Otto-von-Guericke-Universität Magdeburg

Fakultät für Informatik
Institut für Technische Informationssysteme

Diplomarbeit

Identifikation semantisch äquivalenter Datenbankobjekte in Multidatenbanksystemen

Name	:	Thomas Müller
geboren in	:	Halberstadt
geboren am	:	02.08.1972
eingereicht am	:	29.08.1997
Seminargruppe	:	IF DB 92

Betreuer: Prof. Dr. habil. Gunter Saake, Universität Magdeburg, ITI
Dipl.-Inf. Ingo Schmitt, Universität Magdeburg, ITI

Müller, Thomas:
Identifikation semantisch äquivalenter
Datenbankobjekte in Multidatenbanksystemen.
Diplomarbeit
82 Seiten, 15 Abbildungen
Otto-von-Guericke-Universität Magdeburg,
1997.

Vorwort

Die vorliegende Diplomarbeit behandelt ein wichtiges Problem aus dem Bereich der Multidatenbanksysteme: die Identifikation semantisch äquivalenter Datenbankobjekte, d.h. die Erkennung von Datenbankobjekten in unterschiedlichen Datenbanken, die sich auf dasselbe Realweltobjekt beziehen. Die Lösung dieses Problems ist für die semantisch korrekte Ausführung globaler Datenbankoperationen in einem Multidatenbanksystem von großer Bedeutung.

Im Rahmen dieser Arbeit werden verschiedene Konzepte zur Identifikation semantisch äquivalenter Datenbankobjekte vorgestellt und miteinander verglichen. Es wird ein Entscheidungsgraph entwickelt, der die Auswahl eines geeigneten Identifikationsverfahrens für konkrete Anwendungsfälle ermöglicht. Weiterhin wird untersucht, welche funktionalen Anforderungen an Multidatenbanksysteme gestellt werden müssen, damit eine Identifikation semantisch äquivalenter Objekte ermöglicht werden kann. Anhand des objektrelationalen Multidatenbanksystems UniSQL/M wird untersucht, wie sich ausgewählte Identifikationsverfahren realisieren lassen. Außerdem wird eine Beispielanwendung in UniSQL/M implementiert.

Alle Ausführungen im Rahmen dieser Arbeit werden mit Hilfe von Beispielen und Abbildungen veranschaulicht.

An dieser Stelle möchte ich Herrn Prof. Dr. Gunter Saake und Herrn Dipl.-Inf. Ingo Schmitt für die sehr gute Betreuung der Diplomarbeit danken.

Inhalt

1 Einleitung

In zahlreichen Unternehmen werden heute Datenbanksysteme eingesetzt. Sie bilden zumeist die Basis für die elektronische Speicherung und Verarbeitung von Daten. Einzelne Abteilungen verwenden häufig verschiedene Datenbanksysteme, die für die speziellen Aufgaben der jeweiligen Unternehmensbereiche besonders geeignet sind oder deren Einsatz historische Ursachen hat. In den unabhängig voneinander entwickelten Datenbanken wurden die für die jeweiligen Unternehmensbereiche relevanten Realweltausschnitte modelliert. Für die Verarbeitung der gespeicherten Datenbestände wurden im Laufe der Zeit unterschiedliche Applikationen entwickelt und eingesetzt.

Da die Informationen vielfach in den voneinander isolierten Datenbanken der einzelnen Unternehmensbereiche gespeichert sind, handelt es sich um verteilte, heterogene Datenbestände. Dies bedeutet, daß die Daten mit Hilfe unterschiedlicher Datenbankmanagementsysteme in unterschiedlichen Hardwareumgebungen verwaltet werden. Außerdem sind die Datenbestände örtlich verteilt und semantisch heterogen. Informationen über bestimmte Objekte der Realität sind darüber hinaus nur in bestimmten Bereichen eines Unternehmens verfügbar.

Immer häufiger wird der Wunsch geäußert, bestimmten Applikationen Informationen bereichsübergreifend zur Verfügung zu stellen. In diesem Zusammenhang spricht man von einem integrierten Zugriff auf die Informationen des Unternehmens. Durch den immer stärkeren Wettbewerbsdruck ist es beispielsweise notwendig, umfassende Analysen der Daten zu erstellen, auf deren Grundlage dann wichtige Entscheidungen getroffen werden. Außerdem lassen sich durch einen integrierten Datenzugriff aufwendige Datentransfers zwischen einzelnen Unternehmensbereichen vermeiden.

Eine Möglichkeit für die Realisierung eines integrierten Zugriffs auf die Daten eines Unternehmens bildet das Konzept der Föderation von Datenbanksystemen. Der wesentliche Vorteil dieser Lösung gegenüber alternativen Integrationsansätzen besteht darin, daß die Autonomie der beteiligten Systeme gewahrt bleibt. Dies hat zur Folge, daß bestehende Applikationen unverändert weitergenutzt werden können. Hierdurch lassen sich erhebliche Investitionen in neue Applikationen vermeiden. Der integrierte Zugriff wird zusätzlich zum bestehenden Datenbanksystem realisiert und nicht anstelle dessen.

Die Integration erfolgt auf zwei Ebenen: Schemaebene und Instanzebene. Mit der Integration auf der Schemaebene befaßt sich die Schemaintegration. Ihre Aufgaben umfassen die Analyse der lokalen Schemata, die integriert werden sollen, und die Beseitigung von Konflikten zwischen den lokalen Schemata mit dem Ziel der Schaffung eines integrierten Gesamtschemas. Mögliche Konflikte umfassen beispielsweise Unterschiede in der Benennung von Attributen, unterschiedliche Wertebereiche der Attribute und Unterschiede der Daten in Format und Struktur.

Die Integration auf der Instanzebene wird als Instanzintegration bezeichnet und stützt sich im allgemeinen auf die Ergebnisse einer abgeschlossenen Schemaintegration. Infolge der unabhängigen Entwicklung der Datenbanken der ein-

zelnen Unternehmensbereiche (sogenannte lokale Datenbanken) ist es durchaus möglich ist, daß Informationen über ein und dasselbe Realweltobjekt (zum Beispiel eine konkrete Person) in mehreren lokalen Datenbanken gleichzeitig vorhanden sind. Die Instanzintegration ist virtueller Natur und umfaßt die Erkennung von identischen Realweltobjekten in den Datenbeständen der einzelnen lokalen Datenbanken und die Lösung der hieraus entstehenden Probleme, wie zum Beispiel die Behandlung widersprüchlicher Angaben über ein bestimmtes Realweltobjekt (beispielsweise unterschiedliche Altersangaben zu einer Person in verschiedenen Datenbanken).

Im Rahmen dieser Diplomarbeit soll die Instanzintegration näher betrachtet werden. Schwerpunkt ist das Problem der Identifikation von semantisch äquivalenten Datenbankobjekten, d. h. die Erkennung von Datenbankobjekten, die dasselbe Realweltobjekt repräsentieren. Neben der Erkennung derartiger Objekte sollen aber auch damit eng verknüpfte Probleme untersucht werden. Zum Beispiel fällt in diesen Problembereich die Frage: Wie können Informationen darüber, welche Objekte semantisch äquivalent sind, im praktischen Betrieb eines Multidatenbanksystems effizient berechnet und verwaltet werden?

Ziel der Arbeit ist es, geeignete Verfahren auszuwählen oder zu entwickeln, die der Identifikation semantisch äquivalenter Objekte dienen. Hierzu müssen geeignete Kriterien für die Bewertung und den Vergleich der einzelnen Verfahren gefunden werden. Darüber hinaus sollen diese Verfahren auf ihre Realisierbarkeit in dem objektrelationalen Multidatenbanksystem UniSQL/M untersucht werden. Außerdem soll eine Beispielanwendung in UniSQL/M realisiert werden.

Der Aufbau der Diplomarbeit ist wie folgt: Nach einigen einleitenden Worten in Abschnitt 1, wird in Abschnitt 2 auf Multidatenbanksysteme eingegangen. Schwerpunkt der Betrachtungen bilden hierbei föderierte Multidatenbanksysteme.

In Abschnitt 3 wird das Problem der Identifikation semantisch äquivalenter Datenbankobjekte dargestellt und erläutert. Es wird dabei auch auf weitere Probleme eingegangen, die mit dem Problem der Identifikation äquivalenter Objekte eng verbunden sind.

In Abschnitt 4 werden Anforderungen dargelegt, denen Verfahren zur Identifikation semantisch äquivalenter Objekte gerecht werden sollten.

In Abschnitt 5 wird eine Reihe von Konzepten zur Identifikation semantisch äquivalenter Objekte vorgestellt. Ihre Funktionen werden anhand eines durchgängigen Beispiels (eingeführt in Abschnitt 3) verdeutlicht. Außerdem werden die vorgestellten Konzepte anhand von Bewertungskriterien, die in Abschnitt 4 eingeführt wurden, bewertet und miteinander verglichen.

Abschnitt 6 geht auf den Einsatz von Identifikationsverfahren in Multidatenbanksystemen ein. Es wird untersucht, welche Informationen für die Identifikation semantisch äquivalenter Objekte herangezogen werden können, wie ein geeignetes Identifikationsverfahren für einen konkreten Anwendungsfall ausgewählt werden kann und welche funktionalen Anforderungen an ein Multidatenbanksystem gestellt werden, um eine Identifikation semantisch äquivalenter Objekte zu ermöglichen.

In Abschnitt 7 wird auf das objektrelationale Multidatenbanksystem UniSQL/M eingegangen. Es soll untersucht werden, wie eine Identifikation und (virtuelle) Integration semantisch äquivalenter Objekte in UniSQL/M möglich ist.

In Abschnitt 8 wird eine UniSQL/M-Beispielanwendung mit zwei lokalen Datenbanken vorgestellt. Es wird gezeigt, welche Schritte notwendig sind, um die Daten der lokalen Systeme in eine globale UniSQL/M-Datenbank einzubinden und wie hierbei die semantisch äquivalenten Objekte der lokalen Systeme integriert werden können.

Eine Untersuchung der Realisierungsmöglichkeiten, der in Abschnitt 5 vorgestellten Identifikationskonzepte in UniSQL/M erfolgt in Abschnitt 9. Die Arbeit wird von einer Zusammenfassung in Abschnitt 10 beendet.

2 Multidatenbanksysteme

In diesem Abschnitt soll ein Überblick über das Gebiet der Multidatenbanksysteme (MDBSe) gegeben werden. Der Begriff Multidatenbanksystem wird in der Literatur für zum Teil sehr unterschiedliche Arten von Datenbanksystemen (DBSen) verwendet. Ziel dieses Abschnittes ist es, zu klären, was im Rahmen dieser Arbeit unter einem MDBS zu verstehen ist. Es werden zu diesem Zweck verschiedene Klassen von MDBSen unter Angabe ihrer Merkmale vorgestellt. Die Ausführungen stützen sich dabei auf [SL90].

DBSe lassen sich in zentralisierte und verteilte DBSe einteilen. Ein zentralisiertes DBS besteht aus einem Datenbankmanagementsystem (DBMS) und einer Datenbank, die von dem DBMS verwaltet wird. Sowohl das DBMS als auch die Datenbank befinden sich auf einem gemeinsamen Computersystem. Ein verteiltes DBS besteht aus einem verteilten DBMS und mehreren physischen Datenbanken, die von ihm verwaltet werden. Die Datenbanken können hierbei auf verschiedene Computersysteme verteilt sein.

Ein MDBS ist ein DBS, das die Ausführung von Datenbankoperationen auf mehreren DBSen ermöglicht. Diese DBSe werden als Komponentendatenbanksysteme (KDBSe) oder lokale DBSe bezeichnet. Das MDBS realisiert eine logische Integration der beteiligten lokalen Datenbanken. Im Gegensatz zu klassischen verteilten DBSen, die bereits beim Entwurf einer Datenbank die physische Verteilung einer logisch zusammenhängenden Datenbank anstreben, werden bei einem MDBS bestehende isolierte und örtlich verteilte Datenbanken integriert.

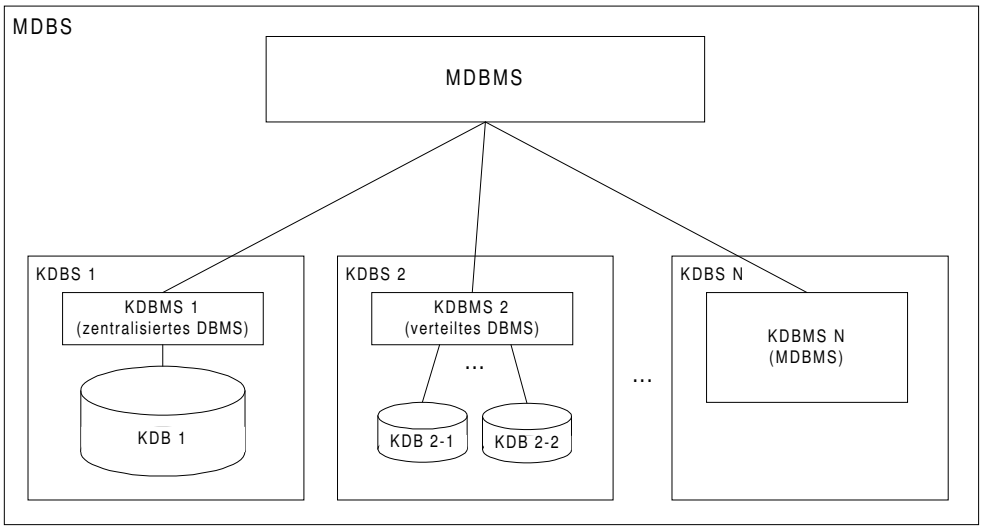


Abbildung 2-1: Ein MDBS mit seinen Komponenten

Ein KDBS kann wiederum ein zentralisiertes oder verteiltes DBS sein und besteht somit aus einem DBMS und einer oder mehreren Datenbanken. Folglich ist es

weiterhin möglich, daß ein MDBS als KDBS auftritt. MDBSe sind homogen, wenn die DBMSe aller KDBSe gleich sind, anderenfalls sind sie heterogen. Primäres Kriterium für die Gleichheit ist dabei das verwendete Datenmodell. Abbildung 2-1 veranschaulicht die vorausgegangenen Ausführungen.

Nachfolgend wird eine Taxonomie für MDBSe vorgestellt. Diese Taxonomie stellt eine Einteilung von MDBSen in Systemklassen entsprechend dem Grad der Autonomie und dem Grad der Integration der KDBSe dar. Eine graphische Darstellung der Taxonomie wird in Abbildung 2-2 gegeben.

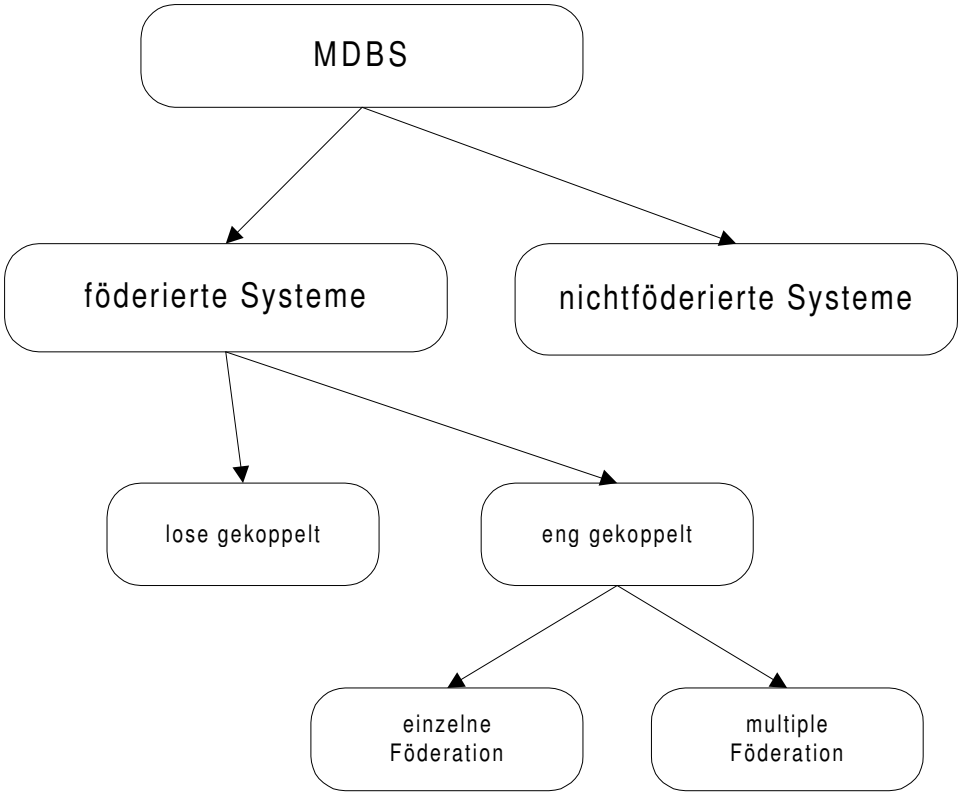


Abbildung 2-2: Taxonomie für MDBSe gemäß (SL90)

MDBSe werden hier zunächst in nichtföderierte und föderierte DBSe eingeteilt. Ein nichtföderiertes MDBS ist gekennzeichnet durch eine Integration von DBSen, die keine Autonomie besitzen. Es existiert hierbei eine zentrale Steuerung. Anders als bei föderierten Systemen unterscheidet ein nichtföderiertes System keine lokalen und globalen Nutzer bzw. Zugriffe. Alle Anfragen beziehungsweise Zugriffe richten sich an die zentrale Steuerung der globalen Ebene. Ein föderiertes Datenbanksystem (FDBS) hingegen besteht aus einer Menge von autonomen KDBSen, die einen Teil ihrer Daten der Föderation zur Verfügung stellen. Es existiert keine zentrale Steuerung, da die KDBSe den Zugriff auf ihre Daten kontrollieren. Al-

lerdings sollten Mechanismen zur Verwaltung globaler Transaktionen unterstützt werden, die in gewisser Weise auch als Steuerung betrachtet werden können.

Im nächsten Unterabschnitt werden föderierte Systeme näher betrachtet. Dort werden die weiteren Klassen von MDBSen erläutert, die in der Taxonomie genannt werden. Es handelt sich hierbei um die Klassen der lose und eng gekoppelten FDBSe sowie um FDBSe mit einfacher und multipler Föderation.

2.1 Föderierte Datenbanksysteme

Ein föderiertes Datenbanksystem (FDBS) ist laut [SL90] eine Sammlung kooperierender, jedoch autonomer KDBSe. Die KDBSe können hierbei heterogen sein. Der Kerngedanke der Föderation besteht in der Zusammenarbeit von unabhängigen Systemen. Die Steuerung der globalen Zugriffe auf Daten in den KDBSen erfolgt durch Software, die als föderiertes Datenbankmanagementsystem (FDBMS) bezeichnet wird. Das FDBMS verwaltet die Zuordnungen der lokalen (Schema-)Objekte zu globalen (Schema-)Objekten und die globalen Transaktionen. Die einzelnen KDBSe können zentralisierte DBSe, verteilte DBSe oder wiederum FDBSe sein. Es ist auch möglich, daß ein KDBS an mehreren Föderationen teilnimmt.

2.1.1 Heterogenität der KDBSe

Bei den KDBSen kann es sich um homogene (gleichartige) oder heterogene (verschiedenartige) Systeme handeln. Die Heterogenität der KDBSe äußert sich in der Art des verwendeten Datenmodells (zum Beispiel Relationenmodell, Netzwerkmodell, objektorientierter Modellansatz etc.), in der Art der verwendeten Anfragesprache (zum Beispiel SQL, QUEL, QBE etc.), als semantische Heterogenität und in den Fähigkeiten bezüglich der Verarbeitung von Transaktionen. Ein großes Problem bei der Integration von Datenbanken ist deren semantische Heterogenität, die auftritt, wenn in verschiedenen KDBSen in Beziehung stehende Daten unterschiedlich interpretiert werden. Das Hauptproblem besteht im Erkennen des Vorhandenseins derartiger Unterschiede.

