Grundlagen der Informationsintegration

Seminararbeit

Im Rahmen der Seminars “Mastering the Information Explosion – Information Integration and Information Quality”

Vorgelegt am Lehrstuhl für
Datenbanken und Informationssysteme
Technische Universität Kaiserslautern

Paul R. Schilling

Betreuer: Dipl. Inf. Christian Mathis

Kaiserslautern, den 30. Juni 2006

Inhaltverzeichnis

Abkürzungsverzeichnis

BaV:	Both as View
BGLaV:	BYU Global Local as View
CSV:	Comma Separated Values
DB:	Datenbank
DBS:	Datenbanksystem
DBVS:	Datenbankverwaltungssystem
DI:	Datenintegration (*data integration*)
EAI:	Enterprise application Integration
ETL:	Extract-Transform-Load
GaV:	Global as View
GLaV:	Global Local as View
II:	Informationsintegration (*information integration*)
IIMS:	Informationsintegrationsverwaltungssystem (*information integration management system*)
IT:	Informationstechnik (*information technology*)
LaV:	Local as View
M&A:	Fusionen und Übernahmen (*mergers and acquisitions*)
PPL:	Peer Programming Language
SQL:	Standard Query Language
XML:	Extensible Markup Language

1. Motivation

Daten sind allgegenwärtig. Wären wir in einer vollkommenen Welt, so könnte man davon ausgehen, dass die Daten in einem Unternehmen ordentlich organisiert in einer einzigen, zusammenhängenden Informationsinfrastruktur abgelegt seien. Informationen wären den richtigen Personen, Prozessen und Anwendungen im Unternehmen zum rechten Zeitpunkt leicht zugänglich. Jedes Unternehmen wäre dazu im Stande, Informationen auf Anfrage zu liefern.

Leider sieht es in den Unternehmen meist ganz anders aus. Zahlreiche Unternehmen sind im Laufe der Zeit gewachsen oder haben sich verändert. Reorganisationen, Fusionen und Übernahmen (*mergers and acquisitions – M&A*), oder aber auch taktische „quick fix" Projekte haben die Kohärenz der Unternehmensdaten stark beeinträchtigt. Ständig werden den Unternehmen durch Hinzufügen von Abteilungen oder Unternehmenseinheiten (*business units*), welche ihre eigenen Datenbanken und Anwendungen benutzen, neue Daten und Informationsquellen (*content*) hinzugefügt. Diese neuen Informationsquellen sind sehr oft von den restlichen Informationen im Unternehmen abgeschirmt und für Anwendungen von außerhalb nicht zugänglich. Daten, welche von mehreren Personen oder Anwendungen genutzt werden können, sind oft von fragwürdiger Nützlichkeit: Unternehmen versagen immer öfter, die Qualität und Konsistenz von herumgereichten Daten zu garantieren.

Da Unternehmen sich immer internationaler orientieren und versuchen ihre Partner, Lieferanten und Kunden mit in das Unternehmen einzubinden wollen, werden Zugänglichkeit und Integration von Informationen von immer größerer Bedeutung.

Eine Umfrage hat ergeben, dass etwa 75% aller Finanzvorstände sich über ungenügende Zugänglichkeit von Kundeninformationen im Unternehmen beklagen [7]. Eine weitere kürzlich durchgeführte Befragung von weltweit 1800 Unternehmensleitern hat ergeben, dass das ihrer Meinung nach größte Einsparpotential eines Unternehmens in der Informationsintegration liegt [7]. Investoren können laut Gesetzgeber immer umfangreichere Informationen von den Unternehmen verlangen, und immer öfter sind es betriebliche Informationssysteme, welche den Anforderungen nicht gerecht werden.

In dieser Seminararbeit, wird in erster Linie die Problemstellung der Datenintegration näher erläutert. Hierzu klären wir zunächst einige Grundbegriffe, welche für die Datenintegration von übergeordneter Bedeutung sind. Anschließend soll dann die replikationsorientierte Integration der virtuellen Integration gegenübergestellt werden und beobachtet werden, welche Probleme bei der Integration auftreten können. In einem weiteren Punkt werden dann die verschiedenen Formen der Heterogenität näher erläutert und diverse Integrationsmethoden und Prozesse vorgestellt. Abschließend betrachten wir uns dann Umsetzung und Nutzen von Datenintegration anhand eines konkreten Beispiels.

2. Die Informationsintegration als Teil der Unternehmensintegration

Im Laufe der letzten Jahrzehnte haben Unternehmen und Organisationen fortschreitend Informationsverarbeitungsplattformen entwickelt, um die Daten und Informationen im Unternehmen handhaben zu können. Dies führte allerdings über die Jahre zu einer Ansammlung von unabhängigen Anwendungen und Informationssystemen. Diese verschiedenen isolierten Datenverarbeitungseinheiten nun zu einem Ganzen zusammenzuführen, stellt die IT Abteilungen in den diversen Unternehmen oft vor eine große Herausforderung. Darüber hinaus wird häufig auch noch Druck von Seiten des Managements ausgeübt, da die Integration und Interoperabilität von verschiedenen unabhängigen Systemen ein enormes Einsparpotential darstellt. Durch eine einheitliche Darstellung der Daten können die Arbeitsproduktivität im Unternehmen gesteigert, Unternehmensprozesse rationalisiert und der Austausch von Daten und Zusammenarbeit verschiedener Abteilungen in einem Unternehmen gewährleistet werden.

Immer öfter ist auch die Rede von Unternehmensintegration (*business integration*) [4]. Hier geht es vorrangig um die Frage, wie vorhandene, oft autonome Systeme, Komponenten oder Datenquellen weiter genutzt werden, und wie Probleme der Datenheterogenität oder der Verteilung der verschiedenen Systeme gelöst werden können. Techniken und Herangehensweisen werden nach [9] wie folgt in vier Kategorien unterteilt:

- *Portale* dienen dazu, dem Benutzer einen einzigen Zugangspunkt zu den verschiedenen Datenquellen zu bieten. Diese Portale sollen die Möglichkeit bieten, an bestimmte Anforderungen des Benutzers angepasst werden zu können. Gängigerweise wird eine Benutzerschnittstelle genutzt, in welcher sich der Benutzer einloggen kann, und welche dann abhängig von den Benutzerrechten sich bei den verschiedenen autonomen Systemen einloggt.
- Die *Integration von Geschäftsprozessen* soll dazu dienen, einzelne Geschäftsprozesse verschiedener Abteilungen oder Unternehmenseinheiten zu einem Geschäftsprozess zusammenzufügen, welcher auf das gesamte Unternehmen, oder gar über die Unternehmensgrenzen, angewandt werden kann. Hierzu werden Ablaufpläne und Geschäftsprozessmodelle erstellt. Da es vorrangig um den anwendungsseitigen Zugriff und den einheitlichen Austausch von Daten geht, gewinnen XML und Web Dienste in diesem Gebiet immer stärker an Bedeutung.
- Bei der *Integration von Anwendungen* geht es vorrangig darum, verteilte Anwendungen miteinander zu verbinden, welche entweder komplementär agieren, oder aber auch Anwendungen, welche ähnliche Aufgaben auf den gleichen Daten durchführen. Hierzu werden hauptsächlich Middleware-Techniken eingesetzt wobei *Application-Connector*- und *Adapter*-Konzepte es erlauben, die verschiedenen Anwendungen miteinander zu verbinden. Die Hauptaufgaben hierbei sind Transformation und Austausch von Daten, wodurch XML auch in diesem Bereich immer wichtiger wird.
- Bei der Informationsintegration (auch als Datenintegration bekannt) geht es nun darum, die verschiedenen Typen von Daten aus den verschiedenen Quel-

len logisch und physisch zusammenzubringen, so dass sie uniform zugänglich sind, und dass Anfragen auf ihnen durchgeführt werden können. Zusammengefasst handelt es sich bei der Informationsintegration also darum, eine holistische Sicht auf die Daten eines Unternehmens zu erstellen, oder gar die Daten unternehmensübergreifend einheitlich darzustellen.

