matrixdarstellung, die die Daten als

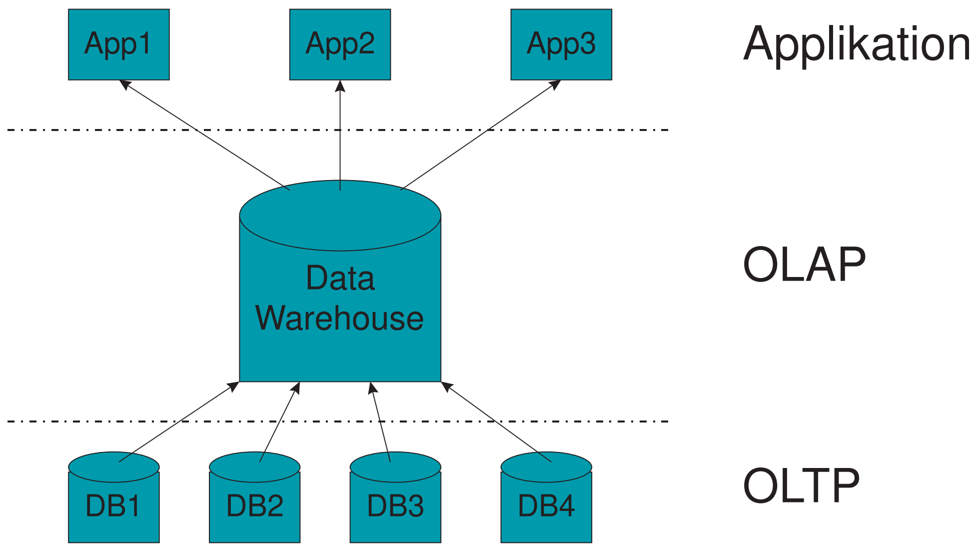

Abbildung 6.2. Drei-Ebenen-Architektur

Werte enthält. Dimensionen bezeichnen die Charakteristika und Bezugsgrößen, welche die Semantik der Werte in der Matrix bestimmen. Die einzelnen Einflußgrößen eines Wertes in der Matrix sind jedoch zunächst noch abstrakte Gebilde mit einer Menge bestimmter Ausprägungen. Dabei wird jede relevante Einflußgröße als eine Dimension dargestellt und die einzelnen Ausprägungen als sogenannte Dimensionspositionen entlang der Dimension abgetragen. Zudem existieren zwischen den einzelnen Dimensionspositionen oft hierarchische Beziehungen.

Multidimensionale Datenstrukturen können entweder in speziellen multidimensionalen Datenbanksystemen, sogenannte MOLAP-Systeme, oder in angepaßten relationalen Datenbanksystemen, sogenannte ROLAP-Systeme, abgebildet werden. Während MOLAP-Systeme speziell auf die Eigenschaften multidimensionaler Datenstrukturen zugeschnitten sind, muß bei der Umsetzung in relationalen Datenbanksystemen (ROLAP) auf spezielle Schemata zurückgegriffen werden. Die am häufigsten verwendeten Schemata sind das Star Schema und das Snowflake Schema (vgl. [18]).

6.1.4 Fazit

Charakteristisch für Data Warehouses ist es, daß sie Daten aus den heterogenen, verteilten Informationssystemen vollständig übernehmen und dann den Bedürfnissen der Benutzer entsprechend aufbereiten. Das Data Warehouse schafft somit im Gegensatz zu anderen Systemen eine materialisierte Integration.

Aufgrund des speziellen Anwendungsgebietes von Data Warehouses, nämlich dessen managementunterstützende Funktion, basiert das zugrundeliegende Schema nicht auf einer vollständigen und korrekten Inklusion der einzelnen Komponentenschemata, sondern auf dessen Abstraktion. Hierfür werden spezielle Datenmodelle, wie z.B. das multidimensionale Modell benötigt.