2.1.2 Autonomie der KDBSe

Wie oben bereits erwähnt ist die Autonomie ein wesentliches Merkmal der KDBSe in einem föderierten System. Die Autonomie eines KDBSs kann als lokale Autonomie bezeichnet werden. Sie läßt sich weiter unterteilen in Betriebsautonomie und Dienstautonomie. Zur Betriebsautonomie gehört, daß das KDBS unabhängig über Punkte wie die folgenden entscheidet:

- Auswahl der verwalteten Daten (abgebildeter Weltausschnitt)
- Darstellung (Datenmodell, Anfragesprache) und Benennung der Datenelemente
- Semantische Interpretation der Daten (durch lokalen DB-Designer festgelegt)
- Verwendete Integritätsbedingungen

- Funktionalität des Systems (Menge der unterstützten Operationen)
- Implementation (z.B. Struktur von Records und Files)
- Ausführung von Operationen (Ausführung oder Ablehnung, Ausführungsreihenfolge)

Aus Sicht der Teilnahme eines KDBSs an einer Föderation ist die Autonomie in der Ausführung von Operationen besonders interessant. Die Entscheidung über die Ausführung von externen Operationen (übermittelt durch das FDBMS der globalen Ebene) liegt beim KDBS. Hierdurch kann eine ungehinderte Ausführung von lokalen Applikationen gewährleistet werden. Außerdem entscheidet das KDBS über die Reihenfolge der Operationen und muß das FDBMS nicht über diese Reihenfolge informieren. In der Praxis ergeben sich hieraus schwerwiegende Probleme für die globale Transaktionsverwaltung. Ein sowohl aus globaler Sicht als auch aus Sicht der jeweiligen lokalen Systeme konfliktfreier Ablauf der Transaktionen, kann unter strikter Berücksichtigung der Autonomie eine Verletzung der Serialisierbarkeit bedeuten.

Der zweite Typ lokaler Autonomie ist die Dienstautonomie. Sie beinhaltet die Entscheidung über Punkte wie:

- Kommunikation mit anderen DBSen
- Grad der Teilung von Funktionalität und Daten mit anderen DBSen

Die oben genannten Punkte drücken aus, daß es einem KDBS, vertreten durch den verantwortlichen Administrator, freisteht darüber zu entscheiden, ob und in welchem Umfang Daten anderen Systemen zur Verfügung gestellt werden. Je nach Grad der Kopplung der Systeme (siehe unten) werden derartige Entscheidungen aber auch in Zusammenarbeit mit anderen Administratoren getroffen.

2.1.3 Grad der Integration

Föderierte Systeme lassen sich nach der Art der Integration der KDBSe in *eng gekoppelte und lose gekoppelte Systeme* einteilen. Bei lose gekoppelten KDBSen liegt die Verantwortung für die Realisierung der Föderation beim Nutzer, wobei mit Nutzer in diesem Zusammenhang der Anwendungsentwickler gemeint ist. Die föderierte (globale) Schicht stellt lediglich grundlegende Dienste für die Kommunikation mit den KDBSen, zum Beispiel durch die Bereitstellung einer Multi-Datenbank-Anfragesprache, zur Verfügung, auf deren Grundlage sich der globale Nutzer sein individuelles globales Schema, entsprechend dem jeweiligen Anwendungszweck, erstellt. Föderierte Systeme mit loser Kopplung werden auch als interoperabele DBSe bezeichnet. Eine Verteilungstransparenz wird hierbei nicht angestrebt. Aufgrund der Nutzerorientiertheit der Föderation können bei loser Kopplung vielfältige föderierte Datenbankschemata existieren. FDBSe mit loser Kopplung ermöglichen somit stets multiple Föderationen.

Bei eng gekoppelten Systemen tragen die Systemadministratoren der beteiligten Systeme die Verantwortung für die Schaffung und die Pflege der Föderation. Sie ermöglichen den Zugriff auf die KDBSe, indem sie ein föderiertes (globales) Datenbankschema zur Verfügung stellen. Wenn bei eng gekoppelten Systemen nur ein föderiertes Schema möglich ist, so spricht man von Systemen mit *einzelner Föderation*. Im Gegensatz dazu spricht man von Systemen mit *multipler Föderation*, falls mehrere föderierte Schemata möglich sind.

Föderierte Systeme mit enger Kopplung streben Verteilungstransparenz an und verhalten sich somit aus Sicht der globalen Anwender wie ein normales (zentralisiertes) DBS.

2.1.4 Schemaarchitektur von FDBSen

Ausgehend von der ANSI/SPARC Schemaarchitektur, bestehend aus den drei Schichten internes Schema, konzeptionelles Gesamtschema und einer Menge von externen Schemata, muß für föderierte Systeme eine Erweiterung dieser Architektur erfolgen, um die spezifischen Merkmale von föderierten Systemen abbilden zu können. In [SL90] wird eine Architektur, bestehend aus fünf Schichten vorgeschlagen. Als unterste Schicht ist eine Menge von lokalen Schemata vorhanden. Jedes lokale Schema verkörpert die konzeptionelle Gesamtsicht eines KDBSs, ausgedrückt im Datenmodell des jeweiligen KDBSs, welches sich durchaus vom Datenmodell des föderierten Systems unterscheiden kann. Die nächsthöhere Schicht besteht aus einer Menge von sogenannten Komponentenschemata. Sie werden aus den lokalen Schemata durch eine Übersetzung in das globale Datenmodell abgeleitet (Schematranslation). Als nächste Schicht ist eine Menge von Exportschemata vorhanden, die eine Untermenge des Gesamtdatenbestandes eines KDBSs definieren, die der Föderation zugänglich gemacht werden soll. An die Schicht der Exportschemata schließt sich die Schicht der föderierten Schemata an. Hierbei handelt es sich um eine Gesamtdarstellung des föderierten Datenbestandes, der durch die Integration der einzelnen Exportschemata bestimmt wird (Schemaintegration). Bei eng gekoppelten Systemen mit einzelner Föderation besteht diese Schicht nur aus einem einzigen föderierten Schema. Die fünfte Schicht vollendet die angesprochene Architektur. Wie auch in der ANSI/SPARC Architektur wird ein angepaßter Zugang zur Datenbank für bestimmte Nutzer durch eine Menge von externen Schemata realisiert. Abbildung 2-3 veranschaulicht die vorangegangenen Ausführungen.

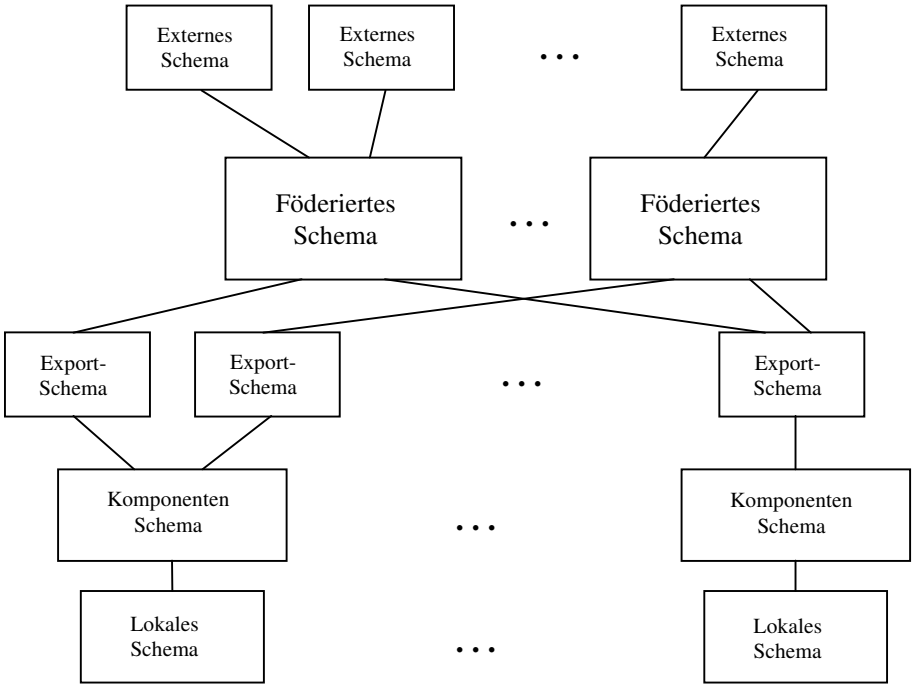


Abbildung 2-3: Fünf-Schichten-Schemaarchitektur eines FDBSs

2.2 Objektidentität in MDBSen

Will man auf in Datenbanken abgelegte Abbilder von Realweltobjekten, also Datenbankobjekte zugreifen, um beispielsweise Operationen auf ihnen auszuführen oder um Beziehungen zwischen verschiedenen Datenbankobjekten herzustellen, so ist es notwendig, die Datenbankobjekte voneinander zu unterscheiden. Die Eigenschaft eines Objektes, die es ermöglicht, es von allen anderen Objekten zu unterscheiden wird gemäß [KC86] Objektidentität genannt.

Die Identität von Realweltobjekten wird durch die Gesamtheit ihrer (unendlich vielen) Eigenschaften charakterisiert. Datenbankobjekte sind Abbilder von Realweltobjekten, die jedoch nur eine endliche Untermenge der Eigenschaften des betreffenden Realweltobjektes besitzen. Ihre Identität wird in einigen DBSen (ODBSe) durch ein spezielles Merkmal, den Objektidentifikator, realisiert. Andere Systeme (zum Beispiel relationale DBSe) unterstützen das Konzept der Objektidentität nur bedingt.

Da die Identität von Datenbankobjekten von besonderer Bedeutung für die Thematik dieser Arbeit ist, soll in diesem Unterabschnitt näher auf sie eingegangen werden. Zunächst werden allgemeine Konzepte zur Realisierung der Objektidentität vorgestellt. Daran anschließend soll dann erläutert werden, welche Probleme hinsichtlich der Identität von Datenbankobjekten in MDBSen auftreten.

2.2.1 Anforderungen an Objektidentifikatoren

Die Identität eines Datenbankobjektes sollte gemäß [SST97] unabhängig sowohl von den Eigenschaften des Datenbankobjektes als auch von seinem Verhalten sein.

Die Art und Weise, in der die Identität eines Objektes in einem DBS realisiert ist kann sehr unterschiedlich sein. In relationalen Systemen werden Datenbankobjekte (Tupel) anhand ihrer Primärschlüsselwerte unterschieden. Da der Schlüsselwert neben der Aufgabe der Identifikation auch noch eine Eigenschaft des Objektes darstellt (zum Beispiel der Name einer Person) ist die obige Forderung nach der zustandsunabhängigen Identifikation von Objekten verletzt. Hieraus folgt auch, daß es nicht möglich ist, Datenbankobjekte mit den gleichen Eigenschaften zu unterscheiden. In objektorientierten Systemen wird zur Unterscheidung von Datenbankobjekten das Konzept der Objektidentifikatoren (OIDen) verwendet. Ein OID ist ein systemvergebenes Merkmal eines Objektes, welches zur Identifikation eines Objektes dient. OIDen werden auch als Surrogate bezeichnet.

Damit OIDen ihre Aufgabe der Identifikation von Datenbankobjekten korrekt erfüllen können werden einige Anforderungen an die Vergabe von OIDen gestellt. Ein OID kann ein Datenbankobjekt nur dann identifizieren, wenn es eindeutig auf genau ein Objekt verweist. Man unterscheidet hierbei gemäß [SST97] die räumliche und die zeitliche Eindeutigkeit von OIDen.

Die räumliche Eindeutigkeit legt fest, in welchem räumlichen Bereich ein OID ein Objekt eindeutig identifiziert. Zum Beispiel kann die räumliche Eindeutigkeit innerhalb einer Datenbank oder auch für alle Datenbanken eines bestimmten räumlichen oder fachlichen Gebietes gewährleistet sein. Eine möglichst große räumliche Eindeutigkeit ist günstig, jedoch unter Umständen nur schwer zu garantieren. Die räumliche Eindeutigkeit von OIDen ist daher meist auf den Bereich einer Datenbank beschränkt.

Im Gegensatz zur räumlichen Eindeutigkeit legt die zeitliche Eindeutigkeit von OIDen den zeitlichen Bereich fest, in dem die Stabilität eines OIDs gewährleistet ist. Wiederum ist eine möglichst große zeitliche Eindeutigkeit wünschenswert. Typischerweise ist die zeitliche Eindeutigkeit eines OIDs für die Lebensdauer der betreffenden Datenbank garantiert. Dies bedeutet, daß der OID während der Objekterzeugung vergeben wird und sich während der Lebenszeit eines Objektes nicht verändert.

In [KAA+93] wird weiterhin gefordert, daß OIDen singular sind. Dies bedeutet, daß es nicht vorkommen kann, daß zwei verschiedene OIDen ein Datenbankobjekt identifizieren. Die Verwendung von Alias-OIDen würde lediglich zusätzlichen Verwaltungsaufwand verursachen.

Ein OID sollte außerdem systemvergeben und „unsichtbar“ sein. Nur wenn die OIDen durch das DBMS vergeben werden, kann ihre Eindeutigkeit und ihre Unabhängigkeit vom Zustand des Objektes garantiert werden. Da der Wert eines OIDs für den Nutzer nicht von Bedeutung ist, sollte er dem Nutzer verborgen bleiben. Er ist damit „unsichtbar“ und lediglich dem System bekannt. Jedoch müs-

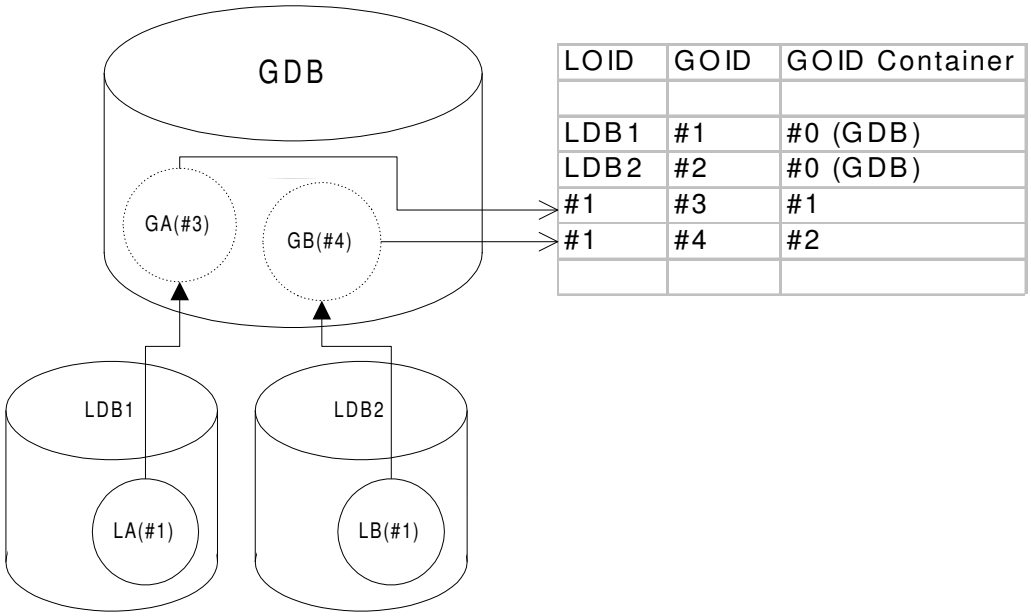
sen Operationen zur Arbeit mit OIDen zum Beispiel für den Vergleich von OIDen oder für das Erstellen von Referenzen angeboten werden.

Als zusätzliches Konzept zu dem der OIDen werden von einigen Systemen Objektnamen zur Identifikation ausgewählter Objekte angeboten. Objektnamen sind dem Nutzer bekannt und können für den Zugriff auf Objekte genutzt werden, die von besonderer Bedeutung für ein spezielles Anwendungsgebiet sind.

2.2.2 Konzepte der Objektidentifikation in MDBSen

In MDBSen existieren hinsichtlich der Identifikation von Datenbankobjekten einige besondere Probleme. Datenbankobjekte treten in einem MDBS in zwei Zusammenhängen auf. Einerseits als Objekte eines KDBSs. In diesem Fall werden sie als lokale Datenbankobjekte bezeichnet, die durch einen lokalen OID (LOID) identifiziert werden. LOIDen gelten für den Bereich eines KDBSs. Andererseits tritt (zumindest ein Teil) der Datenbankobjekte der KDBSe auch als Objekte der globalen (integrierten) Datenbank auf. Da die globalen Objekte aus verschiedenen Datenbanken stammen kann im allgemeinen keine Eindeutigkeit dieser OIDen auf der globalen Ebene gewährleistet werden. Durch eine zentral gesteuerte Vergabe der OIDen ließe sich dieses Problem lösen, jedoch läßt sich eine derartige Lösung bei FDBSen infolge der Autonomie der KDBSe nicht realisieren. Aus diesem Grund werden für die Datenbankobjekte der KDBSe zusätzlich globale OIDen (GOIDen) vergeben, die multidatenbankweit eindeutig sind. Die Beziehungen zwischen den LOIDen und den GOIDen müssen in Zuordnungs-Tabellen der globalen Ebene verwaltet werden. Eine ausführliche Abhandlung zum Thema der Objektidentifikation in MDBSen kann [SS95] entnommen werden. Abbildung 2-4 verdeutlicht die Zuordnung von OIDen.

Aus den beiden dargestellten lokalen Datenbanken LDB1 und LDB2 wird jeweils ein Objekt in die globale Datenbank (GDB) exportiert. Das lokale Objekt aus LDB1 ist LA und besitzt den LOID #1 während das lokale Objekt aus LDB2 LB heißt und ebenfalls den LOID #1 besitzt. Die LOIDen sind somit auf der globalen Ebene nicht eindeutig. Das lokale Objekt LA wird in der GDB zum globalen Objekt GA mit GOID #3 und das lokale Objekt LB wird zum globalen Objekt GB mit GOID #4. Die dargestellte Tabelle verwaltet die Zuordnung zwischen den LOIDen und den GOIDen. Zusätzlich wird der GOID des Objektes vermerkt, der das entsprechende Objekt enthält. Soll nun beispielsweise auf das globale Objekt GA mit GOID #3 zugegriffen werden, so läßt sich aus der Zuordnungstabelle folgern, daß dieses Objekt den LOID #1 besitzt und im globalen Objekt mit GOID #1 enthalten ist. Das Objekt mit GOID #1 ist laut Tabelle LDB1. Damit wurde das lokale Objekt LA erfolgreich lokalisiert. Die Vorgehensweise für Objekt GB ist analog.



LOID	GOID	GOID Container
LDB1	#1	#0 (GDB)
LDB2	#2	#0 (GDB)
#1	#3	#1
#1	#4	#2

Abbildung 2-4: Dynamische OID-Zuordnung

Im Fall von KDBSen, die das Konzept der Objektidentität mit den im letzten Abschnitt dargestellten Anforderungen nicht unterstützen (zum Beispiel relationale Systeme), kann die Vergabe von OIDen auf der Grundlage von Schlüsselattributen der betreffenden Objekt-Typen erfolgen. Es ist günstig, wenn es sich hierbei um unveränderliche Eigenschaften von Objekten handelt (zum Beispiel die Sozialversicherungsnummer einer Person).

3 Problembeschreibung

Eine Datenbank stellt ein Abbild eines bestimmten Bereichs der realen Welt dar. In einem Abstraktionsprozeß werden Objekte der Realwelt in eine Datenbank abgebildet, wobei die Eigenschaften der realen Objekte Berücksichtigung finden, die für den speziellen Anwendungszweck der Datenbank relevant sind, wogegen von für den Anwendungszweck unbedeutenden Eigenschaften abstrahiert wird. Dabei wird im allgemeinen von der Annahme ausgegangen, daß jedes Objekt der Realwelt maximal einmal als Datenbankobjekt in einer bestimmten Datenbank existiert.

Werden mehrere unabhängige Datenbanken in einem Multidatenbanksystem integriert, so ist es möglich, das sich die Extensionen der einzelnen lokalen Datenbanken überlappen. Dies bedeutet, daß bestimmte Objekte der Realwelt in mehr als einer lokalen Datenbank Repräsentationen besitzen. Datenbankobjekte, die sich auf dasselbe Realweltobjekt beziehen, werden in der Literatur als semantisch äquivalente Objekte, isomeric objects [CTK96], matching objects [ZHK+95] oder inter-database instances [WM89] bezeichnet. Zum Beispiel könnten Informationen über eine konkrete Person in einer Datenbank des Finanzamtes und ebenso in einer Datenbank des Einwohnermeldeamtes gespeichert sein. Würden diese zwei Datenbanken in einem Multidatenbanksystem zu einer globalen Datenbank integriert, so stellt sich die Frage, in welcher Weise die Person in der globalen Datenbank repräsentiert werden soll.