Bei einer Unternehmensintegration ist meist eine Kombination der vier oben beschriebenen Techniken erforderlich. Die verschiedenen Techniken werden komplementär eingesetzt. Die Informationsintegration bildet dabei die Basis, da eine einheitliche Verfügbarkeit der Daten eine conditio sine qua non für die Unternehmensintegration ist.

Sowohl in der Forschung als auch in der Industrie wird der Informationsintegration große Aufmerksamkeit geschenkt und es wurden zahlreiche Bemühungen gemacht, Techniken und Lösungen zu erstellen. Die Referenzen [6,12] geben eine Übersicht der verschiedenen Lösungen.

3. Theorie der Datenintegration - Definitionen

Die Theorie der Datenintegration ist ein Teil der Datenbanktheorie und formalisiert die zugrunde liegenden Konzepte des Problems in der Prädikatenlogik. Die Ergebnisse hiervon erklären uns, ob Datenintegration möglich ist und wie schwierig es ist diese durchzuführen. Die Definitionen im Bereich der Datenintegration erscheinen oftmals abstrakt, was darauf zurückzuführen ist, dass diese sehr allgemein gehalten sind. Hiermit soll gewährleistet werden, dass sie auf möglichst viele Integrationsprobleme angewendet werden können.

Datenintegrationssysteme werden formal als <G,S,M> Tripel definiert, wobei G (*global*) das globale Schema ist, S (*source*) die heterogene Menge der Quellenschemata ist, und M (*mapping*) die Menge der Abbildungen ist, welches die Anfragen vom globalen Schema auf die Quellenschemata abbildet. G und S werden beide in einer Sprache über einem Alphabet ausgedrückt welche Symbole ihrer jeweiligen Beziehungen beinhalten. M besteht aus einer Menge von Behauptungen welche Auskunft darüber geben, wie Anfragen in G und Anfragen in S in Beziehung zueinander stehen. Stellt nun ein Benutzer eine Anfrage an ein integriertes Datensystem, so stellt er diese an G. M wandelt diese Anfrage dann in eine oder mehrere auf die Datenquellen anzuwendenden Anfragen um und schafft so eine Beziehung zwischen dem globalen Schema G und den Quellenschemata S.

Eine Datenbank über einem Schema ist definiert als eine Menge von Tabellen, jeweils eine für jede Relation. Die Abhängigkeiten der Relationen untereinander werden als Primärschlüssel-Fremdschlüssel Beziehungen abgebildet. Die Quellendatenbank (*source data base*) ist definiert als Menge von Mengen von Tupeln (Menge von Relationen), welche durch das Quellenschema S bestimmt wird. Obwohl die Quellendatenbank hier als eine Datenbank betrachtet wird kann es sich hierbei um eine Ansammlung getrennter Datenbanken handeln. Die zum virtuellen Vermittlerschema G gehörende Datenbank bezeichnet man als globale Datenbank (*global database*). Die globale Datenbank muss also gemäß den Behauptungen aus M korrekt auf die Quellendatenbanken abbilden. Die Rechtmäßigkeit dieser Abbildung ist abhängig von der

Art der Wechselbeziehungen zwischen G und S. Die beiden bekanntesten Modelle dieser Beziehungen sind *Global as View (GaV)* und *Local as View (LaV)* welche auch in Abbildung 1 bildlich dargestellt sind.

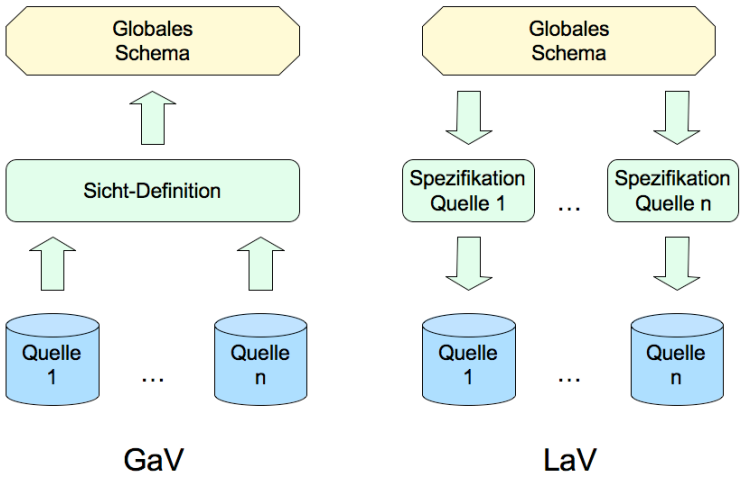

Fig. 1 Graphische Darstellung der Global as View und Local as View Szenarien [1]

3.1. Global as View (GaV)

Beim GaV ist die globale Datenbank als eine Menge von Sichten auf S modelliert. In diesem Fall verbindet M mit jedem Element aus G eine Anfrage auf S. Die Durchführung von Anfragen wird hierbei zu einem unkomplizierten Vorgang, da die Abbildungen zwischen G und S wohldefiniert sind. Die Komplexität dieses Modells liegt in der Implementierung eines Vermittlercodes (*mediator code*) welcher dem Integrationssystem ausführliche Anweisungen darüber gibt, wie ein bestimmtes Element aus der Quelldatenbank erreicht werden kann. Fügt man nach der Integration dem System weitere Daten- und Informationsquellen hinzu, so ist die Anpassung des Vermittlercodes mit wesentlichem Aufwand verbunden. Geht man allerdings von dem Fall aus, dass eine Änderung der Quellendaten unwahrscheinlich ist, so ist dieses Modell sehr empfehlenswert.

Als Beispiel könnte man sich hier einen Onlineshop vorstellen, welcher Produkte von mehreren Lieferanten anbietet. Bei der Entwicklung einer solchen Internetpräsenz (globale Sicht) wäre der größte Aufwand der Integration damit verbunden, den Vermittlercode zu erstellen, welcher bei einer Kundenanfrage zum richtigen Produkt in der Datenbank eines Lieferanten (Quelldaten) führen würde. Ist die Webseite einmal online, und der Vermittlercode entwickelt, so sind Anfragen keine Herausforderung mehr. Will man allerdings einen weiteren Lieferanten hinzufügen, so stellt die Änderung der Vermittlercodes einen sehr großen Aufwand dar. Nicht nur muss der Vermittlercode um den neuen Lieferanten erweitert werden, sondern darüber hinaus muss darauf geachtet werden dass die neuen Anfragen keine anderen Quellen ungewollt

beeinflussen können. Die Vermeidung von Konsequenzen von Anfragen auf den neuen Quelldaten für bereits bestehende Quellen stellt oft eine nicht unbeachtliche Herausforderung dar.

Betrachten wir uns zum GaV-Ansatz in Anlehnung an [17] folgendes anschauliche Beispiel: gegeben ist ein globales Schema mit zwei Relationen:

- `Student: Matrikelnummer, Name, Alter, Semester`
- `Adresse: Matrikelnummer, Ort`

Diese sollen als sicht auf die folgenden lokalen Schemata dargestellt werden:

- `Q1: Matrikelnummer, Name, Ort`
- `Q2: Name, Matrikelnummer, Alter`
- `Q3: Matrikelnummer, Alter, Semester`

Für die Adressen kann nur Quelle `Q1` herangezogen werden (in SQL-Syntax):

```
- CREATE VIEW Adresse AS
      SELECT Matrikelnummer, Ort
      FROM Q1
```

Studentendaten sind bis auf das Semester in der Quelle `Q2` und in Kombination in `Q1` und `Q3` enthalten:

```
- CREATE VIEW Student AS
      SELECT Matrikelnummer, Name, Alter FROM Q2
      UNION
      SELECT Q1.Matrikelnummer, Q1.Name, Q3.Alter
      FROM Q1, Q3
      WHERE Q1.Matrikelnummer = Q3.Matrikelnummer
```

Nebenbedingungen des globalen Schemas können übernommen werden. Falls beispielsweise nur Studenten aus Kaiserslautern berücksichtigt werden sollen, so ist die Sicht der Adressen:

```
- CREATE VIEW Adresse AS
      SELECT Matrikelnummer, Ort
      FROM Q1
      WHERE Ort="Kaiserslautern"
```

3.2. Local as View (LaV)

Beim LaV wird die Quellendatenbank als eine Menge von Sichten auf G modelliert. M bildet jedes Schema einer Relation von S auf eine Anfrage über G ab. Diese Beziehung ist nicht bijektiv, wodurch die exakte Abbildung von G auf S nicht länger wohldefiniert ist. Wie im folgenden Abschnitt veranschaulicht, muss sich bei diesem Modell der Anfrageprozessor der größten Herausforderung stellen, welcher eine globale Anfrage zur Laufzeit in eine für die Datenquellen verständliche Anfrage umwandeln muss.