Die angebundenen Komponentensysteme behalten prinzipiell ihre Autonomie, jedoch verursachen Entwurfsänderungen in den Komponentensystemen Anpassungen im Data Warehouse Schema. Neben der syntaktischen Heterogenität muß vorrangig die logische Heterogenität der Komponentensysteme überwunden werden, wobei Integrationsprobleme ausschließlich in Update-Prozessen auftreten, da Anfragen direkt auf die materialisierten Daten zugreifen. Letztendlich treten jedoch die gleichen Probleme auf wie sie auch in anderen Informationssystemverbünden anzutreffen sind.

Das Data Warehouse System hat sich also in der Phase der Datenakquisition zu bewähren. Dazu müssen geeignete Tools zur Verfügung stehen, die Datenintegrität schaffen (Cleansing House). Die Anforderungen an diese Akquisitionssysteme steigen dabei um so mehr, je heterogener die Systemlandschaft ist. Daher sollte im Anfangsstadium ein Data Warehouse zunächst in kleineren Teilbereichen etabliert werden (Data Marts) und dann schrittweise auf das ganze Unternehmen ausgedehnt werden.

Das Data Warehouse Schema muß evolutionsfähig im Hinblick auf seine logische Erweiterbarkeit sein, um neue Komponentensysteme integrieren zu können. Zudem führen neue Informationsbedürfnisse der Endanwender, insbesondere dem Management, dazu, daß das globale Schema leicht änderbar sein muß. Dieses Kriterium der Evolutionsfähigkeit ist bei der Erstellung eines multidimensionalen Modells immanent wichtig. Ein Data Warehouse sollte schon auf Grund der hohen zu tätigenden Investitionen evolutionsfähig gehalten werden, um eine möglichst lange Lebensdauer zu gewährleisten.

6.2 Verteilte Datenbanken

Verteilte Datenbanken werden als eine Sammlung mehrerer logisch verknüpfter Datenbanken definiert, die über ein Netzwerk verteilt sind [34]. Sie unterscheiden sich von mediator-basierten Systemen im wesentlichen durch die mangelnde Autonomie und der daraus resultierenden Homogenität. Ein verteiltes DBMS wird nicht als ein System aufgefaßt welches einzelne Komponenten integriert. Vielmehr ist es ein in sich geschlossenes System. Die technische und logische *Integration* findet, falls nötig, vorab statt.

Im Gegensatz zu föderierten Datenbanken oder MBIS ist die *Autonomie* der beteiligten Komponenten stark eingeschränkt, oder in vielen Fällen nicht vorhanden. Typischerweise fungiert ein ausgezeichnetes Datenbankmanagementsystem als zentrale Verwaltungsstelle aller beteiligten Datenbanken. Die angebundenen lokalen Datenbanksysteme geben in der Regel die gesamte Kontrolle an das zentrale DBMS ab. So gehen die Entwurfs-, Kommunikations- und weitestgehend auch die Ausführungsautonomie verloren. Den lokalen DBMS bleibt zumeist nur die Autonomie in der Wahl des Ausführungsplans der an sie gestellten Anfrage.

Verteilte Datenbanken enthalten keine logische *Heterogenität*. Die einzelnen Datenbanken verwenden das gleiche Datenmodell, gleiche Schemata und gleiche Struktur. Lediglich die verwendete Hardware kann sich bei den einzelnen Komponenten unterscheiden. Durch die starke Homogenität fällt die Integration der Daten besonders leicht, eine Mediation im herkömmlichen Sinne ist nicht nötig.

Bei der *Verteilungsstrategie* einer Datenbank stehen prinzipiell zwei Möglichkeiten zur Verfügung:

1. Die Partitionierung der Daten, die entweder aufgrund von Organisationsstrukturen vorgegeben ist (dies ist der Regelfall für MBIS), oder aktiv vorgenommen wird.