Eine Möglichkeit wäre die Existenz von zwei Datenbankobjekten auf der globalen Ebene, womit allerdings die oben erläuterte Annahme, daß jedes Realweltobjekt maximal eine Entsprechung in einer Datenbank besitzt, nicht mehr erfüllt wäre. Darüber hinaus würden weitere Probleme durch diese Verfahrensweise verursacht. Zum Beispiel würde die Berechnung der Anzahl der in der globalen Datenbank erfaßten Personen ein recht zweifelhaftes Ergebnis liefern, da die Personen, die in mehr als einer lokalen Datenbank vorhanden sind, mehrfach in das Ergebnis eingehen. Entsprechend würden diese Personen in den Ergebnissen von Anfragen an die globale Datenbank mehrfach auftreten, was sicherlich unerwünscht wäre.

Eine weitere Möglichkeit ist die Schaffung eines einzigen globalen Datenbankobjektes für alle lokalen Objekte, die sich auf ein und dasselbe Realweltobjekt beziehen. In diesem Fall ist die obige Annahme auch für die globale Datenbank erfüllt. Ein Vorteil dieser Lösung wäre, daß in diesem Fall die gesamten Informationen über ein, in der globalen Datenbank dargestelltes Realweltobjekt als Vereinigung der Informationen aus den lokalen Datenbanken über das betreffende Objekt auf der globalen Ebene verfügbar sind.

Abbildung 3-1 verdeutlicht die Situation bei der Modellierung eines Realweltobjektes und der Integration der entstandenen lokalen Datenbankobjekte zu einem globalen Objekt: Ein Realweltobjekt in Gestalt eines Automobils, welches durch eine unendliche Menge von Eigenschaften EIG_AUTO charakterisiert werden kann, soll in zwei Datenbanken abgebildet werden. Die entsprechenden Da-

tenbankobjekte O1 und O2, die das betreffende Auto in den beiden Datenbanken repräsentieren, enthalten jeweils eine endliche Untermenge der Merkmale des realen Autos. Sollen nun die Datenbanken DB1 und DB2, die jeweils ein Bild des realen Autos enthalten, in eine globale Datenbank GDB integriert werden, so muß durch die Integration ein integriertes Datenbankobjekt O1O2 erzeugt werden, welches das Auto in der globalen Datenbank repräsentiert. Die Merkmale des globalen Objektes entsprechen einer (möglicherweise unechten) Untermenge der Vereinigung der Merkmalsmengen der beiden lokalen Objekte.

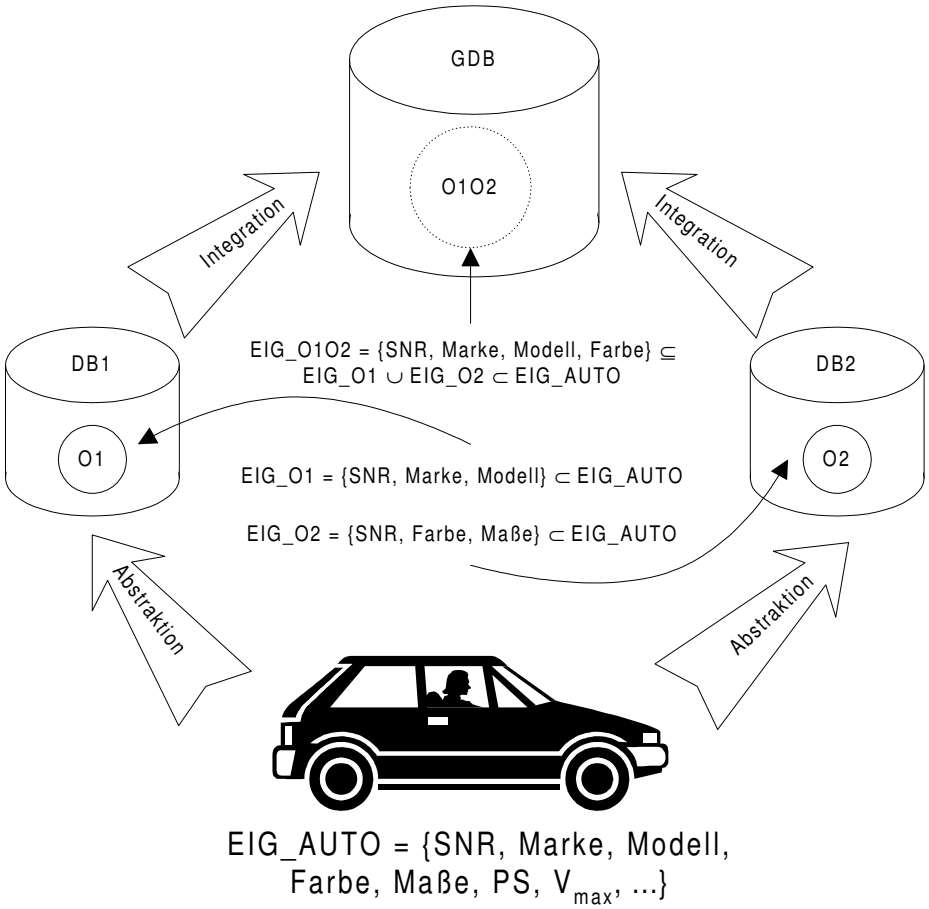


Abbildung 3-1: Datenbankmodellierung von Realweltobjekten

Jedoch verursacht die Forderung nach der Schaffung eines integrierten globalen Objektes auch einige Probleme. Eine Voraussetzung für die (virtuelle) Integration der lokalen Objekte ist, daß die Datenbankobjekte, die sich auf dasselbe Realweltobjekt beziehen, erkannt werden müssen. In diesem Zusammenhang spricht man von der Identifikation semantisch äquivalenter Datenbankobjekte in Multidatenbanksystemen.

Das Problem besteht somit darin, von zwei (oder mehr) Datenbankobjekten unterschiedlicher Datenbanken zu entscheiden, ob sie sich auf dasselbe Realweltobjekt beziehen oder nicht.

Die Untersuchung dieses Problems steht im Mittelpunkt dieser Arbeit. Die Lösung dieses Problems ist von großer Bedeutung für die Realisierung und den Betrieb eines Multidatenbanksystems. Sie ist die Voraussetzung für die intensive und effektive Nutzung einer Multidatenbank, da sie die semantisch korrekte Ausführung von bestimmten Datenbankoperationen (zum Beispiel Joins) oder die Erkennung von Redundanzen im Multidatenbanksystem erst ermöglicht.

Ein Datenbankdesigner, der mit dem Entwurf einer globalen Datenbank als Integration einer Menge von lokalen Datenbanken beauftragt wird, kann allerdings im allgemeinen nicht wissen, welche lokalen Datenbankobjekte Repräsentationen eines gemeinsamen Realweltobjektes sind. Da die Datenbestände darüber hinaus meist sehr umfangreich sind, ist es notwendig, den Vorgang der Erkennung semantisch äquivalenter Objekte (weitgehend) zu automatisieren.

Allerdings handelt es sich hier um ein Problem, das für den allgemeinen Fall algorithmisch unentscheidbar ist. Da ein Realweltobjekt durch eine unendliche Menge von Eigenschaften charakterisiert und identifiziert wird, aber eine Datenbank immer nur einen Teil der Eigenschaften eines Realweltobjektes berücksichtigen kann, ist eine sichere Identifikation anhand der Merkmale eines Datenbankobjektes nicht möglich. Das heißt, daß ein Verfahren zur Erkennung von semantisch äquivalenten Objekten keine sichere Entscheidung treffen kann, ob zwei Objekte äquivalent sind oder nicht. Vielmehr liefert ein derartiges Verfahren möglicherweise beziehungsweise wahrscheinlich äquivalente Objekte.

Eine wichtige Frage, die es zu beantworten gilt, lautet: Welche funktionalen Anforderungen muß ein Multidatenbanksystem erfüllen, um die Erkennung und Verwaltung von semantisch äquivalenten Datenbankobjekten zu ermöglichen. Weiterhin muß geklärt werden, wie im Fall von inkonsistenten Daten (semantisch äquivalente Datenbankobjekte besitzen unterschiedliche Werte für gemeinsame Eigenschaften) zu verfahren ist und welche Voraussetzungen hinsichtlich der Identifikation von Datenbankobjekten notwendig sind.

Durchgängiges Beispiel

Die verschiedenen Konzepte zur Erkennung von semantisch äquivalenten Objekten in MDBSen sollen mit Hilfe eines durchgängigen Beispiels erläutert werden. Es wird hierbei davon ausgegangen, daß in zwei Datenbanken Informationen über Kaufhäuser in Deutschland gespeichert sind. Die eine Datenbank (DB1) wird von einer Verbraucherorganisation verwendet während die andere Datenbank (DB2) von der Industrie- und Handelskammer (IHK) genutzt wird. In DB1 sind die betreffenden Informationen in einem Objekt-Typ A und in DB2 in einem Objekt-Typ B modelliert. Die Intensionen und ein Teil der Extensionen dieser Objekt-Typen sind aus den folgenden Tabellen ersichtlich.

OID	SS#	Name	Ort	Straße	Branche	Start-Datum
	number(5)	string	string	string	string	date
A1	01863	City-Center	HBS	Domplatz	Food	31.05.93
A2	38541	City-Center	HBS	Domplatz	Elektronik	02.06.93
A3	51780	City-Center	MD	Fischmarkt	Food	03.08.94

Tabelle 3-1: Objekt-Typ A (Kaufhäuser) in Verbraucher-Datenbank (DB1)

OID	IHKS#	Name	PLZ	Straße	Topseller	Start-Datum
	number(5)	string	number(5)	string	string	date
B1	01863	City-Center	38820	Domplatz	Pizza	31.05.93

Tabelle 3-2: Objekt-Typ B (Kaufhäuser) in IHK-Datenbank (DB2)

Der Primärschlüssel in A ist die Nummer der Service-Station (SS#). Ein weiterer Schlüsselkandidat für A sei die Kombination aus Name, Ort und Branche eines Kaufhauses. In B ist der Primärschlüssel die IHK-Site-Nummer (IHKS#). Ein weiterer Schlüsselkandidat für B sei der Name eines Kaufhauses.

Für das Bundeswirtschaftsministerium soll nun im Rahmen einer Wirtschaftsstudie die Anzahl der Kaufhäuser in Deutschland festgestellt werden. Grundlage dieser Studie sind die Datenbanken DB1 und DB2. Wenn in dieser Studie korrekte Ergebnisse erzielt werden sollen, so ist es notwendig, daß die Kaufhäuser ermittelt werden, die in beiden Datenbanken vorhanden sind, da anderenfalls diese Kaufhäuser mehrfach in das Ergebnis eingehen würden.

4 Anforderungen an Identifikationsverfahren

In diesem Abschnitt sollen die Anforderungen an Verfahren zur Erkennung von semantisch äquivalenten Datenbankobjekten festgelegt und erläutert werden. Hierzu werden in den weiteren Unterabschnitten die folgenden Anforderungen näher betrachtet:

- Identifikation
- Korrektheit (einschließlich Monotonie)
- Vollständigkeit
- Behandlung von inkonsistenten Attributwerten
- Effizienz

Diese Anforderungen werden dann die Meßlatte für die Bewertung verschiedener Verfahren im nächsten Abschnitt sein.

Identifikation (Mächtigkeit der match-Bedingungen)

Ein Verfahren zur Lösung des vorgestellten Problems muß zunächst einmal den primären Zweck des Verfahrens, namentlich die Erkennung von äquivalenten Objekten erfüllen. Grundlage der Identifikation äquivalenter Objekte sind bestimmte Kriterien beziehungsweise Bedingungen, die erfüllt sein müssen damit zwei Objekte als semantisch äquivalent gelten. Derartige Bedingungen werden auch als match-Bedingungen bezeichnet. In welcher Form sie auftreten kann sehr unterschiedlich sein. Zum Beispiel könnte eine Regel spezifizieren, wann zwei Objekte äquivalent sind. Aber es ist auch möglich, daß ein Mensch, beispielsweise in Gestalt eines Datenbankadministrators, mit Hilfe seines Wissens über die betrachteten Objekte die Entscheidung über die Äquivalenz der Objekte trifft.

Vorteilhaft sind Verfahren, die es ermöglichen, komplexe Kriterien für die Äquivalenz von Objekten zu formulieren. Ein Verfahren ist um so flexibler und breiter einsetzbar, je mächtiger die Möglichkeiten zur Formulierung von Bedingungen für die Äquivalenz von Objekten sind.

Korrektheit

Die Identifikation von semantisch äquivalenten Datenbankobjekten ist, wie in Abschnitt 3 ausgeführt, ein für den allgemeinen Fall unentscheidbares Problem.

Für die sinnvolle Nutzung einer Multidatenbank ist aber ein Verfahren Voraussetzung, welches beim Einsatz in einem MDBS korrekte Ergebnisse liefert, d.h. für zwei als semantisch äquivalent erkannte Datenbankobjekte ist diese Entscheidung auch richtig. Andererseits muß die Entscheidung für zwei als nicht äquivalent erkannte Objekte ebenfalls korrekt sein.

Eine Verletzung der Korrektheitseigenschaft hätte zur Folge, daß Objekte als semantisch äquivalent betrachtet werden, die diese Eigenschaft nicht besitzen. Die Nutzung der betreffenden Daten würde zu falschen Verarbeitungsergebnissen füh-

ren. Damit wäre die Verwendbarkeit der Daten stark eingeschränkt oder sogar unmöglich. Analoges gilt für den Fall der inkorrekt festgestellten Nicht-Äquivalenz.

Will man überprüfen, ob die Ergebnisse der Untersuchung einer Menge von Objektpaaren auf semantische Äquivalenz korrekt sind, so können folgende Überlegungen genutzt werden, um vorhandene Fehler zu erkennen: Man kann sich leicht überlegen, daß es sich bei der semantischen Äquivalenz um eine Äquivalenzrelation im mathematischen Sinn auf der Menge der untersuchten Objektpaare handelt. Sie besitzt somit die Eigenschaften Reflexivität, Symmetrie und Transitivität. Geht man von der eingangs dargelegten Grundannahme aus, daß jedes Realweltobjekt maximal einmal in einer Datenbank abgebildet wird, so ist es nicht möglich, daß ein Objekt einer bestimmten Datenbank mit zwei (oder mehr) Objekten einer anderen Datenbank semantisch äquivalent ist. Der Grund hierfür ist, daß wegen der Transitivität der semantischen Äquivalenz folgen würde, daß die zwei (oder mehr) Objekte der zweiten Datenbank, die mit dem Objekt der ersten Datenbank semantisch äquivalent sind, untereinander ebenfalls semantisch äquivalent sind (siehe Abbildung 4-1). Dies steht aber im Widerspruch zur dargestellten Grundannahme, daß jedes Realweltobjekt nur einmal in einer Datenbank abgebildet wird. Tritt im Ergebnis einer Untersuchung der soeben beschriebene Fall auf, so liegt ein Fehler bei der Identifikation semantisch äquivalenter Objekte vor.

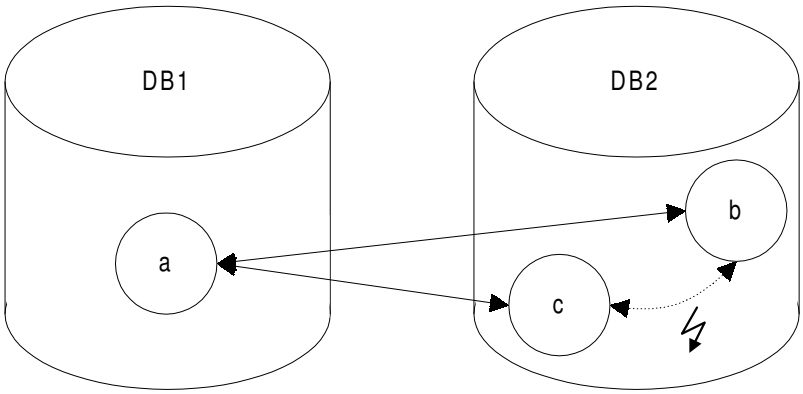


Abbildung 4-1: Erkennung inkorrekter Identifikationsergebnisse

Bei der Identifikation semantisch äquivalenter Objekte wird von der Grundannahme ausgegangen, daß Objekte, die einmal (korrekt) als semantisch äquivalent erkannt wurden, dieses Äquivalenzverhältnis stets bewahren. Dies gilt in analoger Weise für als nicht äquivalent erkannte Objekte. Selbst, wenn sich Attributwerte, die zu einer Erkennung als semantisch äquivalent beziehungsweise semantisch nicht äquivalent geführt haben, geändert werden, bleibt das Äquivalenzverhältnis erhalten. In [ZHK+95] wird dies als Nutzung historischer Informationen bezeichnet. Die dargelegte Grundannahme widerspiegelt sich in der Eigenschaft der Monotonie eines Identifikationsverfahrens.

Die Eigenschaft der *Monotonie eines Verfahrens* besagt, daß eine Entscheidung hinsichtlich einer bestehenden beziehungsweise nicht bestehenden semantischen Äquivalenz von Datenbankobjekten nicht revidiert wird, wenn zusätzliche oder neue Informationen über diese Datenbankobjekte verfügbar werden. Für die Korrektheit eines Verfahrens ist die Eigenschaft der Monotonie erforderlich.

Vollständigkeit

In engem Zusammenhang mit dem Kriterium der Korrektheit steht das Merkmal der Vollständigkeit eines Verfahrens. Es ist zu klären, ob ein Verfahren für alle Objektpaare eine Entscheidung darüber treffen kann, ob die betreffenden Objekte äquivalent sind oder nicht. Kann eine Entscheidung nicht für alle Objektpaare getroffen werden, so muß eine dreiwertige Logik hinsichtlich der Korrektheit des Verfahrens verwendet werden. Die möglichen Ergebnisse der Untersuchung auf semantische Äquivalenz sind in diesem Fall: ja, nein und unbekannt.

Nach [LSP+93] wird die Menge aller zu untersuchenden Objektpaare in drei Klassen vollständig zerlegt (siehe Abbildung 4-2). Es handelt sich um die Klassen der äquivalenten Objektpaare, der nicht äquivalenten Objektpaare und der Objektpaare mit unbekanntem Äquivalenzverhältnis. Für ein Verfahren, welches für jedes untersuchte Objektpaar eine Entscheidung bezüglich der Äquivalenz trifft und somit das Merkmal der Vollständigkeit erfüllt, ist die Menge der Objektpaare mit unbekanntem Äquivalenzverhältnis leer.

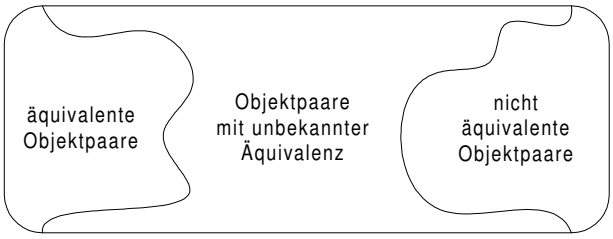


Abbildung 4-2: Zerlegung der untersuchten Objektpaare

Häufig bedeutet bessere Korrektheit geringere Vollständigkeit, wenn für bestimmte Eingaben keine Entscheidung getroffen werden kann. Eine bessere Vollständigkeit bedingt unter Umständen jedoch eine schlechtere Korrektheit, wenn für bestimmte Objektpaare auch falsche Ergebnisse akzeptiert werden. Als Kompromiß erscheint die Möglichkeit des „manuellen“ Eingriffs des Menschen. Ein Verfahren sollte also auch die Möglichkeit bieten, daß der Mensch selbst in die Entscheidungsfindung eingreifen kann. Das Verfahren kann ihm dann als Ratgeber mit Vorschlägen behilflich sein.