Der große Vorteil vom LaV-Ansatz ist, dass das Hinzufügen von Datenquellen mit einem weitaus geringeren Aufwand verbunden ist als beim GaV-Ansatz. Der LaV-Ansatz sollte also bevorzugt werden wenn man nicht davon ausgehen muss, dass der Mediator G (*mediated schema*) sich im Laufe der Zeit verändert.

Betrachten wir noch einmal das Beispiel unseres Onlinevertriebs: Beim LaV-Ansatz würden zuerst die entsprechenden Quelldaten jedes Lieferanten über dem

globalen Schema definiert werden, falls diese noch nicht im globalen Schema vorhanden sind. Datenquellen können an sich beliebig hinzugefügt werden, ohne dass der Vermittlercode wie beim GaV-Modell verändert werden müsste. Die Herausforderung wird hier lediglich vom Programmierer auf den Prozessor übertragen, welcher erst bei einer konkreten Anfrage die Umwandlung von einer Kundenanfrage in eine konkrete Produktanfrage vornimmt und diese dann an die Lieferantendatenbanken (Quelldatenbanken) weiterleitet.

Ähnlich wie für den GaV-Ansatz wollen wir uns zum LaV-Ansatz in Anlehnung an [17] folgendes anschauliche Beispiel betrachten: gegeben sind drei lokale Datenquellen mit folgendem Schema:

- `Q1: Matrikelnummer, Name, Ort`
- `Q2: Name, Matrikelnummer, Alter`
- `Q3: Matrikelnummer, Alter, Semester`

Diese sollen auf folgendes globale Schema abgebildet werden:

- `Student: Matrikelnummer, Name, Alter`

Die Sichten der Quellen auf das globale Schema sind (in SQL):

```
- CREATE VIEW S1 AS
      SELECT Matrikelnummer, Name
      FROM Student
- CREATE VIEW S2 AS
      SELECT Name, Matrikelnummer, Alter
      FROM Student
- CREATE VIEW S3 AS
      SELECT Matrikelnummer, Alter
      FROM Student
```

Auch Assoziationen über mehrere Relationen des globalen Schemas können modelliert werden. Sei im globalen Schema eine weitere Relation enthalten, die Matrikelnummern und Wohnorte einander zuordnet:

- `Adresse: Matrikelnummer, Ort`

Dann lässt sich die Quelle `Q1` darstellen als:

```
- CREATE VIEW S1 AS
      SELECT Student.Matrikelnummer, Student.Name,
      Adresse.Ort
      FROM Student, Adresse
      WHERE Student.Matrikelnummer =
      Adresse.Matrikelnummer
```

Nebenbedingungen von Quellen werden direkt übernommen. Falls beispielsweise die Quelle `Q2` nur Studenten ab 25 Jahren enthält, würde die Sicht folgender entsprechen:

```
- CREATE VIEW S2 AS
      SELECT Matrikelnummer, Name, Alter
      FROM Student
      WHERE Alter >= 25
```

3.3. Global Local a View (GLaV)

Der *Global-Local-as-View*-Ansatz (GLaV) kann als Erweiterung des LAV-Ansatzes betrachtet werden und kombiniert die Vorteile der beiden Basisansätzen, indem er die Erhaltung von neu hinzugefügten Quellen, sowie die Unterstützung von komplexen Anfragetransformation und die Unabhängigkeit des globalen Schemas gewährleistet [2].

Im Gegensatz zum LaV, welcher nur beschränkte Formen der Prädikatenlogik benutzt, und GaV, welcher nur Sichtbeschreibungen definiert, werden beim GLaV-Ansatz flexible prädikatenlogische Konstrukte genutzt. Dies ermöglicht dass eine Sicht über Quellenrelationen als eine Sicht über globalen Relationen mit Quellendefinitionen dargestellt werden kann. Somit erlaubt der GLaV-Ansatz also die Ableitung von Daten über Quellenrelationen, welche über die Möglichkeiten des LaV-Ansatzes hinausgehen. Zudem können durch diesen Ansatz globale Schemata logisch verknüpft werden, was als Erweiterung des GaV-Ansatzes betrachtet werden kann [5].

Ein Problem des GLaV-Ansatzes ist die Tatsache, dass keine semi-automatische Spezifikation der Quellendefinition ermöglicht wird. Somit kann auch mit diesem Ansatz leider die Komplexität des LaV-Ansatzes nicht überwunden werden.

3.4. Both as View (BaV)

Relativ neu ist der *Both-as-View*-Ansatz (BaV). Hierbei wird versucht ein vereinigendes Rahmenwerk für die Ansätze GaV und LaV zu erstellen welches auf der Anwendung von umkehrbaren Schematrasformationssequenzen, so genannten *Pathways*, basiert [1]. Diese *Pathways* führen zum einen zu einer Definition des globalen Schemas als Sicht über den lokalen Schemata. Zum anderen ergeben sich die Definitionen der lokalen Schemata als Sichten über dem globalen Schema. Zusammengefasst heißt das, dass sowohl LaV als auch GaV Sichten aus den *Pathways* abgeleitet werden können. Umgekehrt können ebenfalls BaV *Pathways* aus GaV oder LaV Sichtdefinitionen abgeleitet werden. Darüber hinaus erlaubt dieser Ansatz durch schrittweise Modifizierbarkeit der *Pathways* die Entwicklung von lokalen sowie globalen Schemata[9].

Die Stärke dieses Ansatzes liegt darin, dass für primitive Transformationen automatisch Gegentransformationen gefunden werden können. Zwischen zwei Schemata, welche durch primitive Transformationen miteinander verbunden sind, besteht also eine bidirektionale Beziehung. Aus einer Reihe von solchen Beziehungen, welche verschiedene Schemata verbinden, können somit die GaV und LaV Sichten abgeleitet werden.

Der BaV-Ansatz verbindet also die Vorteile von GaV und LaV indem jedes Vorgehen, welches in GaV oder LaV Sichtdefinitionen möglich sind, auch durch den BaV-Ansatz gewährleistet wird[11]. Der Nachteil des BaV-Ansatzes liegt darin, dass die oben beschriebenen *Pathways* oft sehr feingranular sind, und dadurch eine Optimierung der Anfragen im Gegensatz zu den korrespondierenden GaV und LaV Sichtdefinitionen wesentlich aufwendiger ist. Man kann allerdings auch auf die von den *Pathways* abgeleiteten Sichtdefinitionen die Standardtechniken der Anfragenoptimie-

rung anwenden, so dass Anfragen in BaV-Systemen nicht wesentlich zeitineffizienter sind als in GaV- und LaV-Systemen.

Beispiele, welche die beiden vorstehenden Ansätze konkreter erläutern können, sind meist recht lang und komplex.

Es ist zu beobachten dass der *Global-as-View*-Ansatz zurzeit in der Praxis aufgrund seiner höheren Effizienz am meisten Anwendung findet. Neben den oben beschriebenen Mischformen gibt es noch zwei weitere Ansätze: der *Peer-Programming-Language*- und der *BYU-Global-Local-as-View*-Ansatz. Auf diese beiden etwas komplexeren Ansätze wird in dieser Seminararbeit nicht näher eingegangen. Die Referenz [1] erläutert beide Ansätze und zeigt deren Vor- und Nachteile.

4. Mechanismen der Integration – Replikationsorientierte vs. Virtuelle Integration

Die nachfolgend vorgestellten Integrationsmechanismen sind orthogonal zu den oben definierten Integrationsmodellen. Dies bedeutet dass die im Anschluss definierten Mechanismen beliebig mit den verschiedenen oben vorgestellten Modellen kombiniert werden können. Bei der Integration gibt es zwei mögliche Alternativen, die verteilten Datenmengen miteinander zu verbinden. Zum einen besteht die Möglichkeit, aus jeder Datenquelle die Daten in eine zentrale Datenbank zu kopieren, und mit den Kopien der verschiedenen Daten zu arbeiten. Zum anderen besteht aber auch die Möglichkeit, die verschiedenen Datenquellen bei Anfragen oder Änderungen anzusteuern, und mit den Originaldaten zu arbeiten. Beide Varianten haben Ihre Daseinsberechtigung, wie wir an den folgenden Vor- und Nachteilen der beiden Integrationsmodellen sehen werden.