2. Die Replikation, teilweise oder vollständig, vertikal oder horizontal, die immer aktiv vorgenommen wird.

Die Möglichkeit zur Replikation und der Konsistenzerhaltung replizierter Daten ist ein entscheidender Vorteil verteilter Datenbanken.

Die *Evolutionsfähigkeit* verteilter Datenbanken entspricht der zentralisierter DBMS, mit der besonderen Schwierigkeit, daß logische Änderungen an alle Komponenten propagiert werden müssen. Aufgrund mangelnder Autonomie der lokalen Datenbanken ist dies allerdings eher ein technisches als ein logisches oder strukturelles Problem. Konfigurationsänderungen, etwa die Hinzunahme einer neuen Komponente, sind problemlos – nur das zentrale DBMS muß angepaßt werden, andere lokale Komponenten bleiben unbeeinflußt. Eine systemtechnische Änderung jedoch ist kaum möglich, die Funktionalität aller Komponenten ist identisch. Eine Erweiterung muß somit gleichzeitig bei allen Komponenten stattfinden. Bei vorhandener Kompatibilität kann ein verteiltes DBMS durchaus mit einem anderen System verknüpft werden. Jedoch darf die Funktionalität beider Systeme nicht beschnitten werden.

Die Nachteile eines verteilten DBMS, also die Beschränkungen im Datenmodell, die Inflexibilität usw., werden aufgewogen durch die volle Funktionalität eines Datenbanksystems (Transaktionen, Recovery etc.). Es werden alle ACID Eigenschaften gewährleistet. Ein Nachteil gegenüber zentralisierten DBMS ist die zusätzliche Komplexität und die verlängerten Antwortzeiten durch die Verteilung. Die verwendete *Anfragesprache* wird lediglich durch das verwendete Datenbanksystem beschränkt. In der Regel sind also komplexe Anfragen möglich. Entsprechend bestehen weitere Schwierigkeiten beim Umgang mit Daten, wie sie in MBIS vorkommen, nicht.

Im Gegensatz zu den meisten MBIS ist Anfragebearbeitung auf Replikaten explizit erwünscht. Durch das zugrundeliegende DBMS bleibt die Konsistenz erhalten. Eine verteilte Datenbank hat also die freie Wahl zwischen *virtueller Anfragebearbeitung* auf verteilten Komponenten und der Bearbeitung auf *materialisierten* Replikaten.

Verteilte Datenbanken werden sowohl *Bottom-Up* als auch *Top-Down* entworfen. Bei der Bottom-Up Strategie müssen in der Regel große Änderungen an den einzelnen bereits vorhandenen Komponenten vorgenommen werden. Die Top-Down Strategie entsteht für verteilte Datenbanken nicht nur durch Anforderungen. Vielmehr kann es aus organisatorischen oder aus Kostengründen nötig sein, eine bereits vorhandenes zentralisiertes DBMS nachträglich zu verteilen.

Da es in verteilten Datenbanken nur ein gemeinsames Schema gibt, entspricht die *Architektur* eines solchen Systems der klassischen ANSI/SPARC Architektur.

Kapitel 7

Was ist ein Mediator-basiertes Informationssystem ?

Ausgehend von den separat betrachteten Konzepten und konkreten Systemen wollen wir nun versuchen, zu einer Definition Mediator-basierter Informationssysteme (MBIS) zu kommen und die Frage zu klären, ob hinter diesem neuen Begriff tatsächlich neuartige Forschungsthemen stecken oder es sich lediglich um ein „Modewort“ handelt. Die folgenden Tabellen enthalten dazu einen zusammenfassenden Vergleich der in Kapitel 5 betrachteten Systeme.