Behandlung von inkonsistenten Attributwerten

Ein Verfahren zur Erkennung von semantisch äquivalenten Objekten sollte neben der Erkennung selbst auch die Integration der lokalen Objekte zu einem globalen Objekt unterstützen. Problematisch ist dies insbesondere, wenn in den lokalen

Datenbanken unterschiedliche Informationen bezüglich einer Eigenschaft eines Objektes auftreten, d.h. es bestehen Inkonsistenzen zwischen den lokalen Datenbanken. Das Verfahren sollte Strategien zur Lösung von derartigen Konflikten anbieten.

Effizienz

Für die Nutzung eines MDBSs ist die Frage der Effizienz bei der Erkennung von äquivalenten Objekten von großer Bedeutung. Insbesondere bei der interaktiven Nutzung eines DBSs ist das Antwortzeitverhalten zu berücksichtigen. Damit der Aufwand zur Erkennung äquivalenter Objekte nicht bei jeder Anfrage neu entsteht, ist die persistente Verwaltung von Informationen über als äquivalent erkannte Objekte anzustreben. In diesem Zusammenhang spricht man von der Materialisierung der entsprechenden Informationen. Häufig werden hierzu die OIDen der semantisch äquivalenten Objekte paarweise abgespeichert.

Es entstehen durch diese Verfahrensweise natürlich neue Probleme hinsichtlich der Aktualität dieser Informationen. Zum Beispiel müssen lokale Insert- und Delete-Operationen berücksichtigt werden.

5 Ansätze für die Lösung des Problems

Ziel dieses Kapitels ist es, Vorschläge und Anregungen für die Gestaltung von Verfahren zur Identifikation von semantisch äquivalenten Datenbankobjekten in MDBSen zu liefern. Zunächst wird auf Konzepte eingegangen, auf deren Grundlage die Erkennung von äquivalenten Objekten erfolgen kann. Wegen ihrer großen Bedeutung für die effiziente Nutzung der Ergebnisse des Identifikationsprozesses werden daran anschließend Konzepte zur Verwaltung von Informationen über äquivalente Objekte erläutert. Ausführungen zur Behandlung von Problemen, die durch unvollständige und inkonsistente Datenbestände in den beteiligten lokalen Datenbanken entstehen, schließen das Kapitel ab.

5.1 Konzepte zur Erkennung äquivalenter Datenbankobjekte

Die Erkennung von semantisch äquivalenten Datenbankobjekten stützt sich in jedem Fall primär auf die Eigenschaften der modellierten Realweltobjekte, die zu einem für den jeweiligen Anwendungszweck der betrachteten Datenbank relevanten Teil abgebildet wurden und sich in den Attributen eines Datenbankobjektes widerspiegeln. Für die Nutzung dieser Informationen zur Feststellung der semantischen Äquivalenz von Datenbankobjekten existieren vielfältige Möglichkeiten, angefangen von der Nutzung eines gemeinsamen Primärschlüssels bis hin zur Einbeziehung von Zusatzwissen über die betrachteten Objekte. Jedoch lassen sich für einen konkreten Fall nur speziell für diesen Fall angepaßte Lösungen verwenden.

Bei den folgenden Ansätzen lassen sich zwei Klassen hinsichtlich der Art der Entscheidung über die Äquivalenz von Objekten unterscheiden. Die Verfahren der ersten dieser beiden Klassen liefern exakte Ergebnisse über die Äquivalenz der Datenbankobjekte. Als Ausgabe für die Überprüfung eines Objektpaares auf semantische Äquivalenz gibt es hier nur drei Möglichkeiten: ja, nein, unbekannt. Die zweite Klasse liefert als Ergebnis einen Wahrscheinlichkeitswert, der angibt, wie wahrscheinlich es ist, daß die beiden überprüften Objekte semantisch äquivalent sind. Bei der ersten Klasse handelt es sich genaugenommen um einen Spezialfall der zweiten Klasse, da sich ein Wahrscheinlichkeitswert unter Einbeziehung geeigneter Grenzwahrscheinlichkeiten immer in einen der drei Werte ja, nein, unbekannt „umrechnen“ läßt. Zum Beispiel könnte definiert sein, daß Wahrscheinlichkeitswerte zwischen 1 und 0.9 einem Ja entsprechen, Wahrscheinlichkeitswerte zwischen 0.9 und 0.3 ein unbekanntes Äquivalenzverhältnis anzeigen und Wahrscheinlichkeitswerte zwischen 0.3 und 0 für ein Nein stehen.

Von den in den folgenden Unterabschnitten dargestellten Konzepten liefern alle mit Ausnahme des letzten (kompatible Attributpaare) exakte Ergebnisse.

5.1.1 Verwendung einer Lookup-Tabelle

Bei Verwendung einer Lookup-Tabelle entscheidet der Mensch, welche Datenbankobjekte semantisch äquivalent sind. Hierzu werden die äquivalenten Objekte

gemäß [ZHK+95] mit Hilfe einer Tabelle (sogenannte Lookup-Tabelle) einander zugeordnet. Die Tabelle enthält dabei Paare von OIDen oder Schlüsselwerten der äquivalenten Objekte. Alle Objektpaare, deren OIDen nicht in der Lookup-Tabelle enthalten sind, werden als semantisch nicht äquivalent betrachtet.

Die Kriterien, auf deren Grundlage die Entscheidung über die Äquivalenz oder die Nicht-Äquivalenz von Objekten getroffen wird, sind im Wissen des Menschen (zum Beispiel Datenbankadministrator) enthalten, welches er über die betrachteten Objekte besitzt. In Abhängigkeit davon, wie umfassend das Wissen des betreffenden Menschen ist, sind vollständige und korrekte Ergebnisse möglich.

Der große Nachteil dieses Konzeptes ist, daß die Entscheidung über die semantische Äquivalenz der zu untersuchenden Objekte „manuell“ durch den Menschen getroffen werden muß und nicht automatisch durch das System. Insbesondere bei großen Datenbeständen ergeben sich hieraus große Probleme, da ein erheblicher Aufwand bei der Identifikation der äquivalenten Objekte entsteht. Außerdem ist notwendig, die Lookup-Tabelle nach Insert- und Delete-Operationen zu aktualisieren. Lediglich bei relativ unveränderlichen Datenbeständen tritt dieses Problem in den Hintergrund.

In unserem laufenden Beispiel würden die Objekte A1 und B1 durch folgende Lookup-Tabelle als semantisch äquivalent identifiziert:

OID (DB1)	OID (DB2)
A1	B1

Tabelle 5-1: Beispiel-Lookup-Tabelle

5.1.2 Gemeinsamer Primärschlüssel (common key)

Die Anwendung des Konzeptes der Verwendung eines gemeinsamen Primärschlüssels setzt das Vorhandensein eines solchen (common key) voraus. Man spricht in diesem Fall davon, daß die betreffenden Schlüsselattribute der Entity-Typen in den betrachteten Datenbanken kompatibel sein müssen, d.h. daß sie semantisch und den Wertebereich betreffend äquivalent sind. Weiterhin wird vorausgesetzt, daß dieser Primärschlüssel auch für den integrierten (globalen) Objekt-Typ die Schlüsseleigenschaft besitzt.

Anhand des gemeinsamen Schlüssels lassen sich in derartigen Fällen die äquivalenten Datenbankobjekte leicht bestimmen. Allerdings kann in der Praxis nur selten auf dieses Konzept zurückgegriffen werden, da die verwendeten Schlüssel in unabhängig voneinander entwickelten Datenbanken meist nicht kompatibel sind. Im allgemeinen treten identische Schlüssel nur auf, wenn die betrachteten Datenbanken in einem Zusammenhang stehen. Ein solcher Fall wäre zum Beispiel die Verwaltung von Produkten in verschiedenen Abteilungen eines Unternehmens, wobei die unternehmensweit eindeutige Produktnummer als gemeinsamer Schlüssel dient. Für die Fälle, in denen ein gemeinsamer Schlüssel vorhanden ist, stellt dieses Konzept auch gleichzeitig die beste Lösung dar, da die Be-

rechnung der äquivalenten Objekte effizient über einen Verbund erfolgen kann und das Verfahren vollständige und korrekte Ergebnisse liefert.

Als Variante für Fälle, in denen der Schlüsselvergleich aufgrund der Größe der Schlüsselwerte sehr aufwendig wäre wird in [Pu91] vorgeschlagen, nur Teile der Schlüsselwerte zum Vergleich heranzuziehen. Zum Beispiel könnte der Vergleich einer 250 Zeichen großen Zeichenkette auf die ersten 30 Zeichen beschränkt werden. Obwohl der Vergleich in diesem Fall effizienter ausgeführt werden kann besteht die Gefahr, daß auch fehlerhafte Ergebnisse auftreten.

Für unser Beispiel kann bei Verwendung der SS# von Objekt-Typ A und der IHKS# von Objekt-Typ B als gemeinsamen Schlüssel durch einfaches Vergleichen der Schlüsselwerte festgestellt werden, ob zwei Objekte semantisch äquivalent sind. Dies setzt jedoch voraus, daß diese Schlüssel auch kompatibel sind, d.h. daß sie sowohl semantisch als auch in ihrem Wertebereich übereinstimmen. Man kann unter dieser Voraussetzung vermuten, daß im Beispiel das City-Center mit der SS#=01863 aus A semantisch äquivalent ist mit dem City-Center mit der IHKS#=01863 aus B.

5.1.3 Erweiterter Schlüssel (extended key)

Für Fälle, in denen kein gemeinsamer Schlüssel existiert, läßt sich unter Umständen das in [LSP+93] vorgestellte Konzept des erweiterten Schlüssels (extended key) verwenden. Welche Voraussetzungen erfüllt sein müssen damit das Konzept des erweiterten Schlüssels verwendet werden kann und wie die Identifikation semantisch äquivalenter Datenbankobjekte unter Nutzung dieses Konzeptes erfolgt, soll in diesem Unterabschnitt erläutert werden.

Für eine gegebene Menge von Realweltobjekten E seien R und S zwei Objekt-Typen, die Objekte aus E in unterschiedlichen Datenbanken modellieren. Die Primärschlüssel für R und S sind als Attributmengen K_R und K_S dargestellt. Der erweiterte Schlüssel K_{Ext} ist definiert als die minimale Menge von Attributen der Form $K_R \cup K_S \cup ATT$, die benötigt wird, um eine Instanz des integrierten (globalen) Objekttyps zu identifizieren, der aus der Integration der Objekt-Typen R und S hervorgegangen ist. ATT ist hierbei eine Menge von Attributen der Realweltobjekte aus E, die im integrierten Objekt-Typ, der aus R und S hervorgegangen ist, enthalten sind. Allerdings enthält ATT keine Attribute aus K_R und K_S. Der erweiterte Schlüssel K_{Ext} dient als Primärschlüssel des integrierten (globalen) Objekt-Typs.

Das Konzept des erweiterten Schlüssels schließt auch den Fall ein, daß ein gemeinsamer Schlüssel existiert. In diesem Spezialfall ist der erweiterte Schlüssel gleich dem Schlüssel von R und dem Schlüssel von S, es gilt $K_{Ext}=K_R=K_S$.

In vielen Fällen gilt $K_{Ext}=K_R \cup K_S$. Dies bedeutet, daß die Vereinigung der Schlüssel von R und S ausreichend ist, um eine Instanz des integrierten Objekt-Typs zu identifizieren. Die Menge ATT ist in diesem Fall leer.

Für einen gegebenen erweiterten Schlüssel K_{Ext} existiert eine Identitätsregel (identity rule), die angibt, wann zwei Objekte semantisch äquivalent sind. Sie hat die Form:

$$\forall e_1,e_2 \in E, (e_1.A_1=e_2.A_1) \wedge ... \wedge (e_1.A_k=e_2.A_k) \rightarrow (e_1 \equiv e_2), K_{Ext}=\{A_1,A_2,...,A_k\}$$

Dies bedeutet, daß zwei Objekte semantisch äquivalent sind, wenn sie in ihren korrespondierenden Attributen des erweiterten Schlüssels übereinstimmen. Häufig sind jedoch nicht alle benötigten Attributwerte des erweiterten Schlüssels direkt verfügbar. In diesem Fall müssen die fehlenden Werte mit Hilfe von Zusatzwissen „berechnet“ werden (siehe unten).

Für unser Beispiel sei angenommen, daß die Primärschlüssel der entsprechenden Objekt-Typen A (SS#) und B (IHKS#) nicht kompatibel sind. Aus diesem Grund betrachten wir nun die alternativen Primärschlüssel für die Objekt-Typen A K_A={Name,Ort,Branche} und B K_B={Name}. Der erweiterte Schlüssel sei in diesem Fall $K_{Ext}=K_A \cup K_B=K_A$. Die korrespondierende Indentitätsregel für Objektpaare lautet somit in diesem Fall:

$$\forall e_1, e_2 \in E, (e_1.Name = e_2.Name) \wedge (e_1.Ort = e_2.Ort) \wedge (e_1.Branche = e_2.Branche) \rightarrow (e_1 \equiv e_2)$$

Da der Objekt-Typ B keine Attribute Ort und Branche besitzt ist diese Regel allerdings nicht direkt anwendbar. Wenn es allerdings möglich wäre, Objekt-Typ B unter Nutzung zusätzlicher Informationen um diese Attribute zu erweitern, so könnte die obige Regel zur Erkennung äquivalenter Objekte genutzt werden. Zum Beispiel läßt sich aus dem Attribut für die Postleitzahl (PLZ) des Objekt-Typs B mit Hilfe einer Datenbank, die die Zuordnungen der Postleitzahlen zu Orten enthält auf den vorliegenden Ort schließen (hier gilt 38820=HBS). Die Branchen-Information ließe sich aus dem Attribut Topseller, welches die am meisten verkaufte Produktart enthält, folgern, wenn man beispielsweise annimmt, daß ein Kaufhaus, das viel Pizza verkauft der Food-Branche angehört. Wendet man diese Überlegungen auf das vorliegende Beispiel an, kommt man zu dem Schluß, daß das erste City-Center aus A (SS# = 01863) äquivalent zum City-Center aus B ist.

Derartige semantische Zusatzinformationen werden in [LSP+93] als funktionale Abhängigkeiten auf Instanzebene (instance level functional dependencies, ILFDs) bezeichnet. Sie können genutzt werden, um Werte für fehlende Schlüsselattribute des erweiterten Schlüssels abzuleiten, der dann zur Untersuchung auf semantische Äquivalenz benutzt werden kann. Allgemein ist eine ILFD ein semantischer Constraint bezüglich der betrachteten Realweltobjekte. Sie hat die Form $\forall e \in E, (e.A_1=a_1) \wedge ... \wedge (e.A_n=a_n) \rightarrow (e.B = b)$, wobei die A_i und B Attribute und die a_i und b zugehörige konstante Attributwerte sind. Für unser Beispiel lauten die zwei verwendeten ILFDs:

$$(E.PLZ=38820) \rightarrow (E.Ort=„HBS“) \text{ und}$$
$$(E.Topseller=„Pizza“) \rightarrow (E.Branche=„Food“)$$

Es sei angemerkt, daß alle Datenbankobjekte die ILFDs erfüllen müssen.

Allgemein geht man bei der Anwendung des Konzeptes des erweiterten Schlüssels folgendermaßen vor: Als erstes wird der erweiterte Schlüssel K_{Ext} bestimmt. Hierzu werden die Schlüssel der beiden betrachteten Objekt-Typen vereinigt. Ist der entstandene Schlüssel für den integrierten Objekt-Typ nicht eindeutig, so werden Attribute des integrierten Objekt-Typs derart dem Schlüssel hinzuge-

fügt, daß eine eindeutige Identifikation der integrierten Objekte möglich ist. Für die Attribute, die Bestandteil von K_{Ext} sind, aber in den Ursprungs-Objekt-Typen fehlen, müssen ILFDs gefunden werden, auf deren Grundlage diese Attribute berechnet werden können. Die Existenz derartiger Abhängigkeiten ist eine Voraussetzung für die Anwendbarkeit des Verfahrens.

Sind die fehlenden Informationen berechnet, kann die Identifikation der äquivalenten Objekte durch den Vergleich der Werte des erweiterten Schlüssels erfolgen.

Der große Vorteil dieses Verfahrens gegenüber dem zuerst vorgestellten Konzept der Verwendung eines gemeinsamen Schlüssels ist, daß kein gemeinsamer Schlüssel erforderlich ist, um eine Erkennung semantisch äquivalenter Objekte durchzuführen. Das Verfahren ist somit mächtiger hinsichtlich der Formulierung der Kriterien für die Äquivalenz von Objekten und damit vielseitiger einsetzbar. Außerdem hat die Einbeziehung von zusätzlichen Informationen in Form von ILFDs den Vorteil, daß sich die Informationsmenge vergrößert, die für die Entscheidung, ob semantische Äquivalenz vorliegt, ausgewertet wird. Dadurch wird die Korrektheit der Ergebnisse gegenüber alternativen Ansätzen erhöht, die nur vorhandene (Nicht-Schlüssel-)Eigenschaften auswerten.

Das Hauptproblem des Verfahrens ist die Beschaffung des notwendigen Zusatzwissens. Hier ist ein Eingreifen des Menschen zum Beispiel in Person des Datenbankadministrators kaum zu vermeiden. Weiterhin stellt die mit der Berechnung der fehlenden Informationen verbundene Auswertung der ILFDs zusätzlichen Arbeitsaufwand dar.

Der Vorgang der Erkennung semantisch äquivalenter Datenbankobjekte entspricht der Berechnung eines Verbundes, wobei hier die Gleichheit des erweiterten Schlüssels als Bedingung auftritt. In [Dem89] wird ein sehr ähnliches Konzept zur Lösung eines allgemeineren Problems vorgeschlagen. Um beliebige relationale Operationen im Fall von unterschiedlichen Wertebereichen (mismatched domains) ausführen zu können, werden sogenannte virtuelle Attribute eingeführt. Diese Attribute werden mit Hilfe von Zusatzwissen aus den normalen Attributen berechnet. Das Zusatzwissen entspricht den ILFDs im Ansatz von [LSP+93]. Virtuelle und normale Attribute können dann zur Ausführung der Operationen (zum Beispiel Join) verwendet werden.

Das Konzept in [WM89] geht noch weiter. Hier wird vorgeschlagen, eine wissensverarbeitende Komponente in Informationssysteme zu integrieren. Die Erkennung von semantisch äquivalenten Objekten soll dann unter Nutzung von Hilfsdatenbanken und Regelmengen erfolgen. In den Hilfsdatenbanken ist hierzu Wissen über die Realwelt abgelegt, zum Beispiel Daten über die Entfernungen zwischen zwei geographischen Punkten der Erde. Die wissensverarbeitende Komponente wertet dann die Hilfsdatenbanken unter Anwendung der Regelmengen aus.

5.1.4 Kompatible Attributpaare

In [CTK96] wird ein wahrscheinlichkeitsbasiertes Konzept zur Identifikation semantisch äquivalenter Datenbankobjekte vorgeschlagen. Grundlage der Untersuchung der Objekte auf Äquivalenz sind dabei ihre kompatiblen Attribute.

Gegeben seien zwei Objekt-Typen C_1 und C_2. Die Attribute von C_1 werden durch $a_{11},...,a_{n1}$ und die von C_2 durch $a_{12},...,a_{m2}$ dargestellt. Die Menge der kompatiblen Attributpaare von C_1 und C_2 sei $KA=\{(a_{11},a_{12}),...,(a_{k1},a_{k2})\}$. Zwei Attribute aus C_1 und C_2 sind kompatibel und damit als kompatibles Attributpaar in KA enthalten, wenn sie semantisch äquivalent sind und ihr Wertebereich gleich ist beziehungsweise eine 1-1-Abbildung zwischen ihren Wertebereichen existiert. Es wird nun jedem kompatiblen Attributpaar ein Gewicht zugeordnet, welches die Bedeutung dieses Attributpaares festlegt. Diese Gewichte werden mit w(i) für das i-te Attributpaar bezeichnet. Jedes Gewicht entspricht einem Wert zwischen 0 und 1, und die Summe aller Gewichte beträgt 1. Das Gewicht eines Attributpaares ist von seiner Semantik abhängig. Zum Beispiel ist für eine Person der Name von größerer Bedeutung als das Alter.