4.1. Replikationsorientierte Integration

Die Tatsache, bei der Datenintegration Kopien der gesamten Daten in einer zentralen Datenbank abzulegen, wird in der Literatur auch sehr oft mit *Data Warehousing* bezeichnet. Ein *Data Warehouse* ist eine Ansammlung von digitalen Daten, welche so organisiert abgespeichert sind, dass optimal auf ihnen gearbeitet werden kann und Analysen und Berichte erstellt werden können. Bill Inmon, welcher als *„father of data warehousing“* weltbekannt ist, wählt in [8] folgendes als formale Systemdefinition: ein *Data Warehouse* ist eine Datenbank, welche

- *subjektorientiert* ist, was bedeutet, dass alle Daten, welche sich auf ein bestimmtes Thema einer abgebildeten Realwelt beziehen, miteinander verbunden sind
- *zeitvariant* ist, was bedeutet, dass Veränderungen an Daten gespeichert werden so dass Berichte darüber angefertigt werden können, wie sich die Daten im Laufe der Zeit verändert haben
- *permanent* ist, was heißt dass Daten in der Datenbank nie gelöscht werden dürfen, sondern für spätere Analysen und Berichte gespeichert werden, und

- *integriert* ist, und somit alle geschäftsrelevanten Unternehmensdaten in einer konsistenten Art und Weise abspeichert.

Die replikationsorientierte Integration wird nach dem *Extract, Transform, Load* (ETL) Prinzip durchgeführt. Hierbei werden in einer ersten Phase die Daten aus den einzelnen Datenquellen extrahiert und in einem zweiten Schritt dann in eine einheitliche Form transformiert, um so einheitlich in der Datenbank abgelegt werden zu können. In einem letzten Schritt werden die einheitlichen Daten dann in das Data Warehouse hochgeladen. Wie hier bereits zu erkennen ist, findet der Umwandlungsprozess für jedes Datum nur ein einziges Mal statt. Dies ist ein großer Vorteil dieser Integrationsvariante: nachdem die Daten einmal in ein einheitliches Format gebracht wurden, brauchen diese bei einer Anfrage nicht mehr umgewandelt zu werden. Dies beschleunigt natürlich das Arbeiten auf den Daten.

Ein Nachteil dieses Modells ist das Problem der Aktualisierung der replizierten Daten. Die verschiedenen Datenquellen werden zu einem bestimmten Zeitpunkt auf die zentrale Datenbank übertragen, und können in ihr dann weiterverarbeitet werden. Dies schließt aber nicht aus, dass auch lokale Anwendungen die Quelldaten weiter bearbeiten. Somit entstehen inkonsistente Daten in den verschiedenen Datenbanken. Zudem geschieht das Updaten der Quelldatenbanken nicht automatisch und muss bei Bedarf initiiert werden. Aufgrund der an sich doppelt abgelegten Daten ist dieses Modell darüber hinaus sehr speicheraufwändig.

4.2. Virtuelle Integration

Bei der virtuellen Integration werden die integrierten Daten nicht wie beim Data Warehousing zentral abgelegt, sondern es wird weiterhin auf den originalen Daten gearbeitet. Hierzu wird ein Portal erstellt, in dem man Anfragen an die verschiedenen Datenquellen stellen kann. Diese Anfragen werden dann abhängig von der Datenquelle umgewandelt, um anschließend an die Quelldatenbank weitergeleitet werden zu können. Das Portal gilt also sozusagen als Abfertigungszentrale (*Dispatcher*) für die Anfragen.

Der Vorteil dieser Art der Integration ist ganz klar die Speichereffizienz. Indem nicht von jedem Datum eine Kopie angefertigt werden muss, sondern gleich auf die Originaldaten zurückgegriffen werden kann, werden Unmengen an Speicherkapazität, welche für das replikationsorientierte Modell erforderlich wären, überflüssig. Darüber hinaus wird, indem auf den Originaldaten gearbeitet wird, das Problem des Updatens der Daten, welches beim replikationsorientierten Modell besteht, ebenfalls beseitigt. Sowohl das globale Portal, wie auch jedes einzelne lokale Anwendungsprogramm arbeiten ständig mit den aktuellen Daten.

Klarer Nachteil dieses Modells ist die im Gegensatz zum replikationsorientierten Modell relativ schwache Laufzeiteffizienz. Da jede Anfrage in eine für das lokale Datensystem verständliche Anfrage umgewandelt werden muss, wird der Anfragevorgang stark verlangsamt.

Jede dieser beiden Arten der Integration hat ihre Vor- und Nachteile, mit welchen man sich vor der Integration auseinander setzen muss. Oft werden auch Mischformen der beiden Modelle eingesetzt um bestmöglich die Vorteile der verschiedenen Model-

le verbinden zu können. Im Folgenden werden wir näher darauf eingehen, welche Probleme sich bei der Integration stellen und wie diese gelöst werden können.

5. Problemstellung bei der Integration

Wie kann man nun bei einem Integrationsvorgang konkret vorgehen? Ziel der Informationsintegration ist es, Middlewaresysteme zu liefern, welche

- es Anwendungen ermöglichen, auf Daten zuzugreifen, als würde es sich um lokale Daten handeln, welche in einer einzigen Datenbank abgelegt sind ohne dass man sich über deren Ort und Qualität Gedanken machen müsste.
- anspruchsvolle Such-, Transformations- und Analysedienste anbieten
- die Interaktion mit anderen Middlewaresystemen unterstützen (wie z. B. Datentransfersysteme oder Web Services)

Diese Ziele sind recht anspruchsvoll. Zahlreiche Unternehmen beklagen sich immer häufiger, dass die von ihnen benötigten Informationen nicht auffindbar seien. Allzu oft befinden sich die Daten über das gesamte Unternehmen verstreut an zahlreichen Orten und in diversen Datenspeichern. Sogar wenn man Zugriff auf die benötigten Daten hat, ist es oft noch sehr kompliziert mit diesen Daten zu arbeiten oder diese miteinander zu vergleichen, da sie oft in inkompatiblen Formen abgelegt wurden. Ergebnisse von Datentransformationen müssen dann selbst wieder analysiert und wiederum in die verschiedenen endgültigen Formate umgewandelt werden um sie dann an einem bestimmten Ort im Unternehmen abspeichern zu können.

Betrachtet man nun die Vorteile welche Informationsintegration in den verschiedensten Branchen bieten kann, so wird schnell klar dass es sich lohnt, die oben erläuterten Schwierigkeiten zu überwinden.

Um die Vorteile der Informationsintegration zu erreichen, muss sowohl ein Informationsintegrationsverwaltungssystem (*information integration management system*) als auch ein Entwicklungsprozess erarbeitet werden um dieses zu nutzen. Zudem sollten Werkzeuge genutzt werden um diesen Entwicklungsprozess zu unterstützen. Einige Anforderungen müssen erfüllt werden um eine Umsetzung der Informationsintegration zu gewährleisten [3]. Die folgenden drei Dimensionen können diesbezüglich als Schlüsselfaktoren identifiziert werden [10,16]

- Bei der Informationsintegration muss auf die *Datenheterogenität* geachtet werden. Oft sind die Daten unterschiedlich strukturiert was bei der Integration unbedingt beachtet werden muss. Hierbei müssen strukturierte, semistrukturierte und unstrukturierte Daten gehandhabt werden. Solche Daten liegen oft in den verschiedensten Datenmodellen oder Formaten vor, wie z. B. relationale Daten, XML, Text oder aber auch Objekte (Bilder, Videos, etc). Um diese Daten integrieren und mit ihnen arbeiten zu können müssen die Abhängigkeiten zwischen den verschiedenen Datenformaten dargestellt werden. Darüber hinaus müssen Ansätze zur Handhabung, Manipulation und zur Suche auf diesen Daten erarbeitet werden, da diese stark von dem jeweiligen Datenformat abhängig sind.