	Garlic	**Information Manifold**	**SIMS**
Quellautonomie ?	Hoch (medium wrapper)	Hoch (thin wrapper)	Hoch (thin wrapper)
Quellverteilung ?	Ja	Ja	Replizierte Quellen
Heterogenität ?	Gering	Insb. strukturell	Insb. semantisch
Evolutionsfähigkeit ?	Gut bzgl. Anfragen, schlecht bzgl. Schemata	Sehr gut	Sehr gut
Eng / lose ?	Eng	Eng	Eng
Read-only ?	Ja	Ja	Ja
Mediatormodell ?	ODMG-93	Relational, mit DL Erweiterung	DL LOOM
Anfragebehandlung ?	Kostenbasierte Anfrageoptimierung	Query Containment	Anfrage-subsumption
Abstraktion ?	Nein	Nein	Ja
Virtuell ?	Ja	Ja	Ja
Bottom-Up / Top-Down ?	Eher Bottom-Up	Top-Down	Top-Down

Tabelle 7.1. Einordnung der verschiedene Projekte im Überblick - Teil 1. “n.a.” = nicht anwendbar.

	MeBro	**Tsimmis**	**Mariposa**
Quellautonomie ?	Hoch	Hoch (thick wrapper)	Flexibel (Regelsystem)
Quellverteilung ?	Ja	Egal	Sehr hoch
Heterogenität ?	Hoch	Hoch	Gering
Evolutionsfähigkeit ?	Hoch	Hoch	Konfiguration
Eng / lose ?	Eng auf Metadatenebene	Kein schema = lose	Eng
Read-only ?	Ja	Ja	DBMS Funktionalität
Mediatormodell ?	Multidim./RDM	OEM	Postgress (OO)
Anfragebehandlung ?	Multidim. Indizes	OEM View Expansion	Marktplatz
Abstraktion ?	Ja	Möglich	n.a.
Virtuell ?	Nein bzgl. Metadaten	Ja	Beides
Bottom-Up / Top-Down ?	Beides möglich	Top-Down	n.a.

Tabelle 7.2. Einordnung der verschiedene Projekte im Überblick - Teil 2.

Allen Projekten gemeinsam ist das grundlegende Ziel, einen einheitlichen *lesenden* Zugriff auf eine Menge existierender Datenquellen zu bieten. Durch die ausschließliche Betrachtung lesender Operationen ergeben sich im Gegensatz zu bekannten Ansätzen föderierter Datenbanksysteme Möglichkeiten, sich stärker von den Datenquellen zu lösen. Dies zeigt sich z.B. darin, daß auch Verdichtungen und Abstraktionen (bis hin zur Verwendung von Metadaten anstelle der Daten) vorgenommen werden können.

In der Regel finden wir eine *3-Ebenen-Architektur mit Wrappern und Mediatoren* vor, wie sie in Kapitel 2 beschrieben wurde. Dem Mediator wird dabei die Rolle der logischen oder semantischen Integration zugewiesen. Dementsprechend finden wir meist eine *enge Kopplung* vor, bei der das föderierte Schema zur Lösung dieser Integrationsprobleme verwendet wird (Ausnahme bildet hier TSIMMIS mit seiner losen Kopplung). Allerdings ist hier für jedes System genauer zu untersuchen, welche Art von Integration erfolgt. Ob Daten verschiedener Quellen lediglich gesammelt werden oder ob Datenobjekte „gematched", d.h. fusioniert werden und wenn ja mit welchem Mechanismus ist von zentraler Bedeutung für das Gesamtsystem, erstens für die Nachvollziehbarkeit eines Anfrageergebnisses und zweitens, da die nicht betrachteten Integrationsprobleme immer vom Nutzer gelöst werden müssen.

Besondere Berücksichtigung finden in allen Projekten die Fragestellungen der *Autonomie der Quellen* sowie letztlich die *Evolutionsfähigkeit* des Systems hinsichtlich der Konfiguration (neue Quellen dazu). Dies liegt darin begründet, daß mit einem MBIS ein größeres System mit vielen Quellen adressiert wird. Eine weitgehende Entwurfsautonomie wird dadurch erreicht, daß das föderierte Schema eher *top-down*, unabhängig von konkreten Quellen, entwickelt wird. Die Quellen können dann einzeln mit dem föderierten Schema in Beziehung gesetzt werden, ohne das föderierte Schema zu beeinflussen. Diese Vorgehensweise benötigt wiederum die Einschränkung, lediglich lesende Zugriffe zu erlauben, da so kein genaues Abbild der Quellen auf der globalen Ebene notwendig ist.