Für zwei gegebene Datenbankobjekte o_1 und o_2 der Objekt-Typen C_1 und C_2 kann nun auf der Grundlage ihrer kompatiblen Attributpaare ein Wahrscheinlichkeitswert berechnet werden, der angibt, wie wahrscheinlich es ist, daß die beiden Objekte semantisch äquivalent sind. Dieser Wahrscheinlichkeitswert wird mit $P_{same}(o_1,o_2)$ bezeichnet. Seine Berechnung wird auf die Berechnung der Gleichheit der Attributwerte $o_1.a_{i1}$ und $o_2.a_{i2}$ für jedes kompatible Attributpaar (a_{i1},a_{i2}) zurückgeführt. Der Wert für die Gleichheit der betreffenden Attributwerte ist ein numerischer Wert zwischen 0 und 1 und wird mit $S_i(o_1.a_{i1},o_2.a_{i2})$ bezeichnet. Beträgt S_i für mindestens ein kompatibles Attributpaar 0, so wird $P_{same}(o_1,o_2)$ für die betroffenen Objekte auf 0 gesetzt. Sind alle S_i größer 0, berechnet sich $P_{same}(o_1,o_2)$ wie folgt:

$$P_{same}(o_1,o_2) = \sum_{i=1}^{k}\left(S_i\left(o_1\cdot a_{i1}, o_2\cdot a_{i2}\right)\bullet w(i)\right)$$

Für die Berechnung der $S_i(o_1.a_{i1},o_2.a_{i2})$ muß zunächst darauf hingewiesen werden, daß dominante und nicht dominante kompatible Attributpaare unterschieden werden. Ein kompatibles Attributpaar ist dominant, falls für die Werte der entsprechenden Attribute keine inkonsistenten Daten toleriert werden können. Anderenfalls ist es nicht dominant.

Sind die Werte für zwei kompatible Attribute zweier Objekte o_1 und o_2 gleich $(o_1.a_{i1}=o_2.a_{i2})$, dann gilt $S_i(o_1.a_{i1},o_2.a_{i2})=1$ für das betreffende kompatible Attributpaar. Treten für ein dominantes Attributpaar unterschiedliche Werte auf und es sind keine Null-Werte beteiligt, so gilt $S_i(o_1.a_{i1},o_2.a_{i2})=0$ und damit auch $P_{same}(o_1,o_2)=0$.

Treten Null-Werte als Werte für Attribute der kompatiblen Attributpaare auf, so gilt für die Berechnung der S_i unabhängig von der Dominanz der Attributpaare folgendes: Ist ein Wert ein Null-Wert und der andere Wert bekannt, so beträgt $S_i(v,NULL)=1/n$. Das n ist hierbei die Mächtigkeit des Wertebereichs des Attributes, welches den Null-Wert aufweist. Die Wahrscheinlichkeit, daß genau v als

Wert dieses Attributes auftritt ist demzufolge 1/n, wenn man annimmt, daß alle Werte des Wertebereichs gleich wahrscheinlich auftreten. Sind beide Werte Null-Werte, so ist S_i(NULL,NULL) ebenfalls 1/n. Die Erklärung hierfür ist, daß die Wahrscheinlichkeit, daß beide Werte gleich sind 1/(n*n) beträgt und der Wertebereich hierfür genau n Möglichkeiten bietet, also S_i=(1/(n*n))*n=1/n.

Es gilt also:

$$S_i(o_1.a_{i1},o_2.a_{i2}) = 1/n, \text{ falls } o_1.a_{i1} = NULL \vee o_2.a_{i2} = NULL$$

Für inkonsistente Daten im Fall von nicht dominanten kompatiblen Attributpaaren wird vorgeschlagen, die Wertegleichheit Si aus einer maximal erlaubten Abweichung und der tatsächlich vorhandenen Abweichung der Attributwerte zu berechnen. Übersteigt die Abweichung die maximal erlaubte Abweichung, so ist $S_i(o_1.a_{i1},o_2.a_{i2})$=0. Wie die Definition der Werteabweichung erfolgt, ist vom Wertebereich der Attribute und den Erfordernissen des jeweiligen Anwendungsbereiches abhängig und muß individuell festgelegt werden.

Betrachtet man das laufende Beispiel, so kann man feststellen, daß 3 kompatible Attributpaare existieren, namentlich die Attribute Name, Straße und Start-Datum. Wenn man nun für diese Attribute die Gewichte w(Name)=0.5, w(Straße)=0.3 und w(Start-Datum)=0.2 annimmt, die Attribute Name und Straße als dominant bewertet und Start-Datum als nicht dominant, so folgt für die Wahrscheinlichkeitswerte P_{same} folgendes Ergebnis:

OID (DB1)	OID (DB2)	Psame
A1	B1	1.00
A2	B1	0.94
A3	B1	0.00

Tabelle 5-2: Ergebnisse der P_{same}-Berechnung für das laufende Beispiel

Die Berechnung für A1 und B1 ist einfach, da sie in allen kompatiblen Attributen übereinstimmen. Für die Berechnung der Wahrscheinlichkeit für A2 und B1 wurde zunächst angenommen, daß für das nicht dominante Attributpaar Start-Datum eine maximale Abweichung von 7 Tagen zulässig sei. Für die Berechnung der Ähnlichkeit der Datumswerte soll das Verhältnis von tatsächlicher zu maximal zulässiger Abweichung maßgeblich sein. Da A2 und B1 in ihren dominanten Attributpaaren übereinstimmen beträgt sowohl S_1(A2.Name, B1.Name) als auch S_2(A2.Straße, B1.Straße) eins. Die Abweichung für das Eröffnungsdatum beträgt 2 Tage und somit gilt für die Ähnlichkeit der Datumswerte S_3(A2.Start-Datum, B1.Start-Datum) = 1-2/7 = 0.714. Unter Berücksichtigung der Attributgewichte ergibt P_{same}(A2, B1) = 1*0.5 + 1*0.3 + 0.714*0.2 = 0.94. Die Berechnung für A3 und B1 ist einfach, da für das dominante Attributpaar Straße keine Übereinstimmung besteht. Somit gilt P_{same}(A3, B1)=0.

Die Festlegung, welche Objekte als semantisch äquivalent betrachtet werden sollen, kann durch die Definition einer Grenzwahrscheinlichkeit α erfolgen. Um als semantisch äquivalent betrachtet zu werden, muß für zwei Objekte o_1 und o_2 $P_{same}(o_1,o_2) \geq \alpha$ gelten. Die Grenzwahrscheinlichkeit kann für verschiedene Anwendungsfälle individuell definiert werden. Für unser Beispiel sei $\alpha = 0.95$. Damit sind die Objekte A1 und B1 als semantisch äquivalent erkannt worden.

Voraussetzung für die Anwendung dieses Verfahrens ist das Vorhandensein von kompatiblen Attributpaaren. Im Vergleich zum Konzept der Verwendung eines erweiterten Schlüssels entfällt allerdings die unter Umständen aufwendige Beschaffung und Auswertung von Zusatzwissen. Die im letzten Unterabschnitt dargestellten Vorteile der Verwendung von Zusatzwissen sind aber nicht nutzbar. Eine Aussage über die semantische Äquivalenz von Datenbankobjekten erfolgt nicht mit Hilfe diskreter Werte, sondern unscharf über einen Wahrscheinlichkeitswert. Hierin besteht ein wesentlicher Unterschied zu den bisher dargestellten Konzepten. Die Korrektheit der Ergebnisse ist stark von der Definition der Grenzwahrscheinlichkeit α und der Gewichte w(i) abhängig, da durch sie maßgeblich festgelegt wird, welche Objekte als semantisch äquivalent betrachtet werden. Demzufolge muß die Grenzwahrscheinlichkeit in Abhängigkeit von den konkret zu untersuchenden Objekt-Typen sorgfältig festgelegt werden. Die Vollständigkeit ist für dieses Konzept erfüllt, da mit Hilfe der Grenzwahrscheinlichkeit die Menge der untersuchten Objektpaare in die zwei Mengen der äquivalenten und der nicht äquivalenten Objektpaare zerlegt wird.

5.2 Verwendung von Zusatzinformationen aus der Schemaintegration

In den vorangegangenen Unterabschnitten wurden verschiedene Konzepte zur Identifikation semantisch äquivalenter Datenbankobjekte erläutert. Die Verwendung von Zusatzinformationen aus der Schemaintegration stellt im Gegensatz zu diesen Konzepten kein eigenständiges Verfahren zur Erkennung von äquivalenten Objekten dar. Vielmehr kann durch die Einbeziehung derartiger Informationen ein Identifikationsverfahren erweitert werden.

Der Prozeß der Schemaintegration benötigt eine Vielzahl von semantischen Informationen über die zu integrierenden Schemakonstrukte und Datenbankobjekte. Es wäre vorteilhaft, wenn diese Informationen in der, auf der Schemaintegration aufsetzenden, Instanzintegration ausgenutzt werden könnten, um den Prozeß der Erkennung semantisch äquivalenter Objekte zu verbessern.

Eine Möglichkeit, Informationen der Schemaintegration auszunutzen besteht darin, die extensionalen Beziehungen der Objekt-Typen, die semantisch äquivalente Objekte enthalten, näher zu betrachten. In der Schemaintegration werden die extensionalen Überlappungen zu integrierender Objekt-Typen genau festgestellt. Mögliche extensionale Beziehungen zweier Objekt-Typen R und S beinhalten die Fälle:

- Identische Extensionen ($R \equiv S$)

- eine Extension schließt die andere ein ($R \subseteq S$)
- teilweise überlappende Extensionen ($R \cap S \neq \varnothing$)
- disjunkte Extensionen ($R \cap S = \varnothing$)

Wenn bekannt ist, welche Art der Überlappung der Extensionen der betreffenden Objekt-Typen vorliegt, so kann die Anzahl der auf semantische Äquivalenz zu untersuchenden Objektpaaren eingeschränkt werden.

Betrachten wir hierzu wieder unser Beispiel: Angenommen, der Objekt-Typ B enthält nur Kaufhäuser, die mit Lebensmitteln handeln, wogegen Objekt-Typ A auch andere Kaufhäuser enthält. Um welche Art von Kaufhaus es sich handelt wird für den Objekt-Typ A durch das Attribut Branche festgelegt. Ist nun bekannt, daß Objekt-Typ B nur Kaufhäuser der Food-Branche enthält, so kann die Untersuchung auf semantische Äquivalenz auf die Objekte des Objekt-Typs A beschränkt werden, deren Attribut Branche den Wert „Food“ enthält. Im Fall unsres Beispiels würde sich die Untersuchung dann auf die Objekte A1 und A3 beschränken.

Die extensionalen Beziehungen der Beispiel-Objekt-Typen werden durch Abbildung 5-1 noch einmal graphisch veranschaulicht:

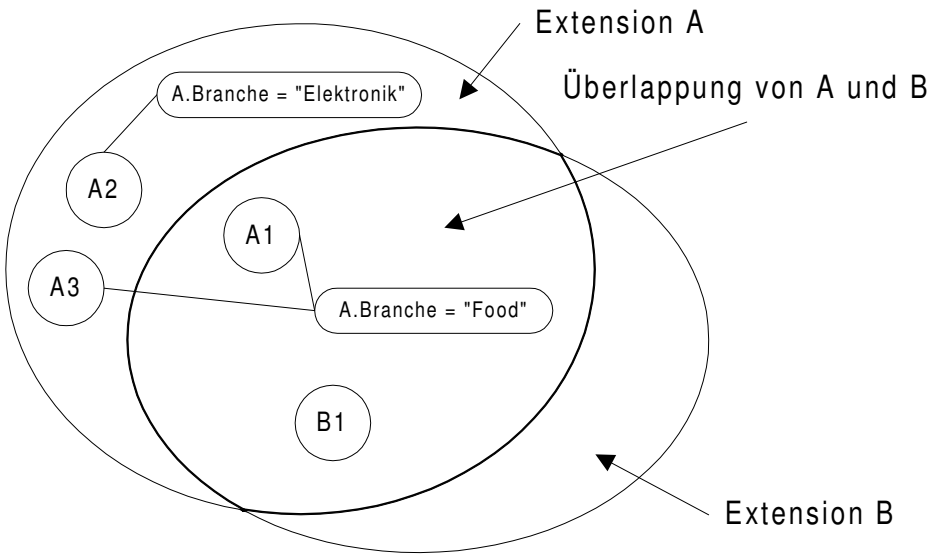


Abbildung 5-1: Extensionale Überlappung der Beispiel-Objekt-Typen

Durch die Verringerung der Anzahl der zu untersuchenden Objektpaare läßt sich der Aufwand zur Erkennung äquivalenter Objekte in Abhängigkeit davon, wieviele Objekte nicht betrachtet werden müssen, unter Umständen erheblich senken. Außerdem werden möglicherweise auftretende falsche Äquivalenzentscheidungen für die auszuschließenden Objekte vermieden. Die Vollständigkeit des Identifikationsverfahrens wird dahingehend verbessert, daß für die Objektpaare, welche Objekte enthalten, die ausgeschlossen werden konnten, sofort eine Entscheidung auf Nichtäquivalenz getroffen werden kann.

Abschließend wird eine Übersicht gegeben, die die Merkmale der vorgestellten Konzepte noch einmal zusammenfaßt:

	match-Bedingung	Zusatzwissen
Lookup Table	menschliches Wissen	menschliches Wissen
Common Key	gemeinsamer Primärschlüssel	kein
Extended Key	erweiterter Schlüssel	ILFD
Kompatible Attribute	kompatible Attribute	kein

	Einsetzbarkeit	Korrektheit	Vollständigkeit
Lookup Table	⇧	⇧	⇧
Common Key	⇩	⇧	⇧
Extended Key	⇨	⇧	⇧
Kompatible Attribute	⇨	⇨	⇨

Tabelle 5-3: Merkmale der Identifikationskonzepte

Die Pfeile in Tabelle 5-3 geben an, wie gut das betreffende Verfahren das jeweilige Merkmal erfüllt. Dabei ist der Pfeil um so mehr nach oben gerichtet, je besser das Merkmal erfüllt ist. Das Merkmal der Einsetzbarkeit bezieht sich auf Abschnitt 4.1 und gibt an, wie flexibel (beziehungsweise mächtig) das betreffende Verfahren hinsichtlich der Formulierung von match-Bedingungen ist. Anzumerken ist, daß bei der Bewertung einige Annahmen getroffen wurden:

Die Einsetzbarkeit des Lookup-Tabellen-Konzeptes ist stark abhängig von der Verfügbarkeit von Menschen, die das notwendige Wissen über die betrachteten Objekte, das zur Erstellung der Lookup-Tabelle notwendig ist, besitzen. Bei der Bewertung der Einsetzbarkeit dieses Konzeptes wurde vorausgesetzt, daß Menschen mit dem entsprechenden Wissen verfügbar sind.

Die Bewertung der Korrektheit und Vollständigkeit aller Konzepte geht von der Annahme aus, daß das betreffende Konzept im betrachteten Anwendungsfall auch anwendbar ist. Dies bedeutet, daß die individuellen Voraussetzungen für die Einsetzbarkeit des Verfahrens (nachzulesen im Unterabschnitt des jeweiligen Ansatzes) erfüllt sind. Zum Beispiel wird im Fall des Ansatzes der Identifikation semantisch äquivalenter Objekte unter Nutzung eines erweiterten Schlüssels vorausgesetzt, daß das notwendige Zusatzwissen in Form von ILFDs, welches zur Berechnung des erweiterten Schlüssels notwendig ist, auch verfügbar ist beziehungsweise beschafft werden kann.

5.3 Verwaltung semantisch äquivalenter Datenbankobjekte

Da die Integration der KDBSe in einem MDBS im allgemeinen virtueller Natur ist und insbesondere in MDBSen mit föderativem Charakter lokale Insert- und Dele-

te-Operationen stattfinden, wäre es eigentlich bei jeder Anfrage an die globale Datenbank notwendig, eine Identifikation semantisch äquivalenter Datenbankobjekte durchzuführen. Speziell für interaktive Datenbankanwendungen wird hierdurch das Antwortzeitverhalten unter Umständen stark negativ beeinflußt. Außerdem kann häufig davon ausgegangen werden, daß sich die Datenbestände und damit auch die Menge der äquivalenten Objektpaare nicht derart schnell verändern, daß ein Identifikationsprozeß für jede Anfrage erforderlich wäre.

Aus diesem Grund wurde in [ZHK+95], [SS95], [LSP+93], [CTK96] und weiteren Artikeln vorgeschlagen, die Informationen darüber, welche Objekte semantisch äquivalent sind, zu materialisieren und persistent in der globalen Datenbank abzuspeichern. Zu diesem Zweck werden die (globalen) OIDen beziehungsweise Primärschlüsselwerte der semantisch äquivalenten Objekte paarweise in einer Tabelle abgelegt. Eine derartige Tabelle wird in der Literatur als matching-Tabelle, same-Relation oder mapping-Tabelle bezeichnet.

Durch die Verwendung einer matching-Tabelle ist es möglich, den Aufwand zur Erkennung semantisch äquivalenter Objekte zu senken, da in diesem Fall nicht bei jeder Anfrage ein neuer Identifikationsprozeß gestartet werden muß. Jedoch stellt sich die Frage, wie die matching-Informationen angesichts lokaler Insert- und Delete-Operationen aktuell gehalten werden können.

In [ZHK+95] werden hierzu einige interessante Konzepte vorgestellt: Zum Beispiel kann die Aktualisierung der matching-Informationen inkrementell-ereignisgesteuert erfolgen. In diesem Falle wird bei einer lokalen Insert-Operation das MDBMS darüber informiert, daß eine derartige Operation (Ereignis) stattfand und welches Objekt eingefügt wurde. Das MDBMS kann dann überprüfen, ob dieses konkrete neue Objekt semantisch äquivalent zu einem bereits vorhandenen ist und gegebenenfalls einen neuen Eintrag in die matching-Tabelle einfügen. Analog wird im Fall einer Delete-Operation verfahren. Jedoch müssen hier gegebenenfalls Einträge aus der matching-Tabelle entfernt werden. Voraussetzung für die Realisierung dieser Strategie ist allerdings die Fähigkeit der lokalen Systeme, das MDBMS in geeigneter Weise durch Nachrichten zu informieren.

Sind die KDBSe nicht fähig, das MDBMS über ihre lokalen Aktivitäten zu informieren, so besteht eine weitere Möglichkeit in der periodischen Auffrischung der matching-Informationen. Hierbei wird in bestimmten zeitlichen Intervallen ein neuer Identifikationsprozeß angestoßen. Die Festlegung der Periodenlänge muß individuell für jeden Anwendungsfall bestimmt werden. Der Änderungsgrad der Daten in den lokalen Systemen hat hierbei einen entscheidenden Einfluß.

Für spezielle Anwendungsfälle wird in [ZHK+95] vorgeschlagen, neben den OIDen der semantisch äquivalenten Objekte, weitere Daten der integrierten Objekt-Typen zu materialisieren. Diese Verfahrensweise kann in Fällen von Vorteil sein, in denen bestimmte lokale Systeme, zum Beispiel infolge einer unsicheren Netzanbindung, nicht permanent dem MDBMS zugänglich sind.

5.4 Behandlung inkonsistenter Daten

Bei der (virtuellen) Integration semantisch äquivalenter lokaler Datenbankobjekte zu einem globalen integrierten Datenbankobjekt tritt in einem MDBS häufig das

Problem inkonsistenter Daten auf. Dies bedeutet, daß die betrachteten äquivalenten Objekte für bestimmte gemeinsame Attribute unterschiedliche Werte aufweisen. In diesem Fall ist zu klären, welcher Wert für das globale Objekt gelten soll.

Die Lösung dieses Problems ist einfach, falls bekannt ist, in welcher Datenbank die „richtigen“ Werte zu finden sind. Als „richtige“ Werte gelten die Werte, die den tatsächlichen Zustand des abgebildeten Realweltobjektes zum aktuellen Zeitpunkt widerspiegeln. In diesem Fall kann der entsprechende Wert für das globale Objekt eindeutig ausgewählt werden. Außerdem ist es zusätzlich möglich, die Datenbestände der beteiligten KDBSe abzugleichen indem „alte“ Werte aktualisiert werden.