- Die *Vereinigung und Verteilung von Daten* beschäftigt sich mit der Tatsache, dass die Daten die integriert werden sollen, normalerweise auf eine Vielzahl von verschiedenen Datenquellen verteilt sind, welche oftmals über das gesamte Unternehmen verstreut sind. Für dieses Problem gibt es zwei Lösungen. Konsolidierung ist eine Möglichkeit, bei der man die verschiedenen Daten aus deren Quellendatenbank in eine neue Datenbank kopiert. Es handelt sich hierbei meistens um eine *read-only* Datenbank, welche normalerweise nicht mehr mit den ursprünglichen Datenbanken verlinkt ist. Diese Variante, welche auch als replikationsorientierte Integration bekannt ist, verlangt meistens weitere Schritte zur Transformation und Bereinigung der Daten. Ein alternativer Ansatz ist die Nutzung von verteilten Anfragen, welche es erlaubt die Daten in ihren ursprünglichen Quellen zu belassen und die Daten bei Bedarf abzurufen. Dieses Verfahren ist auch unter der oben beschriebenen virtuellen Integration bekannt. Wohlgemerkt haben beide Verfahren vor und Nachteile bezüglich Zeiteffizienz und Aktualität der Daten, und werden meist alternativ zueinander eingesetzt. Sehr wichtig ist, dass in beiden Fällen die initiale Struktur der Daten nicht verändert werden darf, so dass weiterhin bestehende Anwendungen auf den Daten arbeiten können.
- Ein zentrales Ziel der Datenintegration ist das Schaffen von *business intelligence*. Komplexe Analysen sowie Anfragen auf heterogenen Daten müssen durchgeführt werden, um dem Unternehmen wertvolle Informationen zur Verfügung zu stellen. Diese Informationen sind von fundamentaler Bedeutung für Unternehmensentscheidungen und können dem Unternehmen einen beachtlichen Wettbewerbsvorsprung verschaffen. Man kann entweder einzelne Informationen oder aber auch Ansammlungen von Daten analysieren. Operationen wie die Klassifikation oder Bewerten von Daten und Informationen fallen unter die erste Kategorie, Gruppieren und Verknüpfungen von Daten und Informationen fallen unter die zweite. Transaktionen der ersten Kategorie werden oft als aktive Analysetätigkeiten betrachtet und werden durch das Hinzufügen oder Ändern von Daten ausgelöst. Diese Transaktionen beziehen oft aktuelle oder Abstammungsdaten, was erklärt wieso eine feste Integration dieser Analyse Transaktionen im Informationsintegrationsverwaltungssystem (*information integration management system - IIMS*) erforderlich ist. Andere Analyse Szenarien nutzen die flexible Unterstützung von asynchronen Verarbeitungsparadigmen, vor allem dann wenn die Analyse ohnehin nicht zur Laufzeit (*transactional speed*) ausgeführt werden kann, beziehungsweise welche menschliche Interaktion erfordern (z. B. zur Bearbeitung von Ausnahmen (*exceptions*)). Für die verschiedenen Stufen von Heterogenität bestehen verschiedene Analyse und Anfragetransaktionen. Durch Informationsintegration sollen die Ergebnisse dieser Analysen zu einem Ganzen zusammengefügt werden und so das Gesamtergebnis für das Unternehmen verbessern.

Zusätzlich zu den hier vorgestellten Problemen, werden hohe Ansprüche an die IT-Lösungen gestellt, vor allem im Hinblick auf Interoperabilität. Zudem muss das IIMS *robusten Austausch und Tranformation von Daten*, basierend auf *offenen und plattform-unabhängigen Standards* ermöglichen [4]. Im Bereich der Datenspeicherung gewinnen die Sprache XML als auch Internet Dienste (Web-Services) immer mehr an Bedeutung. Zudem muss das IIMS dazu im Stande sein, in einem breiteren Rahmen

von Geschäftsprozessen und *enterprise application integration* (EAI) Architekturen zu arbeiten.

6. Formen von Heterogenität

Bei der Informationsintegration spielt vor allem die Art der Datenheterogenität eine wichtige Rolle. Im Folgenden werden die verschiedenen Erscheinungsformen der Heterogenität vorgestellt und klassifiziert.

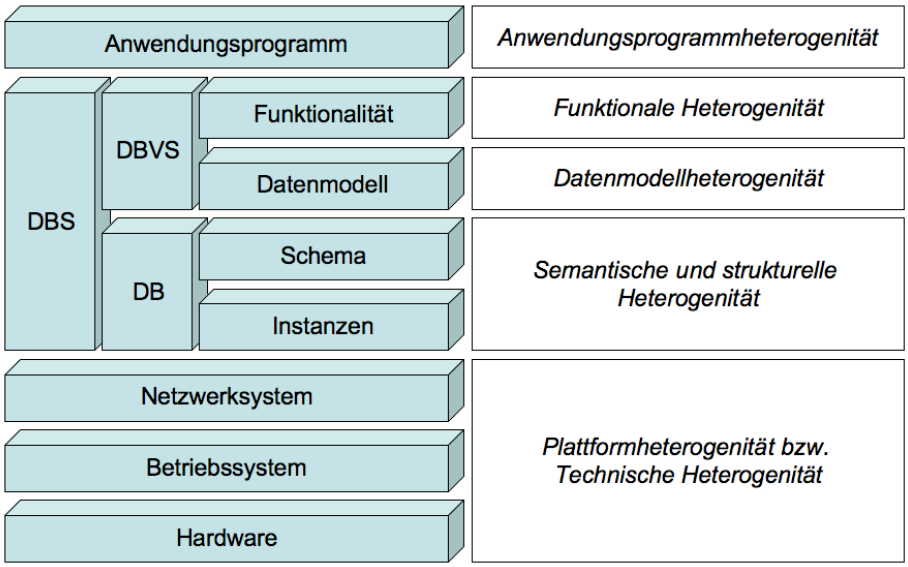

Fig. 2. Verschiedene Formen von Heterogenität [15]

Um die verschiedenen Formen von Heterogenität voneinander abgrenzen zu können, kann man sich z. B. an den verschiedenen Schichten der Systemarchitektur orientieren [14]. In Abbildung 2 werden diese Heterogenitätsformen anschaulich dargestellt. Betrachten wir zunächst die verschiedenen Teile der Systemarchitektur: Informationsverarbeitungsprogramme arbeiten auf Datenbanksystemen, welchen Netzwerksysteme, Betriebssysteme und Hardware zugrunde liegen. Unterschiede in den drei untersten Schichten, welche den Datenbanksystemen als Plattform dienen, wie z. B. unterschiedliche Betriebssysteme oder Netzwerke, führen zu Plattformheterogenität. Gibt es Unterschiede auf der Anwendungsprogramm-Ebene, so spricht man von Anwendungsprogrammheterogenität. Das Datenbanksystem (DBS) kann wiederum in Datenbankverwaltungssystem (DBVS) und in die eigentliche Datendank (DB) unterteilt werden. Das DBVS stellt auf der einen Seite die Funktionalität (z. B. Mehrbenutzerbetrieb, Transaktionsverwaltung oder Fehlerbehandlungsstrategien) des DBS gemäß allgemeinen Konzepten sicher. Gibt es auf dieser Ebene Unterschiede, so ist

die Rede von funktionaler Heterogenität. Auf der anderen Seite wird im DBVS aber auch das Datenmodell festgelegt (z. B. relational oder objektorientiert). Werden also in den verschiedenen Informationsquellen unterschiedliche Datenmodelle benutzt, so spricht man von Datenmodellheterogenität. Die Datenbank kann wiederum in zwei unterteilt werden: ein Schema, welches festlegt nach welcher Struktur die in der Datenbank abgelegten Datenobjekte gespeichert werden sollen sowie die eigentlichen Daten selbst, welche man auch als Instanzen bezeichnet. Treten hierbei Unterschiede in der Bedeutung von Schemaelementen oder auf Daten selbst auf, so handelt es sich um semantische Heterogenität (z. B. Relationen und Attribute im relationalen Datenmodell). Sind die Schemata unterschiedlich strukturiert, so handelt es sich um strukturelle Heterogenität.