Verteilung wird in keinem Projekt explizit behandelt, sondern vielmehr als Gegebenheit angesehen, die mittels existierenden Kommunikationstechniken überwunden werden kann. Ebenso als gegeben angenommen wird die *Heterogeneität* der Datenquellen, da die verschiedensten existierenden Systeme als Komponenten integriert werden sollen. In allen Projekten findet sich die grundlegende Teilung in Wrapper und Mediator, wobei Wrapper mindestens Schnittstellen- und Datenmodellheterogenität addressieren und Mediatoren die logische Heterogenität. Darüber hinaus können Wrapper jedoch auch weitere Funktionalität übernehmen: sie werden dann zu „thick“ Wrappern (vgl. TSIMMIS). Bezüglich der integrierbaren Systeme sollen in fast allen Projekten sowohl strukturierte als auch semi-strukturierte Quellen berücksichtigt werden. Dies ist sicherlich ein Trend, den MBIS von föderierten DBS unterscheidet, wo immer von DBS als Komponenten ausgegangen wird. Inwieweit jedoch eine solche Offenheit gegeben ist, ist in den meisten Projekten nicht bewiesen und eher als weitergehendes Ziel zu verstehen.

Welche Konflikte bzgl. *logischer Heterogenität* von einem System gelöst werden ist unterschiedlich. Wie oben bereits beschrieben, werden unterschiedliche Arten der logischen Integration verfolgt, die sich auf entsprechende Heterogenitätskonflikte beziehen. Zu beobachten ist der *Trend zu höheren Datenmodellen*, wie objektorientierte Modelle oder Description Logics, wohingegen in föderierten Datenbanksystemen häufig auf dem RDM oder einfachen Objektmodellen aufgesetzt wird. Anscheinend sind die Heterogenitätskonflikte in MBIS aufgrund des Erhalts der Entwurfsautonomie der Quellen häufiger oder komplizierter und auf der Basis „herkömmlicher“ Datenmodelle nicht mehr zu lösen.

Weiterhin ist festzustellen, daß die meisten MBIS eine *virtuelle Integrationsstrategie* verfolgen. Materialisierungen treten nur bei Metadaten (vgl. MeBro) oder zur Optimierung auf. Dies läßt sich ebenfalls damit erklären, daß die Autonomie von Quellen und Flexiblität des Systems hinsichtlich der Konfiguration (Anzahl integrierter Datenquellen) besonders berücksichtigt werden. Bei einer regen Fluktuation der integrierten Komponenten kann nur ein virtueller Integrationsansatz mit dem dynamischen Rückgriff auf die Quellen die Forderung nach Datenaktualität erfüllen. Auf der anderen Seite bekommen dadurch Probleme der Anfragebearbeitung und -optimierung sowie Fragestellungen der Verfügbarkeit von Quellen eine besondere Bedeutung, um eine akzeptable Antwortzeit zu gewährleisten.

Abschließend folgern wir für uns:

> Ein Mediator-basiertes Informationssystem (MBIS) ermöglicht einen einheitlichen lesenden Zugriff auf eine Vielzahl weitgehend autonomer Informationssysteme. Es ist aufgrund der Autonomie der Quellen per Definition verteilt und heterogen. Die Architektur beinhaltet Wrapper und Mediatoren. Wrapper dienen der Kapselung von Datenquellen, insbesondere der Überbrückung von Datenmodellheterogenitäten. Mediatoren erreichen die enge Kopplung inklusive der logischen Integration der Datenbestände.