Da sich aber häufig nicht sicher feststellen läßt, welcher Wert für ein gemeinsames Attribut zweier äquivalenter Objekte der Wert ist, der den aktuellen Zustand des betrachteten Realweltobjektes widerspiegelt, werden leistungsfähigere Konzepte zur Lösung derartiger Konsistenzprobleme benötigt.

Ein Lösungsvorschlag wird in [CTK96] gegeben. Hier wird das Problem mit Hilfe des Konzeptes der wahrscheinlichkeitsbasierten partiellen Werte gelöst. Dieses Konzept ist eine Weiterentwicklung des in [Dem89] vorgeschlagenen Konzeptes der partiellen Werte.

Die Idee dieses Konzeptes besteht darin, daß beim Auftreten widersprüchlicher Daten für ein gemeinsames Attribut, diesem Attribut für das globale Objekt ein wahrscheinlichkeitsbasierter partieller Wert zugeordnet wird. Ein wahrscheinlichkeitsbasierter partieller Wert gibt keine exakte Auskunft über den tatsächlichen Wert eines Attributes. Vielmehr enthält er eine Menge von Werten, die ihrerseits mit Wahrscheinlichkeitswerten versehen sind. Genau ein Wert aus der Menge, der in einem wahrscheinlichkeitsbasierten partiellen Wert enthaltenen Werte ist der wahre Wert für das Attribut. Die Wahrscheinlichkeitswerte der einzelnen Elemente geben an, wie wahrscheinlich es ist, daß dieser Wert der wahre ist. Jedoch ist nicht notwendigerweise der Wert mit der höchsten Wahrscheinlichkeit auch der wahre. Als Beispiel sei der folgende wahrscheinlichkeitsbasierte partielle Wert gegeben:

$$O.Name = \left[\text{"}Müller\text{"}^{\frac{1}{4}}, \text{"}Meier\text{"}^{\frac{1}{4}}, \text{"}Schulze\text{"}^{\frac{1}{2}} \right]$$

Er drückt aus, daß der wahre Wert für das Attribut Name des Objektes O entweder die Zeichenkette „Müller“ mit einer Wahrscheinlichkeit von ¼ oder „Meier“ mit einer Wahrscheinlichkeit von ¼ oder „Schulze“ mit einer Wahrscheinlichkeit von ½ ist.

Für weitergehende Informationen über das Konzept der wahrscheinlichkeitsbasierten partiellen Werte sei auf [CTK96] verwiesen.

6 Einsatz von Identifikationsverfahren in MDBSen

Wie bereits bei der Beschreibung des Problems der Identifikation semantisch äquivalenter Datenbankobjekte ausgeführt wurde, kann dieses Problem für den allgemeinen Fall nicht algorithmisch gelöst werden. Aus diesem Grund muß darüber nachgedacht werden, wie in einem konkreten Anwendungsfall vorgegangen werden sollte, um eine Lösung des Identifikationsproblems für diesen konkreten Fall zu finden.

Im vorangegangenen Abschnitt wurde einige Konzepte zur Identifikation äquivalenter Objekte erläutert. Jedoch ist jedes dieser Konzepte immer nur für eine begrenzte Anzahl von Anwendungsfällen erfolgreich einsetzbar. Will man jedoch ein MDBS mit Fähigkeiten zur Identifikation von äquivalenten Objekten ausstatten, so sollten diese Fähigkeiten für möglichst viele Fälle anwendbar sein.

Um dieses Ziel zu erreichen ist es sinnvoll zu untersuchen, welche Informationen zur Identifikation von äquivalenten Objekten herangezogen werden können. Ein möglichst breites Spektrum an Grundkonzepten und auswertbaren Informationen könnte dann in Form von Einzelkomponenten in einer Toolbox zur Identifikation semantisch äquivalenter Objekte als Bestandteil eines MDBSs zusammengefaßt werden.

Für einen konkreten Anwendungsfall ist es dann möglich aus der Menge von Grundkonzepten und zusätzlich auswertbaren Informationen ein individuell angepaßtes Identifikationsverfahren zusammenzustellen.

In den folgenden Unterabschnitten soll zunächst dargestellt werden, welche Informationen als Eingabeparameter für eine Erkennung äquivalenter Objekte erforderlich sind und welche Informationen, in Abhängigkeit vom konkreten Anwendungsfall, möglicherweise noch zusätzlich einbezogen werden können oder sollten.

Weiterhin soll anhand eines Entscheidungsgraphen verdeutlicht werden, wie man ein anwendbares Identifikationsverfahren für einen konkreten Anwendungsfall auswählen kann.

Abschließend sollen Überlegungen dargelegt werden, welche Anforderungen an ein MDBS gestellt werden müssen, in dem ein flexibles (an spezielle Anwendungsfälle anpaßbares) Modul zur Erkennung semantisch äquivalenter Objekte realisiert werden soll.

6.1 Informationsbasis für Identifikationsverfahren

Ein Verfahren zur Identifikation von semantisch äquivalenten Objekten benötigt als Informationsbasis eine Menge von Metadaten über die zu untersuchenden Datenbankobjekte, auf deren Grundlage die Erkennung äquivalenter Objekte erfolgen kann. Eine wichtige Frage, die sich in diesem Zusammenhang stellt, lautet: Wie können diese wichtigen Informationen beschafft werden?

Folgende Informationen können, hinsichtlich der Identifikation äquivalenter Objekte, von Identifikationsverfahren ausgewertet werden:

- Informationen darüber, welche Objekt-Typen eine extensionale Überlappung aufweisen
- Informationen über vorhandene gemeinsame Eigenschaften
- Informationen über die Bedeutung der einzelnen Eigenschaften
- Dominanz von Eigenschaften
- Zusatzinformationen aus der Schemaintegration
- Zusatzwissen zur Ableitung neuer Eigenschaften
- Informationen über die Verläßlichkeit der Daten

Nachfolgend soll auf die genannten auswertbaren Informationen näher eingegangen werden.

Voraussetzung für die Erkennung äquivalenter Objekte sind Informationen darüber, welche Objekt-Typen der lokalen Datenbanken eines MDBSs semantisch äquivalente Objekte enthalten oder anders ausgedrückt, eine extensionale Überlappung aufweisen. Da diese Informationen auch für die Durchführung einer Schemaintegration der lokalen Schemata zu einem integrierten Gesamtschema benötigt werden, können sie aus der Schemaintegrationsphase übernommen werden.

Weiterhin sind Informationen über vorhandene gemeinsame Eigenschaften der betrachteten Objekt-Typen von großer Bedeutung. Eigenschaften sind den jeweiligen Objekt-Typen gemeinsam, wenn sie semantisch äquivalent (ihrer Realweltbedeutung nach gleich) sind und ihre Wertebereiche gleich oder durch eine 1-1-Abbildung aufeinander abbildbar sind. Die Informationen darüber, welche Eigenschaften äquivalent sind, können ebenfalls aus der Schemaintegration übernommen werden.

Wie bereits im vorangegangenen Abschnitt bei der Diskussion des Identifikationsansatzes unter Nutzung kompatibler Attributpaare ausgeführt wurde, besitzen nicht alle Eigenschaften eines Objekt-Typs die gleiche Bedeutung hinsichtlich der Identifikation semantisch äquivalenter Objekte. Vielmehr lassen sich einzelnen Eigenschaften spezifische Gewichte zuordnen, die in den Identifikationsprozeß einbezogen werden können. Ebenfalls Ausdruck der Wichtigkeit einer Eigenschaft ist ihre Dominanz. Eigenschaften werden hierbei in dominante und nicht dominante Eigenschaften eingeteilt. Die Dominanz einer Eigenschaft gibt an, ob für diese Eigenschaft abweichende Werte zweier Objekte toleriert werden können oder nicht. Unterschiede in den Werten dominanter Eigenschaften führen direkt zur Feststellung fehlender semantischer Äquivalenz, wogegen Unterschiede in nicht dominanten Eigenschaften bis zu einem bestimmten Grad (maximum difference distance) toleriert werden.

Der Bereich der extensionalen Überlappung der betrachteten Objekt-Typen läßt sich in bestimmten Fällen durch Zusatzinformationen aus der Schemaintegration näher spezifizieren. Ein Beispiel hierfür wurde im letzten Abschnitt angeführt. Hierdurch wird der Aufwand zur Identifikation semantisch äquivalenter Objekte gesenkt, da die Menge potentiell äquivalenter Objekte verkleinert wird.

Weiterhin ist es möglich durch die Verwendung von Zusatzwissen über die abgebildeten Realweltobjekte, neue Eigenschaften für die Datenbankobjekte der betreffenden Objekt-Typen abzuleiten. Dies kann unter Nutzung von Informationen zusätzlicher Datenbanken erfolgen und verbreitert die Informationsbasis, auf deren Grundlage die Untersuchung auf semantische Äquivalenz erfolgen kann.

Informationen darüber, wie verläßlich die Daten der einzelnen lokalen Systeme sind, lassen sich ausnutzen, um nach einer erfolgreichen Identifikation äquivalenter Objekte deren Integration zu einem globalen (virtuellen) Objekt, insbesondere bei widersprüchlichen Angaben für bestimmte Eigenschaften, zu begünstigen.

Quelle der Metainformationen ist primär der Mensch, insbesondere in Gestalt eines Datenbankadministrators oder -designers. Er ist derzeit am besten befähigt, notwendige Realweltsemantik der zu untersuchenden Datenbankobjekte zu erfassen und wiederzugeben. Dies geht einheitlich aus allen betrachteten Publikationen zur Problematik der Identifikation semantisch äquivalenter Datenbanksysteme hervor.

Als alternative Quellen zur automatischen Beschaffung von Metainformationen werden in [LSP+93] Wissensaquisitionstools und in [CTK96] neuronale Netze angegeben.

6.2 Auswahl eines Identifikationsverfahrens

Da ein Identifikationsverfahren nur unter spezifischen Voraussetzungen einsetzbar ist, muß für jeden Anwendungsfall geklärt werden, welche Verfahren eingesetzt werden können beziehungsweise eingesetzt werden sollten.

Grundlage dieser Entscheidung ist ein Vergleich der Bedingungen, die für den Einsatz eines Verfahrens erfüllt sein müssen, mit den Gegebenheiten des betreffenden Anwendungsfalles.

Ein Mensch (zum Beispiel Datenbankadministrator), der nach einem, für seinen Anwendungsfall, geeigneten Identifikationsverfahren sucht, kann durch eine Reihe von zu beantwortenden Fragen, organisiert in einem Entscheidungsgraphen, zu einer geeigneten Auswahl eines Verfahrens geführt werden. Es wäre denkbar, einen derartigen Entscheidungsgraph in Form eines Auswahl-Assistenten als Bestandteil der Komponente zur Identifikation semantisch äquivalenter Objekte eines MDBSs zu implementieren.

Der Vorgang der Auswahl eines geeigneten Verfahrens ließe sich weiter automatisieren, wenn es möglich wäre, das Wissen zur Beantwortung der Auswahlfragen, zumindest teilweise, automatisch zu beschaffen. Wie die automatische Beschaffung des erforderlichen Wissens, zum Beispiel mit Hilfe von Wissensaquisitionstools, erfolgen kann, soll im Rahmen dieser Arbeit nicht näher betrachtet werden.

Als Ergebnis der Untersuchung ausgewählter Identifikationskonzepte im letzten Abschnitt wurde der in Abbildung 6-1 dargestellte Entscheidungsgraph als Hilfe zur Auswahl eines geeigneten Verfahrens entwickelt. Es handelt sich hierbei um einen gerichteten azyklischen Graph.

Nachfolgend soll nach einigen vorbereitenden Worten das Auswahlverfahren, entsprechend Abbildung 6-1 erläutert werden.

Der Ausgangspunkt beim Prozeß der Auswahl eines Identifikationsverfahrens sind die beiden Objekt-Typen A und B, die überlappende Untermengen einer Menge von Realweltobjekten E modellieren. Die Grundidee beim Aufbau des Graphen besteht dabei darin, Gemeinsamkeiten zwischen den Eigenschaften der Realweltobjekte, die in den betrachteten Objekt-Typen A und B abgebildet wurden festzustellen, um so auf die möglicherweise vorhandene „größte“ Gemeinsamkeit in Gestalt der semantischen Äquivalenz zu schließen. Die größte Priorität wird dabei der Ausnutzung der Primärschlüsseleigenschaften der Objekt-Typen zugeordnet. Im Idealfall sind die Primärschlüssel der Objekt-Typen kompatibel. Ist dies nicht der Fall, so lassen sich möglicherweise mit Hilfe von Zusatzwissen erweiterte Schlüssel (extended keys) bilden. Ist auch dies nicht möglich, wird auf sonstige gemeinsame Eigenschaften der beiden betrachteten Objekt-Typen zurückgegriffen. Sind derartige gemeinsame Eigenschaften nicht vorhanden und auch nicht durch Zusatzwissen „beschaffbar“, so ist keine automatische Identifikation semantisch äquivalenter Objekte möglich und es bleibt nur die Möglichkeit der Nutzung einer Lookup-Tabelle.

Bei der Betrachtung vorhandener gemeinsamer Eigenschaften der Objekt-Typen wird in diesem Zusammenhang von einer Wichtung der in den jeweiligen Objekt-Typen vorkommenden Eigenschaften der Realweltobjekte aus E ausgegangen. Das Gewicht einer Eigenschaft (hier auch als Relevanz bezeichnet) ist einerseits Ausdruck des semantischen Reichtums der betreffenden Eigenschaft und andererseits Ausdruck der Fähigkeit, ein Objekt des betreffenden Objekt-Typs zu identifizieren.

Zum Beispiel ist der Name einer Person semantisch reich und besitzt ein hohes Potential, eine Person zu identifizieren und damit ein hohes Gewicht. Die Schuhgröße einer Person ist eine semantisch reiche Eigenschaft. Jedoch ist sie sicher wenig tauglich, eine Person zu identifizieren. Aus diesem Grund wird ihr Gewicht niedriger sein als das des Namens. Eine Personalausweisnummer ist semantisch arm, da es sich um eine künstliche Eigenschaft einer Person handelt. Dennoch ist ihr Potential, eine Person zu identifizieren hoch. Bei der Festlegung ihres Gewichtes muß dies berücksichtigt werden.

Die einzelnen Gewichte sind subjektive Größen für die Bedeutsamkeit der Eigenschaften von Datenbankobjekten. Sie müssen bestimmt werden bevor der Auswahlprozeß beginnt. Eine Möglichkeit der Festlegung besteht darin, eine qualitative Bewertung durchzuführen. Hierbei könnten die Eigenschaften in eine Menge von Relevanzklassen eingeteilt werden (zum Beispiel drei Relevanzklassen: geringe Relevanz, mittlere Relevanz und hohe Relevanz).

In [CTK96] wird hinsichtlich der Wichtung von Eigenschaften vorgeschlagen, die Bedeutsamkeit einer Eigenschaft als numerischen Wert zwischen 0 und 1 anzugeben. Der betreffende Wert für die Bedeutung einer Eigenschaft gibt an, wie groß der Bedeutungsanteil der Eigenschaft im Verhältnis zu den anderen Eigenschaften des betrachteten Objekt-Typs ist. Die Summe aller Gewichte eines Objekt-Typs ist damit stets 1.

Nach diesen vorbereitenden Darlegungen soll nun der Ablauf des Auswahlverfahrens für ein Identifikationsverfahren gemäß dem vorgeschlagenen Entscheidungsgraphen (siehe Abbildung 6-1) erläutert werden.

Auswahlverfahren

Das Verfahren beginnt im Startzustand mit der Feststellung, daß die beiden Objekt-Typen A und B überlappende Extensionen besitzen. Die erste Frage, die zu beantworten ist, betrifft die Existenz gemeinsamer Eigenschaften der Objekt-Typen A und B. Kann diese Frage mit einem Ja beantwortet werden, so existieren semantisch äquivalente Eigenschaften, die möglicherweise zur Identifikation der äquivalenten Objekte genutzt werden können. Durch die darauf folgende Frage nach der Existenz eines gemeinsamen Primärschlüssels soll die Art der vorhandenen intensionalen Überlappung weiter eingeschränkt werden. Als gemeinsame Primärschlüssel werden an dieser Stelle alle Primärschlüsselkandidaten der beiden Objekt-Typen A und B in Betracht gezogen. Ist tatsächlich ein gemeinsamer Primärschlüssel (common key) vorhanden, so kann er zur korrekten und vollständigen und darüber hinaus auch effizienten Identifikation der semantisch äquivalenten Objekte genutzt werden. In diesem Fall endet das Auswahlverfahren an dieser Stelle.

Konnte die Frage nach dem gemeinsamen Primärschlüssel nur negativ beantwortet werden, so ist zu klären, ob die Schlüsseleigenschaften des Objekt-Typs A, die dem Objekt-Typ B fehlen für die Objekte in B mit Hilfe von Zusatzwissen abgeleitet werden können. Entsprechendes gilt umgekehrt für die Schlüssel-Eigenschaften von B, die in A nicht vorhanden sind. Ist die Berechnung der fehlenden Eigenschaften möglich, so ist damit wiederum ein gemeinsamer Primärschlüssel durch die Nutzung von Zusatzwissen erzeugt worden. Da jedoch nicht die Original-Schlüssel der Objekt-Typen verwendet werden, sondern diese einer Erweiterung unterzogen werden, spricht man in diesem Fall von einer Identifikation mit Hilfe eines erweiterten Schlüssels (extended key).

Lassen sich jedoch die gegenseitig fehlenden Schlüsseleigenschaften nicht durch Zusatzwissen ableiten, so kann die Identifikation der semantisch äquivalenten Objekte anhand der gemeinsamen Eigenschaften (kompatible Attributpaare) erfolgen, falls diese ein ausreichend hohes Gewicht besitzen. Die Entscheidung darüber, ob die Bedeutung der gemeinsamen Eigenschaften (Gewicht) ausreichend ist, um eine hinreichend erfolgreiche Identifikation zu gewährleisten muß durch die Administratoren der beteiligten Datenbanken getroffen werden. Zum Beispiel besitzt das Alter einer Person, hinsichtlich der Identifikation der betreffenden Person, sicher ein geringeres Gewicht als ihr Name. Die Identifikation semantisch äquivalenter Personen-Objekte anhand ihres Alters ist wahrscheinlich nicht sinnvoll durchführbar, wogegen die Verwendung der Namensangabe bessere Möglichkeiten bietet. Existieren mehrere gemeinsame Eigenschaften, so muß an dieser Stelle ihr Gesamtgewicht als Summe der Einzelgewichte betrachtet werden.

Wird die Bedeutsamkeit der gemeinsamen Eigenschaften als zu gering eingeschätzt, so bleibt zu prüfen, ob sich gemeinsame Eigenschaften hoher Relevanz durch Zusatzwissen ableiten lassen. Ist dies möglich, so können diese neuen Ei-

genschaften (kompatible Attributpaare) zur Identifikation der äquivalenten Objekte genutzt werden. Anderenfalls bleibt nur die Nutzung einer Lookup-Tabelle zur Festlegung, welche Objekte semantisch äquivalent sind.

Konnte die eingangs gestellte Frage nach vorhandenen gemeinsamen Eigenschaften nur negativ beantwortet werden, so ist zu untersuchen, ob neue Eigenschaften durch Zusatzwissen ableitbar sind. Ist dies nicht möglich gibt es keinerlei Anhaltspunkte zur automatischen Identifikation der semantisch äquivalenten Objekte und es bleibt nur die Nutzung einer Lookup-Tabelle.

Anderenfalls ist zu klären, ob die Primärschlüssel der Objekt-Typen durch Zusatzwissen in der oben beschriebenen Weise erweiterbar sind. Wenn eine entsprechende Erweiterung der Schlüssel möglich ist, so kann der erweiterte Schlüssel (extended key) zur Identifikation äquivalenter Objekte verwendet werden.

Ist keine Schlüsselerweiterung möglich, so können möglicherweise neue gemeinsame Eigenschaften (kompatible Attributpaare) hoher Relevanz durch Zusatzwissen abgeleitet werden, die dann als Grundlage der Identifikation äquivalenter Objekte dienen. Können auch keine neuen Eigenschaften abgeleitet werden, so bleibt wiederum nur die Nutzung einer Lookup-Tabelle.