Die größte Herausforderung, welche die Datenheterogenität mit sich bringt, ist die Lösung der Konflikte, welche sich dadurch ergeben, dass Daten aus heterogenen Quellen, welche denselben Sachverhalt beschreiben, sich widersprechen. Betrachten wir uns in den folgenden Abschnitten die drei Hauptarten von Heterogenität noch einmal genauer.

6.1. Datenmodellheterogenität

Datenmodellheterogenität liegt vor, wenn Quellendaten gemäß verschiedener Datenmodelle abgelegt sind. Dies kann man z. B. an der unterschiedlichen Mächtigkeit der Datenmodelle erkennen. Einige Datenmodelle unterstützen unter Umständen Generalisierung, Aggregation oder Assoziationen wobei andere Datenmodelle diese Konzepte nicht unterstützen. Je größer solche Unterschiede zwischen zwei oder mehreren Datenmodellen sind, umso stärker ist der Grad der Datenmodellheterogenität. In einem objekt-orientierten Datenmodell existiert z. B. das Konzept der Generalisierung was im relationalen Datenmodell nicht der Fall ist. Obwohl in beiden Datenmodellen der gleiche Sachverhalt einer realen Welt abgebildet werden kann, sind die Daten aufgrund der heterogenen Datenmodelle strukturell sehr unterschiedlich. Darüber hinaus müssen natürlich auch die Anfragesprachen den Datenmodellen angepasst werden.

6.2. Semantische Heterogenität

Semantische Heterogenität ist meistens darauf zurückzuführen, dass Daten in verschiedenen Datenquellen nach logisch erscheinenden Schemata von verschiedenen Datenbankentwicklern abgelegt werden. Versucht man dann diese Datenquellen zu integrieren, so steht man oft vor dem Problem, dass auf gleiche Art und Weise abgelegte Daten nicht identisch zu interpretieren sind. Hierzu stellt man sich am besten folgendes Beispiel vor.

In einer Datenbank sind verschiedene Personen mit Vor- und Nachnamen abgespeichert. In der ersten Datenbank wird im ersten Attribut der Vorname und im zweiten Attribut der Nachname gespeichert. In der zweiten Datenbank werden der Nachname im ersten und der Vorname im zweiten Attribut gespeichert. Da nun aber die Attributsbezeichnungen in verschiedenen Datenbanken, welche z. B. als Textdateien

abgespeichert sind (CSV-Dateien, Excel Tabellenkalkulationsdokumente ohne Spaltenbezeichnungen), nicht mitgespeichert werden, hätte man für eine Person namens Peter Schmidt die beiden folgenden Einträge <Peter, Schmidt> und <Schmidt, Peter>, welche anhand der entsprechenden Semantiken interpretiert werden müssten. Es ist leicht erkennbar dass dieses Problem des fehlenden Kontextes bei der Integration von den beiden Datenbanken zu bewältigen wäre.

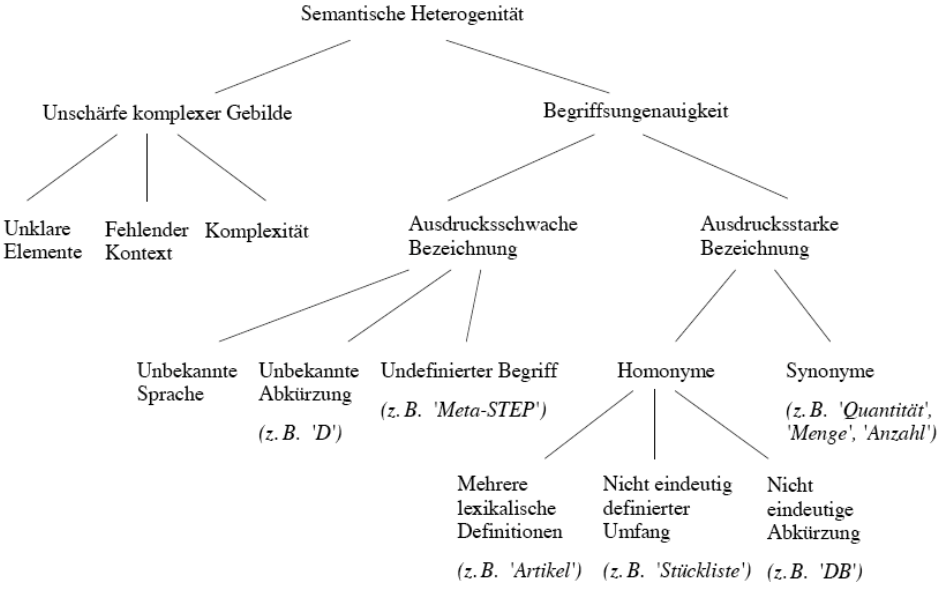

Fig. 3. Klassifikation semantischer Heterogenität (nach [14])

In Abbildung 3 werden die diversen Varianten von semantischer Heterogenität, welche auftreten können, veranschaulicht.

Die verschiedenen Komponenten, in welchen die Daten abgelegt sind (z. B. Tabellen, Spalten, Klassen, etc.) werden als Schemaelemente bezeichnet. Die Daten können also anhand dieser Schemaelemente, welchen sie zugewiesen sind, definiert werden. Somit stellen diese Schemaelemente die Schlüsselfaktoren der Analyse bei der Datenintegration dar. Gegeben durch oftmals fehlende Dokumentation, erschwert die semantische Heterogenität oft den Integrationsvorgang. Ein weiteres, oft schwer erkennbares Problem ist, dass die Definitionsbereiche von Daten in verschiedenen Quellen nicht übereinstimmen (z. B. verschiedene Währungen). Diese Probleme machen ersichtlich, dass die verschiedenen Datenquellen vor der Integration sorgfältig analysiert werden müssen, um spätere Konflikte zu vermeiden.

Aufgrund diverser Einflussfaktoren ist die Beseitigung von semantischer Heterogenität nicht komplett automatisierbar, so dass meist Personen die Interpretation durchführen müssen. Es ist allerdings möglich automatische Verfahren unterstützend einzusetzen, um so diese Art der Heterogenität zu überbrücken.

6.3. Strukturelle Heterogenität

Im Abschnitt über Datenmodellheterogenität wurden auf die strukturellen Konsequenzen von verschiedenen Datenmodellen hingewiesen. Aber auch bei Schemata, welche die gleichen Datenmodellen benutzen aber unabhängig voneinander entwickelt wurden, sind immer wieder strukturelle Differenzen zu beobachten. Wie sich Daten strukturell voneinander unterscheiden können, kann man anhand des folgenden Beispiels (angelehnt an [13]) leicht erkennen.

Gehen wir davon aus, dass wir Personen mit Vor-, Nachnamen und Geschlecht in unserer Datenbank abspeichern wollen, um später Anfragen bezüglich des Geschlechts durchführen zu können. Hierzu bieten sich mehrere Varianten an:

- Man kann die Daten als Relation abspeichern. Hierzu würde man für jedes Geschlecht eine Tabelle anlegen:

Tabelle A: männliche Personen

Vorname	**Nachname**
Peter	Schmidt

Tabelle B: weibliche Personen

Vorname	**Nachname**
Johanna	Wunder

- Alternativ hierzu kann man gleichen Sachverhalt aber auch anhand von Attributsausprägungen darstellen. Hierzu würden wir für eine Person zusätzlich die beiden Attribute männlich und weiblich einführen und je nach Geschlecht das Attribut auf 1 bzw. 0 setzen.

Tabelle A: Personen

Vorname	**Nachname**	**männlich**	**weiblich**
Peter	Schmidt	1	0
Johanna	Wunder	0	1

- Als dritte Variante würde sich anbieten, das Geschlecht als ein Attribut abzuspeichern. Hierbei hätte man an Stelle von zwei binären Attributen nur noch ein Attribut, welches die Werte männlich oder weiblich annehmen kann.