Die wesentlichen Unterschiede zu der Forschungsrichtung föderierter Datenbanksysteme ist darin zu sehen, daß – unter Einschränkung auf lediglich lesende Zugriffe – eine höherwertige Integration verschiedenster Datenquellen angestrebt wird. Besonderes Augenmerk liegt dabei auf der Autonomie der Quellen bzgl. der Teilnahme an der Föderation sowie auf der Berücksichtigung verschiedenster Komponentenarten - von strukturierten DBS über semi-strukturierten bis hin zu unstrukturierten Datenbeständen.

Inwieweit die Vision eines so flexiblen MBIS mit weitreichender Informationsintegration zwecks Kooperation, wie sie in [19] beschrieben wird, jedoch erreicht werden kann, bleibt abzuwarten. Die Forschung steckt hier sicherlich noch in den „Kinderschuhen“. Die Anforderungen in komplexen Anwendungsgebieten, wie etwa der Molekularbiologie oder der Umweltdaetn, zeigen jedoch den Bedarf an neuen Konzepten, Methoden und Systemen in diesem Bereich.

Literaturverzeichnis

[1] S. Abiteboul, D. Quass, J. McHugh, J. Widom, and J. Wiener. The lorel query language for semistructured data. *International Journal on Digital Libraries*, 1(1):68–88, 1997.

[2] Y. Arens, C.-N. Hsu, and C. A. Knoblock. Query processing in the sims information mediator. In A. Tate, editor, *Advanced Planning Technology*, pages 61–69. AAAI Press, Menlo Park, California, 1996.

[3] Y. Arens, C. A. Knoblock, and W.-M. Shen. Query reformulation for dynamic information integration. *Journal of Intelligent Information Systems - Special Issue on Intelligent Information Integration*, 6(2/3):99–130, 1996.

[4] P. Atzeni, G. Mecca, and P. Merialdo. Semistructured and structured data in the web: going back and forth. In *ACM SIGMOD Workshop on Management of Semistructured Data*, Tuscon, Arizona, 1997.

[5] C. Batini, M. Lenzerini, and S. B. Navathe. A comparative analysis of methodologies for database schema integration. *ACM Computing Surveys*, 18(4):323–364, 1986.

[6] A. Borgida. Description logic in data management. *IEEE Transactions on Knowledge and Data Engineering*, 7(5):671–682, 1995.

[7] S. Busse, R.-D. Kutsche, C. Schöning, and et al. Metainformation im bereich umwelt- und geoinformationssysteme – eine modellierungsstudie. Forschungsberichte des Fachbereichs Informatik Nr. 97-25, Technische Universität Berlin, November 1997.

[8] S. Conrad. *Foederierte Datenbanksysteme: Konzepte der Datenintegration.* Springer Verlag, Berlin, Heidelberg, New York, 1997.

[9] P. Dadam. *Verteilte Datenbanken und Client/Server-Systeme – Grundlagen, Konzepte und Realisierungsformen.* Springer, 1996.

[10] The dublin core homepage. http://purl.oclc.org/metadata/dublin-core/.

[11] O. M. Duschka and M. R. Genesereth. Query planning in infomaster. In *12th ACM Symposium on Applied Computing*, San Jose, CA, 1997.

[12] C. H. Goh, M. E. Madnick, and M. D. Siegel. Context interchange: Overcoming the challenges of large-scale interoperable database systems in dynamic environments. In N. Adam, B. Bhargava, and Y. Yesha, editors, *3rd International Conference on Information and Knowledge Management*, pages 337–346, Gaithersburg, Maryland, 1994. ACM Press.

[13] L. Gravano and Y. Papakonstantinou. Mediating and metasearching on the internet. *Data Engineering Bulletin*, 21(2), 1998. http://www.db.ucsd.edu/publications/deb98.ps.

[14] P. Gray and H. Watson. *Decision Support in the Data Warehouse.* Prentice Hall PTR, 1998.