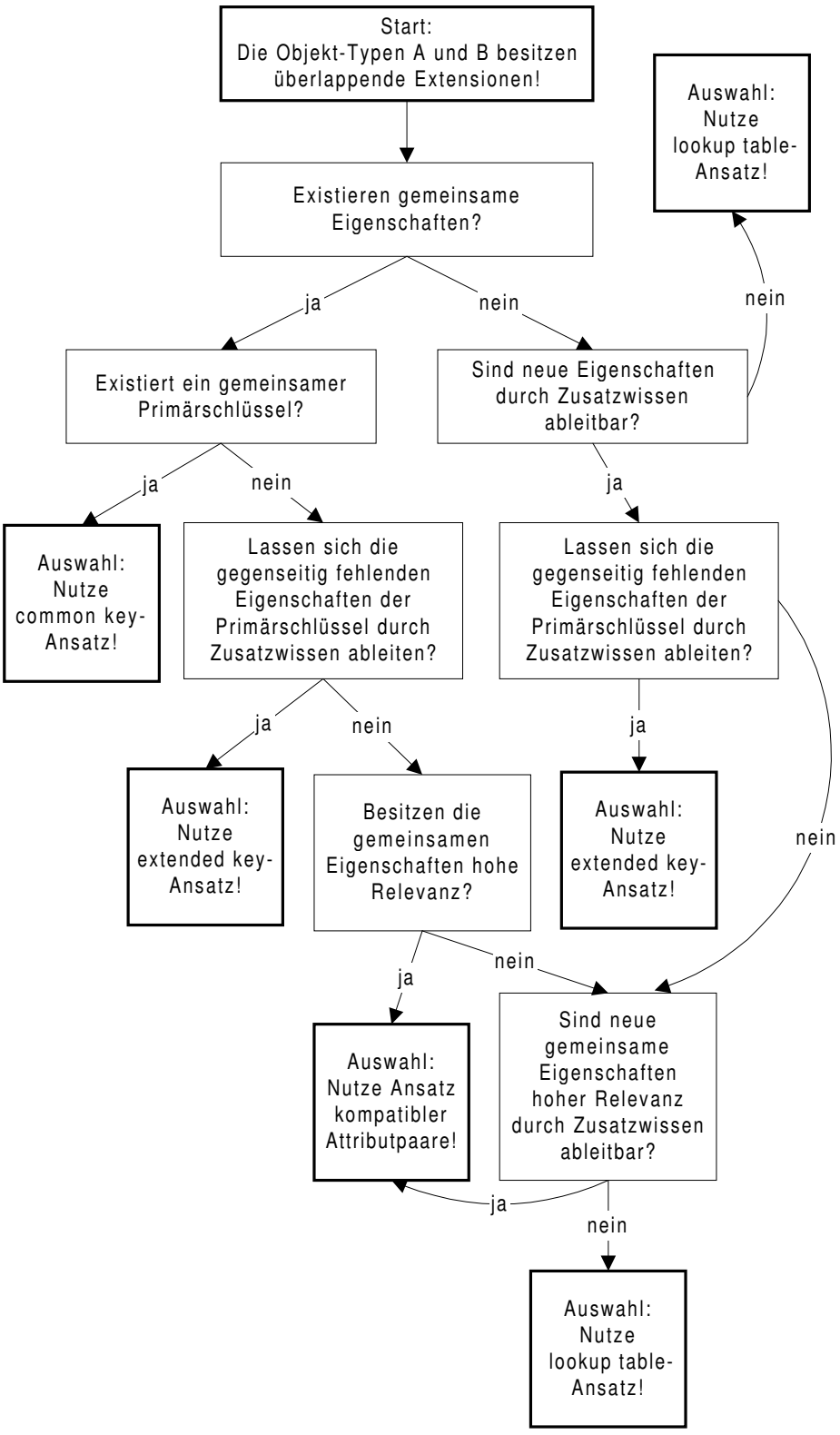


Abbildung 6-1: Entscheidungsgraph zur Auswahl eines Identifikationsverfahrens

6.3 Funktionale Anforderungen an MDBSe

Die Realisierung von Konzepten zur Identifikation semantisch äquivalenter Datenbankobjekte in einem MDBS erfordert das Vorhandensein verschiedener zusätzlicher Datenstrukturen und funktionaler Voraussetzungen. Diese Voraussetzungen beinhalten:

- erweiterte globale Metadatenstrukturen
- eine Identifikationskomponente
- ein Administrations-Tool für die Identifikationskomponente
- eine erweiterte Anfragebearbeitungskomponente

In Abbildung 6-2 sind die Datenstrukturen und funktionalen Komponenten eines MDBSs, die der Erkennung, Verwaltung und Verarbeitung von semantisch äquivalenten Objekten dienen, schematisch dargestellt. Kernbestandteil der Abbildung ist das MDBMS, welches eine funktionale Komponente zur Erkennung äquivalenter Objekte enthält. Diese Identifikationskomponente stellt eine Toolbox dar, die Identifikationswerkzeuge zur Erkennung äquivalenter Objekte auf der Basis einer Menge von Basiskonzepten (siehe Abschnitt 5) anbietet.

Bevor die Identifikationskomponente zum Einsatz kommen kann, muß sie konfiguriert werden. Hierbei wird mit Hilfe eines Administrationstools ein, für den konkreten Anwendungsfall, geeignetes Identifikationskonzept (zum Beispiel auf Grundlage des Entscheidungsgraphen aus Abbildung 6-1) ausgewählt. Außerdem werden die globalen Metadaten um Informationen, die zur Erkennung semantisch äquivalenter Objekte auf der Basis des gewählten Identifikationsverfahrens notwendig sind, erweitert. Diese Metadaten werden ebenfalls mit Hilfe des Administrationstools der Identifikationskomponente übermittelt und in der globalen Datenbank gespeichert.

Bei der Ausführung des gewählten Identifikationsverfahrens durch die Identifikationskomponente werden, unter Nutzung der globalen Metadaten (enthalten Abbildung von lokalen (Schema-)Objekten auf globale (Schema-)Objekte) und der zusätzlichen Metadaten zur Identifikation, die lokalen Objekte der betreffenden KDBSe gelesen und auf semantische Äquivalenz untersucht. Die Ergebnisse dieser Untersuchung werden in einer matching-Tabelle abgelegt. Sie enthält Paare von OIDen semantisch äquivalenter Objekte und verhindert, daß eine Identifikation bei jeder (globalen) Datenbankanfrage notwendig ist.

Werden im alltäglichen Betrieb des MDBSs durch globale Applikationen Anfragen an die integrierte Datenbank gestellt, so nutzt die Anfragebearbeitungskomponente des MDBSs neben den globalen Metadaten die Informationen der matching-Tabellen, um aus den lokalen Objekten integrierte globale Objekte zu erstellen, die dann in Form von Anfrageergebnissen den Applikationen übergeben werden. Voraussetzung hierfür ist eine Anfragebearbeitungskomponente, die derart modifiziert wurde, daß sie zusätzlich zur „normalen“ Anfragebearbeitung auch noch eine Verarbeitung der matching-Informationen erlaubt.

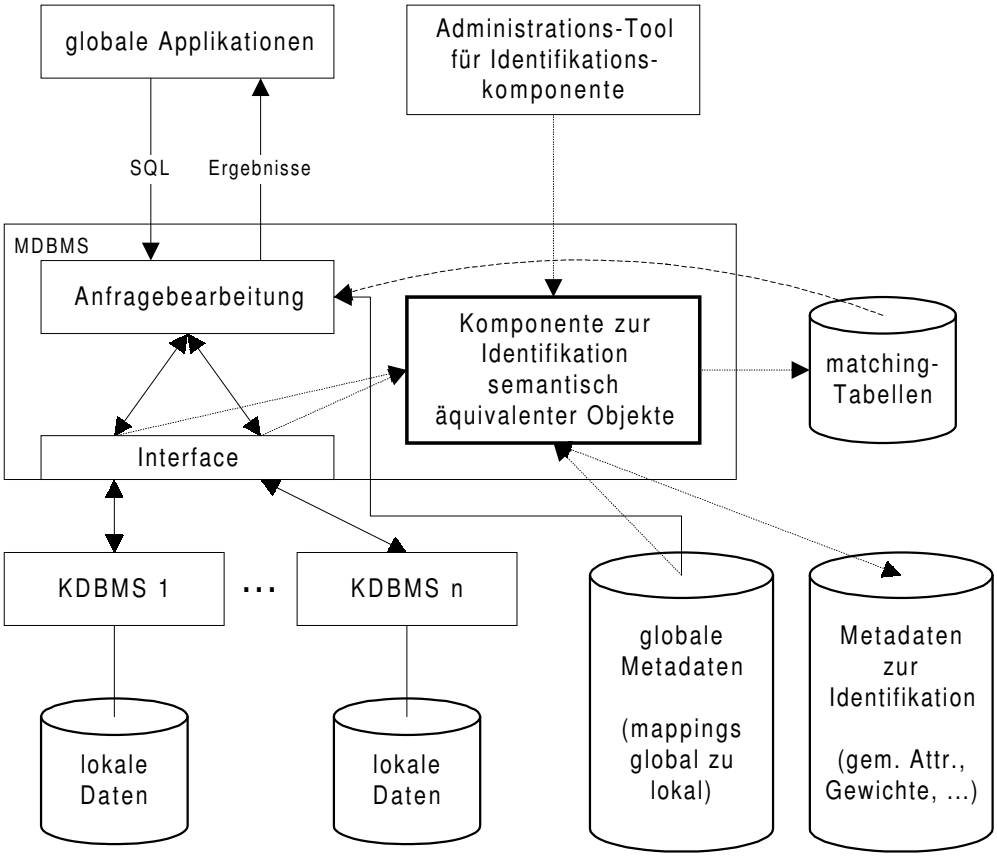


Abbildung 6-2: Einsatz einer Identifikationskomponente

In diesem Unterabschnitt wurde der Einsatz von Identifikationsverfahren in MDBSen konzeptionell diskutiert, ohne auf konkrete MDBSe einzugehen. In Abschnitt 9 wird auf der Basis der Erkenntnisse dieses Unterabschnittes untersucht, wie Identifikationsverfahren in dem objektrelationalen MDBS UniSQL/M realisiert werden können.

7 Das objektrelationale MDBS UniSQL/M

Nachdem in den vorangegangenen Abschnitten auf allgemeine Konzepte zur Identifikation semantisch äquivalenter Datenbankobjekte eingegangen wurde, soll das Problem der Erkennung äquivalenter Objekte in diesem und den weiteren Abschnitten am Beispiel eines konkreten MDBSs untersucht werden. Als Beispiel dient das objektrelationale MDBS UniSQL/M. Dieses MDBS wird von der UniSQL Inc. in Austin, Texas (USA) entwickelt. Im Rahmen dieser Arbeit findet die Version 3.5 dieses Systems Verwendung.

In diesem Abschnitt soll zunächst auf die Merkmale objektrelationaler Systeme eingegangen werden. Daran anschließend wird dann das Datenmodell von UniSQL/M erläutert. Den Schwerpunkt bildet dabei die Frage: Wie unterstützt UniSQL/M die Identifikation semantisch äquivalenter Objekte?

Abschließend wird gezeigt, wie die Einbindung von Komponentensystemen in eine UniSQL/M-Datenbank erfolgt.

7.1 Objektrelationale Datenbanksysteme

In den achtziger Jahren konnten sich relationale Datenbanksysteme als marktbeherrschend durchsetzen. Mit dem breiten Einsatz relationaler Systeme wurden aber auch die Grenzen dieser Systeme deutlich. Während sie für sogenannte Standardanwendungen zum Beispiel im Versicherungs- und Bankwesen sehr gut geeignet waren, zeigte sich, daß sie für sogenannte Nichtstandardanwendungen, zum Beispiel in den Bereichen CAD, Multimedia oder geographische Informationssysteme nur begrenzt einsetzbar sind. Aus diesem Grund wurden Datenbanksysteme entwickelt, die objektorientierte Konzepte unterstützen und auf einem ein Objektdatenmodell basieren.

Eine Klasse von Objektdatenbanksystemen (ODBSen) bildet die der objektrelationalen Datenbanksysteme (ORDBSe). ORDBSe werden gemäß [SST97] als strukturiert objektorientiert bezeichnet und verfolgen einen evolutionären Ansatz. Dies bedeutet, daß die bewährte relationale Datenbanktechnologie um objektorientierte Konzepte erweitert wird. Um welche Konzepte es sich im Fall von UniSQL/M handelt soll im folgenden Unterabschnitt dargelegt werden.

Der Vorteil einer derartigen Weiterentwicklung liegt darin, daß so ein sanfter Übergang zum objektorientierten Paradigma ermöglicht wird. Das Datenmodell, welches von objektrelationalen Systemen verwendet wird, ist kompatibel zum relationalen Modell. Aus diesem Grund ist es möglich, bestehende Anwendungssysteme mit sehr geringem Aufwand zu migrieren.

7.2 Das Datenmodell von UniSQL/M

Das Datenmodell von UniSQL/M ist eine Verallgemeinerung des relationalen Datenmodells. Es unterstützt somit alle Konzepte, die auch das relationale Modell bietet. Jedoch wurden weitere (objektorientierte) Konzepte hinzugefügt. Die Begriffe Klasse und Tabelle, virtuelle Klasse und Sicht, Objekt und Tupel, Attribut

und Spalte sowie Methode und Funktion werden in UniSQL/M synonym verwendet. Die jeweils ersten Begriffe deuten auf den objektorientierten Charakter von UniSQL/M hin und werden im Rahmen dieser Arbeit verwendet.

An dieser Stelle soll ein kurzer Überblick über einige Erweiterungen des relationalen Datenbankmodells eingegangen werden. Detaillierte Informationen über das Datenmodell von UniSQL/M können [SST97] und [Uni96a] entnommen werden.

Eine Erweiterung des Relationenmodells besteht in der Unterstützung von Kollektionstypen. Mit ihrer Hilfe ist es möglich, komplexe Datentypen zu bilden. Folgende Kollektionstypen werden in UniSQL/M angeboten:

- **`set`**: Menge (keine Duplikate)
- **`multiset`**: Multimenge (Duplikate möglich)
- **`list`**: (oder `sequence`): geordnete Liste (Duplikate möglich)

Der `tuple`-Typkonstruktor ist in UniSQL/M nicht vorhanden.

Jedes Objekt einer UniSQL/M-Datenbank besitzt einen eindeutigen und systemvergebenen OID. OIDen können unter anderem verwendet werden, um Referenzen zwischen Objekten aufzubauen. Es ist ferner möglich, Klassen in Spezialisierungshierarchien einzuordnen. Dabei wird auch die Mehrfachspezialisierung unterstützt.

In UniSQL/M ist es möglich, das Verhalten einer Klasse in Form von benutzerdefinierten Methoden zu modellieren.

Speziell für die Realisierung einer Multidatenbank, in welche eine Menge von KDBSen einbezogen wird, existiert in UniSQL/M das Proxy-Konzept. Ein Proxy ist ein Schemaobjekt der globalen Datenbank, welches als Stellvertreter für ein lokales Schemaobjekt (zum Beispiel eine Tabelle eines relationalen KDBSs) auftritt.

7.3 Identifikation äquivalenter Objekte in UniSQL/M

Von besonderem Interesse für diese Arbeit ist die Frage: Wie unterstützt UniSQL/M die Identifikation semantisch äquivalenter Datenbankobjekte?

In den Handbüchern zu UniSQL/M in der aktuellen Version 3.5 wird darauf eingegangen, daß in verschiedenen KDBSen semantisch äquivalente Objekt-Typen existieren können, für deren Integration virtuelle Klassen angeboten werden, bei deren Definition mehrere Select-Anweisungen zulässig sind. Die Möglichkeit, daß äquivalente Objekt-Typen eine extensionale Überlappung aufweisen können, wird allerdings nicht betrachtet. Konzepte, die speziell der Identifikation und Verwaltung semantisch äquivalenter Objekte dienen, sind in der aktuellen Version des MDBSs nicht vorhanden.

Es können aber bestimmte allgemeine Konzepte, die UniSQL/M anbietet, genutzt werden, um eine Identifikation und (virtuelle) Integration sowie die Verwaltung äquivalenter Objekte zu realisieren.

Zum Beispiel ist es möglich, äquivalente Objekte mit Hilfe von Verbundsichten zu erkennen und zu integrieren. Hierzu wird eine outer join-Operation über die betreffenden Objekt-Typen realisiert. Als Kriterium zur Erkennung äquivalenter Objekte können bei dieser Strategie nur einfache Vergleiche bestimmter gemeinsamer Attribute der beiden betrachteten Objekt-Typen, verkörpert in der join-Bedingung, herangezogen werden. Insbesondere für den Fall der Existenz eines gemeinsamen Primärschlüssels ist diese Strategie anwendbar.

Allerdings kann in UniSQL/M auf Verbundsichten nur lesend zugegriffen werden. Hierdurch entsteht eine erhebliche Einschränkung bei der Nutzung der integrierten Datenbank.

Eine weitere Möglichkeit zur Erkennung und Verwaltung äquivalenter Objekte besteht in der Nutzung benutzerdefinierter Methoden, die durch UniSQL/M angeboten werden. Hierbei wird einem integrierten Objekt-Typ eine Methode für die Erkennung äquivalenter Objekte zugeordnet. In dieser Methode ist ein Verfahren zur Erkennung der äquivalenten Objekte der betreffenden lokalen Objekt-Typen implementiert. Als Ausgabe liefert diese Methode Paare von OIDen äquivalenter Objekte. Diese Informationen werden in einer zusätzlichen Klasse materialisiert und in die join-Bedingung der Definition der virtuellen Klasse, die den betreffenden integrierten Objekt-Typ darstellt, einbezogen. Die Identifikations-Methode muß dann in geeigneten Intervallen zur Aktualisierung der match-Informationen angestoßen werden.

Mit Hilfe der Nutzung einer Methode zur Identifikation der äquivalenten Objekte lassen sich auch komplexere Kriterien für die Äquivalenz von Objekten, wie zum Beispiel das vorgestellte Konzept der kompatiblen Attributpaare, implementieren. Allerdings wird auch in diesem Fall eine Verbundsicht als integrierter Objekt-Typ verwendet. Somit ist auch hier nur ein lesender Zugriff auf die integrierten Objekte möglich.

Um auch schreibend (update, delete, insert) auf eine Verbundsicht zugreifen zu können, sind zusätzliche Methoden notwendig, die diese Operationen realisieren.

7.4 Einbindung von KDBSen in UniSQL/M

UniSQL/M bietet als MDBMS die Möglichkeit, KDBS beziehungsweise lokale Datenbanksysteme in ein MDBS einzubinden. In der aktuellen Version 3.5 werden als KDBSe die RDBSe Oracle, Sybase, Informix und Microsoft SQL Server unterstützt . Weiterhin können UniSQL/X oder UniSQL/M als KDBSe auftreten. Mit UniSQL/X wird hierbei die zentralisierte Version des ORDBSs der Firma UniSQL Inc. bezeichnet. Sie bietet alle Möglichkeiten, die auch UniSQL/M bietet, jedoch keine Multidatenbankfähigkeit.

In Abbildung 7-1 wird ein Beispiel für den Aufbau eines MDBSs durch die Einbindung verschiedener KDBSe auf der Basis von UniSQL/M gegeben. Dadurch, daß auch UniSQL/M selbst als KDBS auftreten kann, sind auch mehrstufige Einbindungen von KDBSen möglich. Hierdurch ist eine sehr flexible Gestaltung des MDBSs möglich. Jedoch ist zu beachten, daß der Graph der Einbindun-

gen von KDBSen keine Zyklen enthalten darf. Weitere Informationen sind [Uni96b] zu entnehmen.