Tabelle A: Personen

Vorname	**Nachname**	**Geschlecht**
Peter	Schmidt	männlich
Johanna	Wunder	weiblich

Diese verschiedenen strukturellen Darstellungen veranschaulichen, wie es sogar bei gleichem Datenmodell zu struktureller Heterogenität kommen kann. Die Verwendung verschiedener Elemente eines Datenmodells zur Darstellung des gleichen Sachverhalts, welche in den obigen Beispielen veranschaulicht wurde, wird in der Literatur als *schematische Heterogenität* bezeichnet. Die schematische Heterogenität ist ein Teil der strukturellen Heterogenität, bei welcher unterschiedliche Elemente eines Datenmodells zur Darstellung eines gleichen Sachverhalts genutzt werden. Als strukturell heterogen werden aber auch verschiedene Darstellungen angesehen, welche die gleichen Elemente eines gemeinsamen Datenmodells benutzen. Dies soll das folgende Beispiel kurz illustrieren:

Daten kann man entweder in einer einzigen Tabelle abspeichern, man kann sie aber auch auf beliebig viele Tabellen aufteilen. Die beiden folgenden Darstellungen bilden beide denselben Sachverhalt ab, ohne hierbei unterschiedliche Elemente des Datenmodells zu nutzen. Trotzdem handelt es sich hierbei um strukturell heterogene Schemata.

Tabelle A: Buch (Schema 1)

ID	Titel	Autor_ID	Autor
12345	Datenbankanwendungen	1	Härder

Tabelle A: Buch (Schema 2)

ID	Titel	Autor_ID
12345	Datenbankanwendungen	1

Fremdschlüsselbeziehung →

Tabelle B: Autor (Schema 2)

Autor_ID	Autor
1	Härder

In Schema 1 wird zu jedem Buch der Autor mit gespeichert, wobei sich alle Informationen in einer Tabelle befinden. In Schema 2 verfügt jeder Bucheintrag über einen Fremdschlüssel, welcher auf einen bestimmten Eintrag in der Tabelle der Autoren verweist. Auch dieser Fall von struktureller Heterogenität muss im Falle einer Integration gelöst werden.

In den weiteren Vorträgen dieser Seminarreihe wird näher auf die verschiedenen Probleme eingegangen und Lösungen für die verschiedenen Arten von Datenheterogenität dargelegt.

7. Integrationsmethoden und Prozesse

Die Integration von Datensystemen gestaltet sich in einer ersten Phase in vier Punkten [7]: Interpretation der Daten, Bereinigung der Daten, Transformation der Daten und Verbinden der Daten. In einer zweiten Phase können dann verschiedene Datensysteme miteinander verbunden werden. Im Folgenden werden die einzelnen Schritte näher erläutert.

7.1. Interpretieren der Daten

Daten werden zwischen Unternehmen in immer massiveren Volumen, in einer Vielzahl verschiedener Formate und mit immer höheren Geschwindigkeiten ausgetauscht, sehr häufig ohne dass man Einblick in die Datenquellen oder in die Struktur der Daten hat. Komplexe Transaktionen werden zwischen dem Unternehmen und seinen Kunden und Partnern getätigt, Informationen werden innerhalb des Unternehmens hin und her gereicht, jedoch können wichtige Unternehmensentscheidungen nur richtig getroffen werden, wenn die empfangenen Informationen auch richtig interpretiert werden.

Der erste wichtige Prozess bei der Datenintegration ist das durchkämmen der zu integrierenden Daten, um das Geheimnis des Inhalts und der Struktur der Quellendaten zu lüften und die Daten interpretieren zu können. Diese erste Phase der Datenintegration kann mittlerweile auch bereits mit Hilfe von Softwarelösungen automatisiert werden, welche Einblicke in die Unternehmensinformationen von verschiedensten Blickpunkten verschaffen können. Hierbei können innerhalb einer relationalen Datenbank (z. B. IBM DB2 mit IBM WebSphere Information Integrator) Schemata auf externe Datenquellen erstellt werden und als virtuelle Relationen integriert werden.

Dieser erste Schritt liefert nun dann eine Ansammlung interpretierter Daten, welche in einem nächsten Schritt von Redundanzen oder fehlerhaften Daten befreit werden können.

7.2. Bereinigen der Daten

Die Zahl der Akteure, welche Einfluss auf Unternehmensentscheidungen haben, steigt ständig an. Immer häufiger muss auf Kunden und Partnerinformationen zurückgegriffen werden, um adäquate Entscheidungen treffen zu können. Mit der ansteigenden Zahl der Einflussnehmer steigt natürlich auch die Menge an falschen Informationen, welche zu folgeschweren Fehlentscheidungen führen könnte. Um den vollen Nutzen aus den Informationen ziehen zu können, müssen die Daten also bereinigt werden: Daten müssen identifiziert, standardisiert, aneinander angepasst und von Redundanzen befreit werden.

Auch für diesen Prozess können Softwarelösungen genutzt werden, welche Teile des Vorgangs automatisieren können. Solche Anwendungen können z. B. Datenquellen durchlaufen, diese von Redundanzen befreien und zusammenhängende Daten miteinander verknüpfen. Solche Verknüpfungen können z. B. durch Einführung von Fremdschlüsselbeziehungen erreicht werden. Ziel ist es für jede Information einen einzigen hochwertigen Datensatz in der Datenbank zu speichern, welcher den zuständigen Personen bei Bedarf zur Verfügung steht.

Das Bereinigen von Daten kann auf jeder Ebene des Unternehmens angewandt werden, sei es um Kunden- oder Lieferantendaten und -informationen zu bereinigen oder aber auch um Ordnung in die innerbetrieblichen Daten wie Mitarbeiter- oder Produktinformationen aufzubereiten.

Die Bereinigung von Daten ist unabdinglich bei der Datenintegration und ist der einzige Weg zu hochqualitativen Daten und Informationen. Zudem ermöglicht die Bereinigung von Daten dem System eine einheitliche Sicht auf die Unternehmensinformationen welche einen wichtigen Teil des Geschäftskapitals darstellen. Diese nun

von Redundanzen und Fehlern bereinigten Daten bieten nun die Basis für die nächste Etappe: die Transformation der Daten.

7.3. Transformation der Daten

Geschäftsinformationen erreichen, durchqueren und verlassen das Unternehmen, vergleichbar mit dem menschlichen Organismus. Um Erfolge erzielen zu können muss das Unternehmen in diesen Fluss eingreifen indem es Informationen in Daten der erforderten Qualität und Format umwandelt. Die strukturierten Daten können so zum richtigen Zeitpunkt innerhalb oder außerhalb des Unternehmens an die Zielgruppen weitergeleitet werden.

Auch existieren mittlerweile hoch effiziente IT Werkzeuge, welche große Volumina von komplexen Daten transformieren und anschließend weiterleiten können.

Nachdem die Daten nun in erforderlichem Format und angemessener Qualität vorliegen, können diese in einem nächsten Schritt miteinander verbunden werden.

7.4. Verbinden der Daten

Das Verbinden der verschiedenen Datenquellen ist einer der wichtigsten Schritte der Datenintegration. Die Art, Geschäfte zu machen, kann durch die Integration heterogener Daten grundlegend verändert werden. Durch den nahtlosen Zugriff auf Informationen von Personen und Anwendungen können Fehlentscheidungen vermieden werden.

Auch hierzu wird Software eingesetzt, welche es ermöglicht über ein einziges Portal auf mehrere Datenquellen zuzugreifen. Diese Software verbindet die einzelnen Datenquellen und erlaubt es Personen und Anwendungen auf Informationen zuzugreifen, unabhängig davon in welchem Format und an welchem Ort sie abgelegt sind. Darüber hinaus behalten die einzelnen Datenquellen ihre Autonomie und können mit den auf ihnen arbeitenden Anwendungen weiter bestehen.

Zusätzlich zu diesen Basisfunktionalitäten von Softwarelösungen gibt es Werkzeuge, welche zusätzlich ein industriekompatibles SQL-Paradigma nutzen, um so Datenquellen auf einer breiteren Ebene integrieren zu können.