[15] K. Greve. Objekte der metadatenverarbeitung im umweltbereich. In *Datenqualität und Metainformation in Geo-Informationssystemen*, Tagungsband zum Workshop, 7./8. Oktober 1996, Interner Bericht Heft Nr. 5, pages 25–33. Institut für Geodäsie und Geoinformatik, FB Landeskultur und Umweltschutz, Universität Rostock, Okt. 1996.

[16] T. R. Gruber. A translation approach to portable ontology specifications. *Knowledge Aquisition*, 5(2):199–220, 1993.

[17] Health level 7 standards page. http://www.mcis.duke.edu/standards/HL7/hl7.htm; last visited: 10. März 1999.

[18] J. Holthuis. Multidimensionale datenstrukturen - modellierung, strukturkomponenten, implementierungsaspekte. pages 143–193, 1998.

[19] Reference Architecture for the Intelligent Integration of Information, 1995. Version 2.0 (Draft).

[20] O. Kapitskaja, A. Tomasic, and P. Valduriez. Dealing with discrepancies in wrapper functionality. Technical Report 3138, INRIA: Institute National de Recherche en Informatique et en Automatique, March 1997 1997.

[21] V. Kashyap and A. Sheth. Semantic and schematic similarities between database objects: a context-based approach. *The VLDB Journal*, 5(4):276–304, 1996.

[22] L. V. S. Lakshmanan, F. Sadri, and I. N. Subramanian. Schemasql: A language for interoperability in relational multidatabase systems. In T. M. Vijayaraman, A. P. Buchmann, C. Mohan, and N. L. Sarda, editors, *22nd Conference on Very Large Databases*, Bombay, India, 1996. Morgan Kaufman Publishers.

[23] T. Langer. Mebro - a framework for metadata-based information brokerage. In *Proc. 1st Int. Workshop on Practical Information Mediation and Brokering, and the Commerce of Information on the Internet (I'MEDIAT'98)*, Sept. 1998.

[24] A. Y. Levy, A. O. Mendelzon, Y. Sagiv, and D. Srivastava. Answering queries using views. In *14th ACM Symposium on Principles of Database Systems*, pages 95–104, San Jose, CA, 1995.

[25] A. Y. Levy, A. Rajaraman, and J. J. Ordille. Query-answering algorithms for information agents. In *13th AAAI National Conf. on Artificial Intelligence*, Portland, Oregon, 1996.

[26] A. Y. Levy, A. Rajaraman, and J. J. Ordille. Querying heterogeneous information sources using source descriptions. In *22th Conference on Very Large Databases*, pages 251–262, Bombay, India, 1996.

[27] A. Y. Levy, D. Srivastava, and T. Kirk. Data model and query evaluation in global information systems. *Journal of Intelligent Information Systems - Special Issue on Networked Information Discovery and Retrieval*, 5(2):121–143, 1995.

[28] W. Litwin, L. Mark, and N. Roussolpoulos. Interoperability of multiple autonomous databases. *ACM Computing Survey*, 22(3):267–293, 1990.

[29] L. Liu and C. Pu. Dynamic query processing in diom. *IEEE Quarterly Bulletin on Data Engineering; Special Issue on Improving Query Responsiveness*, 20(3), 1997.

[30] J. McHugh, S. Abiteboul, R. Goldman, D. Quass, and J. Widom. Lore: A database management system for semistructured data. *SIGMOD Record*, 26(3):54–66, 1997.

[31] Meyers lexikon in drei bänden. Bibliographisches Institut & F.A. Brockhaus AG, 1995.

[32] I. Mohr. Informationsvermittlung im internet. Vortrag, INFO'98, 9.11. 1998.

[33] H. Mucksch, J. Holthuis, and M. Reiser. Das data warehouse-konzept - ein überblick. *Wirtschaftsinformatik*, 38(4):421–432, 1996.

[34] T. Özsu and P. Valduriez. *Principles of Distributed Database Systems*. Prentice Hall International Editions, 1991.

[35] Y. Papakonstantinou. *Query Processing in Heterogeneous information systems (condensed version.* available at: http://www.db.ucsd.edu/publications/save.ps, 1997.