7.5. Verbinden von verschiedenen Datensystemen

Durch die immer größer werdende geographische Verteilung von Unternehmen und Konsolidierungen wird ein Zusammenschließen von mehreren Datensystemen immer attraktiver. Verschiedene Datensysteme, welche oft bereits eine Vielzahl von Datenquellen miteinander verbinden, müssen miteinander verlinkt werden, um so globale Geschäftsentscheidungen treffen zu können. Auch hier gibt es mittlerweile Softwarelösungen, welche Informationen über die Grenzen der Systeme hinaus miteinander verbinden zu können. Diese Lösungen ermöglichen einen direkten Zugriff auf alle relevanten Datenquellen der verbundenen Systeme und erlauben die Synchronisation der unvereinbaren Datenquellen. Diese so genannten Vernetzungslösungen können

für einzelne Anwendungen genutzt werden, können aber auch ganz allgemein zur Verdichtung der gesamten Unternehmensdaten und -informationen eingesetzt werden.

8. Implementierung in der realen Welt

In den vorangehenden Abschnitten wurden Techniken, Modelle und Prozesse erläutert, welche während eine Integration beachtet oder befolgt werden müssen. Viele dieser Punkte mögen recht abstrakt wirken, und der übergroße Aufwand der einzelnen hier beschriebenen Schritte mag übertrieben und ungerechtfertigt sein. Wozu sollte man Unmengen an finanziellen Mitteln investieren, um eigentlich außer Komfort keinen Mehrwert zu erzielen, vor allem wenn man bedenkt dass die Integration verteilter und unkompatibler Systeme mehrere Millionen Euro kostet?

Diese Bedenken lassen sich aber sehr schnell beseitigen, wenn man sich ein Beispiel der realen Welt vor Augen führt.

Wachovia ist mittlerweile die viertgrößte Bank der Vereinigten Staaten von Amerika. Das war aber nicht immer so. Ursprünglich in Charlotte basiert, fusionierte Wachovia im Jahr 2001 mit der First Union. Im darauf folgenden Jahr dann übernahm Wachovia das *Prudential's-Retail-Brokerage*-Unternehmen. Das Unternehmen wuchs rasant, fusionierte 2004 mit der South Trust Corporation und ist heute eine der größten Banken der USA. Das Unternehmen, welches im Jahr 2000 nur etwa 21,000 Mitarbeiter zählte beschäftigt heute knapp unter 100,000 Leute. Die Angebote sind mittlerweile auch sehr breit gefächert: vom Kapitalmanagement über Investment Banking bis hin zu Krediten für Unternehmen und Privatleute bietet die Wachovia alles an. Die Fusionen aber stellten viele Geschäftsbereiche vor große Herausforderungen: vor allem jene, welche über umfangreiche Kundeninformationen verfügen müssen, um Auskünfte geben zu können, hatten Probleme schnell über alle erforderlichen Daten zu verfügen, welche für eine angemessene Beratung nötig gewesen wären. Aber nicht nur Servicepersonal stand vor einer großen Herausforderung. Vor allem auch die IT Abteilungen der ehemaligen isolierten Unternehmen, welche sich zu einer neuen zentralen IT Abteilung zusammenschlossen, hatten mit einer Vielzahl verschiedener Datenformate zu kämpfen. Millionen von gescannten Dokumenten, Schecks und Unterschriften waren in verschiedenen meist verschlüsselten Formaten in verschiedenen Datenbanken abgelegt. Daneben gab es vor allem bei den kleineren Übernahmen Unmengen an Dokumenten, welche nur in Papierform vorlagen.

Um all diese Probleme zu lösen beauftragte Wachovia die IBM und veranlasste die Anfertigung einer an das gegebene Problem angepasste Lösung. Man entschied sich für eine virtuelle Integration, welche mit Hilfe eines zentralen Portals realisiert werden sollte. Neben einem Portal welches die verschiedenen Datenquellen miteinander verbinden sollte, mussten vor allem Desktopanwendungen erstellt werden, mit denen die Mitarbeiter der einzelnen Abteilungen mühelos auf die benötigten Informationen zugreifen können. Je leichter den Mitarbeitern die Kundeninformationen zugänglich waren, umso besser konnte Wachovia ihrem Ruf als kundenorientiertes Unternehmen gerecht werden.

Den genauen Preis der gesamten Integration der verschiedenen Informationsquellen bei Wachovia ist leider nicht bekannt. Bekannt ist nur, dass Wachovia in den

beiden ersten Jahren mit dem integrierten System Ersparnisse in Höhe von 2,3 Millionen Dollar. Darüber hinaus sind Datenanfragen um 50% gestiegen, was auf eine höhere Aufmerksamkeit den Kunden gegenüber deuten lässt. Für jede weitere Datenquelle, um welche das neue System erweitert wird, rechnet man mit zusätzlichen Gewinnen von einer Million Dollar. Diese Zahlen machen klar, wie unabdinglich eine Integration bei einem Zusammenschluss von mehreren Unternehmenseinheiten ist.

9. Ausblick

In dieser Ausarbeitung wurde zahlreich auf die Bedeutung von Integration eingegangen und verschiedene Vor- und Nachteile diverser Ansätze erläutert. In den kommenden Vorträgen dieser Seminarreihe wird näher auf die verschiedenen Konzepte eingegangen und konkrete Lösungen vorgestellt, wie bei der Integration vorgegangen werden kann.

Abschließend sollte man sich vor Augen halten, dass mit der momentanen wirtschaftlichen Entwicklung und durch den Einfluss von Globalisierung Unternehmen welche über den Globus verteilt sind immer näher zusammenwachsen, kooperieren oder gar fusionieren. Diese Verschmelzungen werden mit Sicherheit in den nächsten Jahren nicht abnehmen, ganz im Gegenteil. Datenintegration wird dadurch eine immer stärkere Rolle spielen. Daten sind allgegenwärtig.

Literaturverzeichnis

[1] G. Bulgu, S. Mavridou, “Klassische Datenintegration”, Seminar zum Thema Datenintegration, 2004.

[2] A. Cali, D. Calvanese, G. De Giacomo, M. Lenzerini, “On the Expressive Power of Data Integration Systems”, Dipartimento di Informatica e Sistematica, Universität von Rom, 2002

[3] S. Deßloch, E. Lin, N. Mattos, D. Wolfson, K. Zeidenstein, “Towards an Integrated Data Management Platform for the Web”, Datenbankspektrum 2, 5-13, 2002.

[4] S. Deßloch, A. Maier, N. Mattos, D. Wolfson, “Information Integration – Goals and Challenges”, Datenbanken-Spektrum 1, 1-6, 2003.

[5] M. Friedman, A. Levy, T. Millstein, “Navigational Plans for Data Integration”, The 16th National Conference on Artificial Intelligence, Orlando, 2004.

[6] T. Härder, “Datenintegration”, Informatik Forschung und Entwicklung, Vol. 17, No. 3, 2002.

[7] “The power of Integrated Information”, IBM Software Group, 2005.

[8] W. H. Inmon, “Building the Data Warehouse”, 3rd edition, 2002.

[9] E. Jasper, N. Tong, P. McBrien, A. Poulovassilis, “View Generation Optimisation in the AutoMed Data Integration Framework”, 2003.

[10] A. D. Jhingran, N. Mattos, H. Pirahesh, “Information Integration: A research agenda”, IBM Systems Journal 41, No. 4, special issue on Information Integration, 2002.

[11] P. McBrien, P.Poulavassilis, “Data Integration by Bi-Directional Schema Transformation Rules”, 2003.

[12] R. Meersmann, A. Sheth, ACM SIGMOD Record, vol. 31, No. 4, special section on Semantic Web and Data Management, 2002.

[13] F. Naumann, “Informationsintegration 1”, Vorlesung an der Humboldt-Universität Berlin, 2003.

[14] G. Sauter, “Interoperabilität von Datenbanksystemen bei struktureller Heterogenität”, Dissertation zu Datenbanken und Informationssystemen Band 47, 1998.

[15] J. O. Schnabel, “Formen der Heterogenität”, Seminararbeit zum Thema Grundlagen webbasierter Informationssysteme, 2004.

[16] A. P. Sheth, J. A. Larson, "Federated Database Systems for Managin Distributed, Heterogenous, and Autonomous Databases", ACM Computing Surveys 22 (3), 183-236, 1990.

[17] J. Ullmann, "Information Integration Using Logical Views", Proceedings of the 6th International Conference on Database Theory, p. 19-40, 1997.