[36] Y. Papakonstantinou, S. Abiteboul, and H. Garcia-Molina. Object fusion in mediator systems. In T. M. Vijayaraman, A. P. Buchmann, C. Mohan, and N. L. Sarda, editors, *22th Conference on Very Large Data Bases*, Bombay, India, 1996. Morgan Kaufmann.

[37] Y. Papakonstantinou, A. Gupta, and L. Haas. Capabilities-based query rewriting in mediator systems. *PDIS*, page 42, 1996. http://www.db.ucsd.edu/publications/cbr.ps, Kluwer Academic Publishers, Boston.

[38] E. Rahm. *Mehrrechner-Datenbanksysteme: Grundlagen der verteilten und parallelen Datenverarbeitung.* Addison-Wesley, 1994.

[39] A. Rajaraman, Y. Sagiv, and J. D. Ullman. Answering queries using templates with binding patterns. In *14th ACM Symposium on Principles of Database Systems*, pages 105–112, San Jose, CA, 1995. ACM Press.

[40] I. Schmitt. *Schemaintegration fuer den Entwurf foederierter Datenbanken.* DISBIS. Infix Verlag, Sankt Augustin, 1998.

[41] A. Sheth and J. A. Larson. Federated database systems for managing distributed, heterogeneous and autonomous databases. *ACM Computing Survey*, 22(3):183–236, 1990.

[42] S. Spaccapietra, C. Parent, and Y. Dupont. Model independent assertions for integration of heterogeneous schemas. *The VLDB Journal*, 1(1):81–126, 1992.

[43] M. Stonebraker, P. M. Aoki, R. Devine, W. Litwin, and M. Olson. Mariposa: A new architecture for distributed data. In *Tenth Int. Conf. on Data Engineering*, pages 54–65, Houston, Texas, 1994. IEEE, IEEE Computer Society Press.

[44] M. Stonebraker, P. M. Aoki, A. Sah, J. Sidell, C. Staelin, and A. Yu. Data replication in mariposa. In *Proc. of the 12th Int. Conf. on Data Engineering (ICDE)*, Orlando, Feb. 1996.

[45] D. Suciu. Semistructured data and xml. In *Proc. International Conference on Foundations of Data Organsiation*, 1998.

[46] A. Tomasic, L. Raschid, and P. Valduriez. Scaling heterogeneous databases and the design of disco. In *16th Int. Conference on Distributed Computing Systems*, pages 449–457, Hong Kong, 1996. IEEE Computer Society.

[47] M. Tork Roth and P. M. Schwarz. Don't scrap it, wrap it! a wrapper architecture for legacy data sources. In *23rd Conference on Very Large Database Systems*, Athens, Greece, 1997.

[48] P. R. S. Visser, D. M. Jones, T. J. M. Bench-Capon, and M. Shave. An analysis of ontological mismatches; heterogeneity versus interoperability. In *AAAI 1997 Spring Symposium on Ontological Engineering*, Stanford, USA, 1997.

[49] J. Widom. Research problems in data warehousing. In *4th International Conference on Information and Knowledge Management*, pages 25–30, Baltimore, Maryland, 1995.

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

[51] G. Wiederhold. Interoperation, mediation and ontologies. In *Int. Symposium on 5th Generation Computer Systems; Workshop on Heterogeneous Cooperative Knowledge-Bases*, volume W3, pages 33–48, Tokyo, Japan, 1994.

[52] C. Yu and W. Meng. *Principles of database query processing for advanced applications : Multimedia, object-oriented and distributed/heterogeneous databases.* Morgan Kaufmann, 1998.

[53] A. Zwick. Mediatoren – seminar data warehousing im sommersemester 1996. Seminar-ausarbeitung, Universität Karlsruhe, Juli 1996. http://www.swbv.uni-konstanz.de/depot/dokersch/5600000/5739000/5739676k.html.