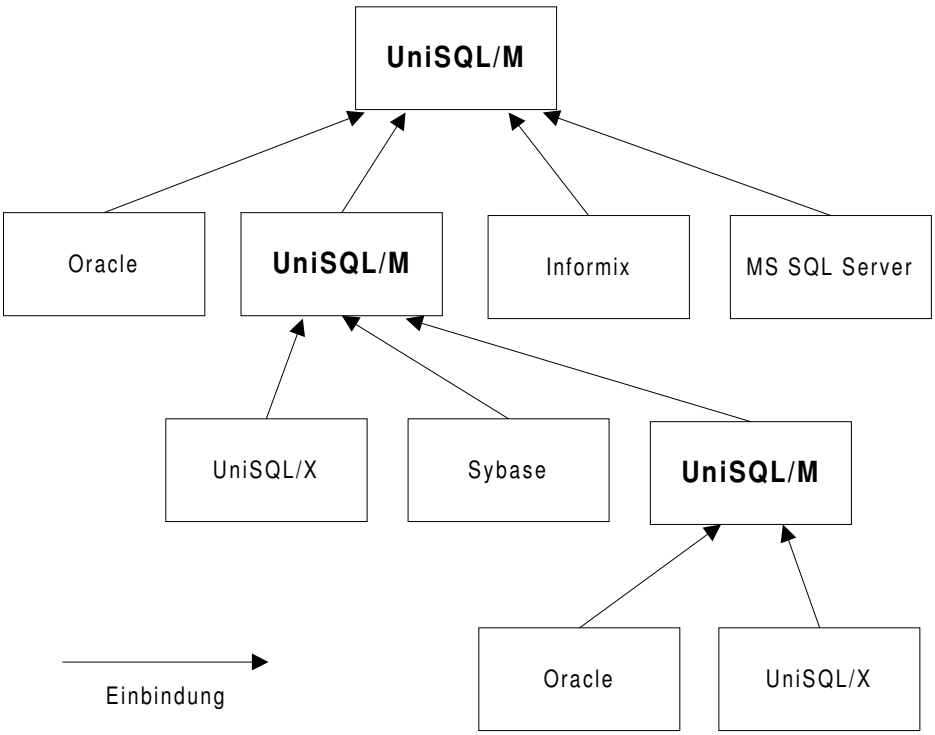


Abbildung 7-1: Beispiel für die Einbindung von KDBSen

Die Einbindung der KDBS besitzt im Fall von UniSQL/M föderativen Charakter. Dies bedeutet, daß die Komponentensysteme Autonomie besitzen. Lokale Applikationen können daher auch weiterhin auf die lokalen Systeme zugreifen.

Dem Grad der Einbindung nach findet eine enge Kopplung der KDBS statt, da globalen Applikationen ein integriertes Schema zur Verfügung gestellt wird.

7.5 Aufbau einer UniSQL/M-Datenbank

In diesem Unterabschnitt soll die Vorgehensweise zur (virtuellen) Einbindung von lokalen Datenbeständen in eine globale integrierte UniSQL/M-Datenbank beschrieben werden. Voraussetzung für eine Einbindung sind Informationen darüber, welche Teile der Datenbestände der lokalen Systeme in die globale Datenbank einbezogen werden sollen und eine abgeschlossene Schemaintegration der betreffenden lokalen Subschemata zu einem integrierten Gesamtschema.

7.5.1 Registrierung der lokalen Datenbanken

Als erster Schritt müssen die Datenbanken der lokalen Systeme in der globalen UniSQL/M-Datenbank angemeldet werden. Dieser Vorgang wird als Registrierung bezeichnet und erfolgt mit Hilfe des Kommandos **`register ldb`**

ldb_name. Durch die Registrierung werden Name, Ort, Typ des KDBSs (zum Beispiel Oracle) und Nutzerinformationen für die Anmeldung im betreffenden lokalen System in der globalen Datenbank bekannt gemacht.

7.5.2 Erstellung von Proxies lokaler Objekt-Typen

Nachdem die lokalen Datenbanken registriert wurden, können in einem zweiten Schritt Stellvertreter-Objekt-Typen, sogenannte Proxies, lokaler Objekt-Typen, die in die globale Datenbank einbezogen werden sollen, in der globalen Datenbank angelegt werden. Die Proxies werden mit Hilfe des Kommandos **`create proxy`** *proxy_name* **`on ldb`** *ldb_name* erstellt. Sie bilden die Brücke aus den lokalen Systemen in das globale UniSQL/M-System. Sie realisieren dabei eine Übersetzung der jeweiligen lokalen Objekt-Typen aus dem lokalen Datenmodell des KDBSs in das globale objektrelationale Datenbankmodell von UniSQL/M. Es findet somit eine Schematranslation statt.

Darüber hinaus bietet die Definition der Proxies auch die Möglichkeit, den Anteil eines lokalen Objekt-Typs, der in der globalen Datenbank verfügbar sein soll (Exportanteil) selektiv und projektiv zu spezifizieren. Für KDBSe, die keine OIDen unterstützen kann mit Hilfe der Klausel **`object_id`** (*attribute_name* [{, *attribute_name* } ...]) innerhalb der Proxy-Definition die Erzeugung globaler OIDen auf der Basis einer Menge von Attributwerten (zum Beispiel Primärschlüssel) der lokalen Objekte veranlaßt werden.

Da durch die Proxy-Definition neben der Übersetzung der lokalen Schemakonstrukte in das globale Datenmodell auch der Anteil der global verfügbaren lokalen Daten festgelegt wird, entspricht die Menge aller Proxies eines lokalen Systems einem Komponenten- und Exportschema in der fünfschichtigen Schemaarchitektur für föderierte Systeme entsprechend [SL90].

Die Menge aller Proxies wird in [Uni96c] als Tranformationsschema bezeichnet.

7.5.3 Das integrierte Gesamtschema

Die zu exportierenden Subschemata der lokalen Datenbanken sind nun in Gestalt der Menge von Proxies lokaler Objekt-Typen global verfügbar. Allerdings muß aus diesen isolierten lokalen Schemata noch ein integriertes Gesamtschema erstellt werden. Dies erfolgt durch die Definition virtueller Klassen in der globalen Datenbank. Virtuelle Klassen werden mit dem Kommando **`create vclass`** *vclass_name* definiert.

Die Integration semantisch äquivalenter lokaler Objekt-Typen in eine globale virtuelle Klasse wird dadurch erreicht, daß es möglich ist, mehrere Select-Anweisungen bei der Definition einer virtuellen Klasse anzugeben, welche durch eine **`union all`**-Klausel (Bildung der Vereinigungsmenge) verbunden sind.

Beim Aufbau des integrierten Gesamtschemas muß die Identifikation der möglicherweise vorhandenen semantisch äquivalenten Objekte berücksichtigt werden. Wie bereits angesprochen stehen hierfür in UniSQL/M derzeit aber nur

begrenzte Möglichkeiten in Form von Verbundsichten und benutzerdefinierten Methoden zur Verfügung.

Wenn globalen Applikationen Informationen zur Verfügung stehen sollen, die über die in den lokalen Systemen abgelegten Datenbestände hinausgehen, ist es möglich, die globale Datenbank um weitere Klassen zur Verwaltung dieser Informationen zu erweitern. In Form derartiger zusätzlicher Klassen lassen sich dann auch Informationen darüber, welche Objekte semantisch äquivalent sind, verwalten.

Die Menge aller virtuellen Klassen bildet das integrierte Gesamtschema der globalen Datenbank.

7.5.4 Externe Schemata

Für einzelne Nutzergruppen oder Applikationen kann mit Hilfe zusätzlicher virtueller Klassen, die auf das integrierte Gesamtschema aufsetzen ein nutzerspezifischer Datenbankzugang in Form von externen Schemata realisiert werden. Diese virtuellen Klassen werden ebenfalls mit Hilfe des Kommandos **`create vclass`** *vclass_name* angelegt.

Die dargelegten Ausführungen bezüglich des Aufbaus einer UniSQL/M-Datenbank werden durch Abbildung 7-2 noch einmal zusammengefaßt und graphisch veranschaulicht:

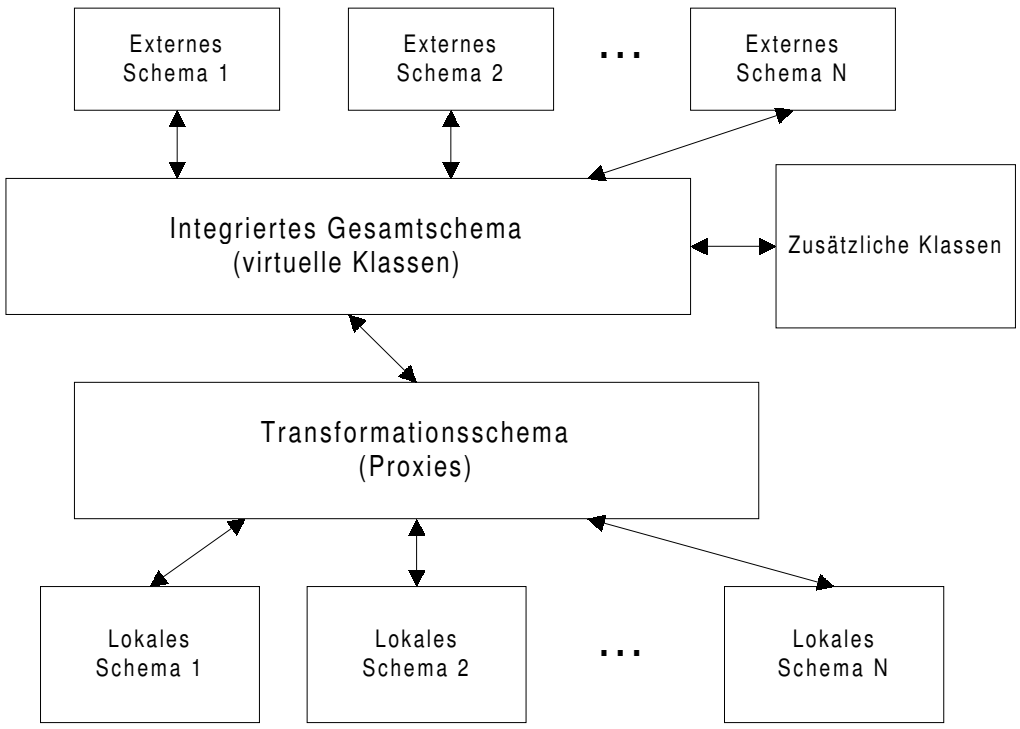


Abbildung 7-2: Schemaarchitektur von UniSQL/M

8 Beispielanwendung eines Multidatenbanksystems

In diesem Abschnitt soll an einem praktischen Beispiel die Vorgehensweise beim Aufbau einer UniSQL/M-Multidatenbank veranschaulicht werden. Dabei soll insbesondere dargestellt werden, wie die semantisch äquivalenten Objekte der KDBSe, die infolge der extensionalen Überlappung der lokalen Datenbanken auftreten, erkannt und integriert werden können.

Ausgangspunkt für das zu realisierende Anwendungsbeispiel sind zwei heterogene Datenbanken eines Unternehmens. Die erste der beiden Datenbanken wird von der Vertriebsabteilung des Unternehmens verwendet und enthält Informationen über Transportmaschinen, die von dem Unternehmen produziert wurden und nun verkauft werden sollen. Diese Datenbank wurde mit Hilfe einer relationalen Oracle-Datenbank implementiert.

Die zweite Datenbank enthält Daten der Fertigungsabteilung über Rollenförderer, die im Unternehmen entwickelt und produziert werden. Sie wurde in einer objektrelationalen UniSQL/X-Datenbank realisiert.

Zwischen diesen beiden Datenbanken tritt eine extensionale Überlappung auf. Dies bedeutet, daß es Maschinen gibt, die sowohl in der Datenbank der Fertigungsabteilung als auch in der Datenbank der Vertriebsabteilung gespeichert sind. Dabei sind in den jeweiligen Datenbanken genau die Daten über eine bestimmte Maschine gespeichert, die für die betreffende Abteilung des Unternehmens relevant sind.

Ziel dieses Abschnittes ist es zu zeigen, wie die beiden beschriebenen heterogenen Datenbanken des Unternehmens mit Hilfe des MDBSs UniSQL/M in eine globale Datenbank (virtuell) integriert werden können, um eine unternehmesweite Analyse der wirtschaftlichen Situation des Unternehmens auf der Basis von OLAP-Tools (OnLine Analytical Processing Tools) zu ermöglichen. Damit ein logisch korrekter globaler Datenbestand realisiert werden kann, muß hierbei das Problem der Identifikation und Integration der vorhandenen semantisch äquivalenten Objekte für die betrachteten Datenbanken gelöst werden.

Die Datenbankschemata der beiden lokalen Datenbanken sowie das integrierte Gesamtschema wurden [CHJ+96] entnommen.

In Abbildung 8-1 wird die Struktur des MDBSs, welches in unserem Beispiel-Unternehmen aufgebaut werden soll, schematisch veranschaulicht.

In den folgenden Unterabschnitten sollen zunächst die Schemata der KDBSe vorgestellt werden. Daran anschließend wird dann auf das integrierte Gesamtschema eingegangen. Eine Diskussion darüber, wie in diesem Fall eine Erkennung und Integration der semantisch äquivalenten Objekte erfolgen kann beschließt diesen Abschnitt.

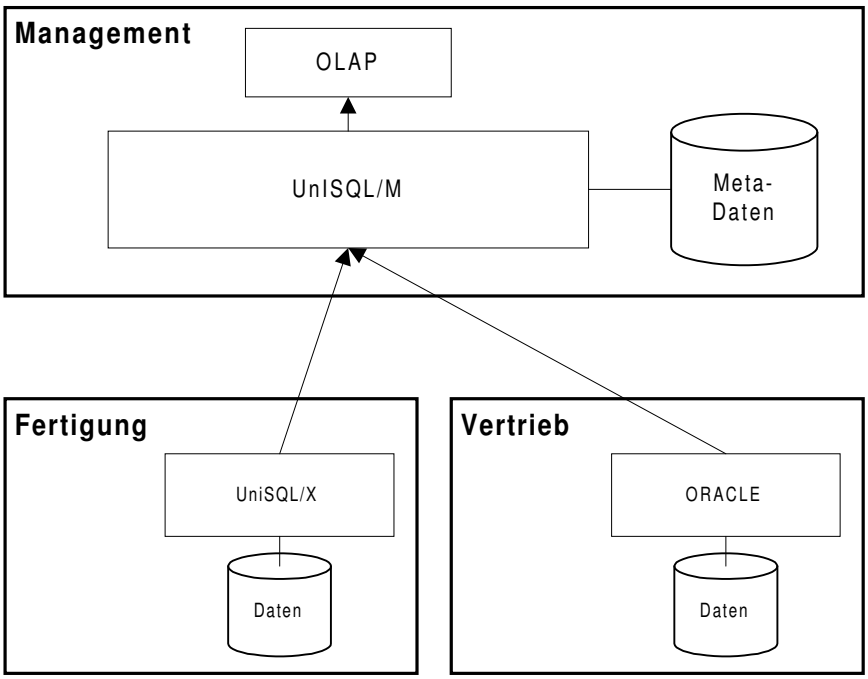


Abbildung 8-1: Struktur des MDBSs im Beispiel-Unternehmen

8.1 Die Schemata der KDBSe

Das Schema der relationalen Datenbank der Vertriebsabteilung besteht aus einer einzelnen Tabelle, welche Informationen über zu verkaufende Transportmaschinen (`transport_machine`, TM) enthält. Die Struktur der Tabelle geht aus folgendem Oracle-SQL-DDL-Statement hervor:

```
create table transport_machine (
  product_no           char(6) primary key,
  product_name         varchar2(40),
  price                number(10,2),
  delivery_period      number(3),
  machine_class        varchar2(30));
```

Der Primärschlüssel für die zu verwaltenden Maschinenobjekte ist ihre 6-stellige Produktnummer (`product_no`). Weiterhin sind neben dem Produktnamen (`product_name`) typische Vertriebsinformationen über die Produkte, wie Preis (`price`) und Lieferzeit (`delivery_period`) erfaßt. Außerdem ist vermerkt, um was für eine Art von Maschine es sich handelt (`machine_class`). Gültige Zeichenketten für dieses Attribut sind ‘roller conveyor’ (Rollenförderer), ‘belt conveyor’ (Förderband), ‘crane’ (Kran) und ‘fork-lift truck’ (Gabelstapler).

In der Datenbank der Fertigungsabteilung sind Informationen über Rollenförderer (`roller_conveyor`, RC) und motorgetriebene Rollenförderer (`driven_roller_conveyor`, DRC) in Form zweier, in einer Spezialisierungsbeziehung stehender, Klassen gespeichert. Das Schema dieser Datenbank in SQL/X-DDL lautet:

```
create class roller_conveyor (
  product_no          char(6) not null unique,
  product_name        varchar(20),
  length              integer,
  width               integer,
  project_state       varchar(15))

create class driven_roller_conveyor
  as subclass of roller_conveyor (
  max_speed           float,
  motor_voltages      set(float))
```

Die Klasse `roller_conveyor` besitzt eine 6-stellige Produktnummer (`product_no`) als Primärschlüssel. Weiterhin sind neben dem Namen (`product_name`) Produktdaten, wie Länge (`length`) und Breite (`width`) einer Maschine sowie ein Projektstatus (`project_state`) vermerkt. Zulässige Projektstati sind ‘in_design’ (in Entwicklung), ‘in_construction’ (in Bau) und ‘completed’ (fertiggestellt).

Die Klasse `driven_roller_conveyor` wird von der Klasse `roller_conveyor` abgeleitet (spezialisiert). Sie erbt alle Attribute (einschließlich des Primärschlüssel-Constraints für die Produktnummer) der Klasse `roller_conveyor`, besitzt jedoch zusätzliche Eigenschaften, wie eine Maximalgeschwindigkeit (`max_speed`) und eine Menge möglicher Motorspannungen (`motor_voltages`), die speziell für motorbetriebene Rollenförderer interessant sind.

Die extensionalen Beziehungen der drei Objekt-Typen der beiden lokalen Datenbanken wird durch Abbildung 8-2 verdeutlicht:

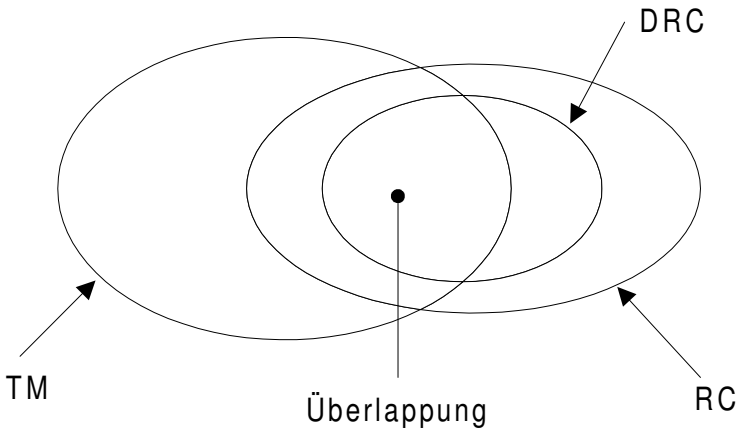


Abbildung 8-2: Überlappende Extensionen der Beispiel-Datenbanken

Die Überlappung der Extensionen der Datenbanken unseres Beispiel-Unternehmens entsteht dadurch, daß es Transportmaschinen in der Vertriebsabteilung gibt, die in der Fertigungsabteilung für Rollenförderer hergestellt wurden. Allerdings existieren in beiden Datenbanken Maschinen-Objekte, die in der jeweils anderen Datenbank nicht vorhanden sind. Dies sind für die Vertriebsabteilung zum Beispiel Transportmaschinen, die keine Rollenförderer sind. Für die Fertigungsdatenbank könnten dies andererseits Rollenförderer sein, die noch nicht verkauft werden sollen.

Die Erkennung der semantisch äquivalenten Objekte der beiden Datenbanken kann im vorliegenden Beispiel über die unternehmensweit einheitliche Produktnummer der Transportmaschinen erfolgen.

8.2 Das integrierte Gesamtschema

Nachdem im letzten Unterabschnitt die lokalen Schemata der beiden KDBSe vorgestellt wurden, soll nun das Schema der integrierten (globalen) Datenbank betrachtet werden. In Abbildung 8-3 sind die globalen (virtuellen) Klassen mit ihren Spezialisierungsbeziehungen dargestellt:

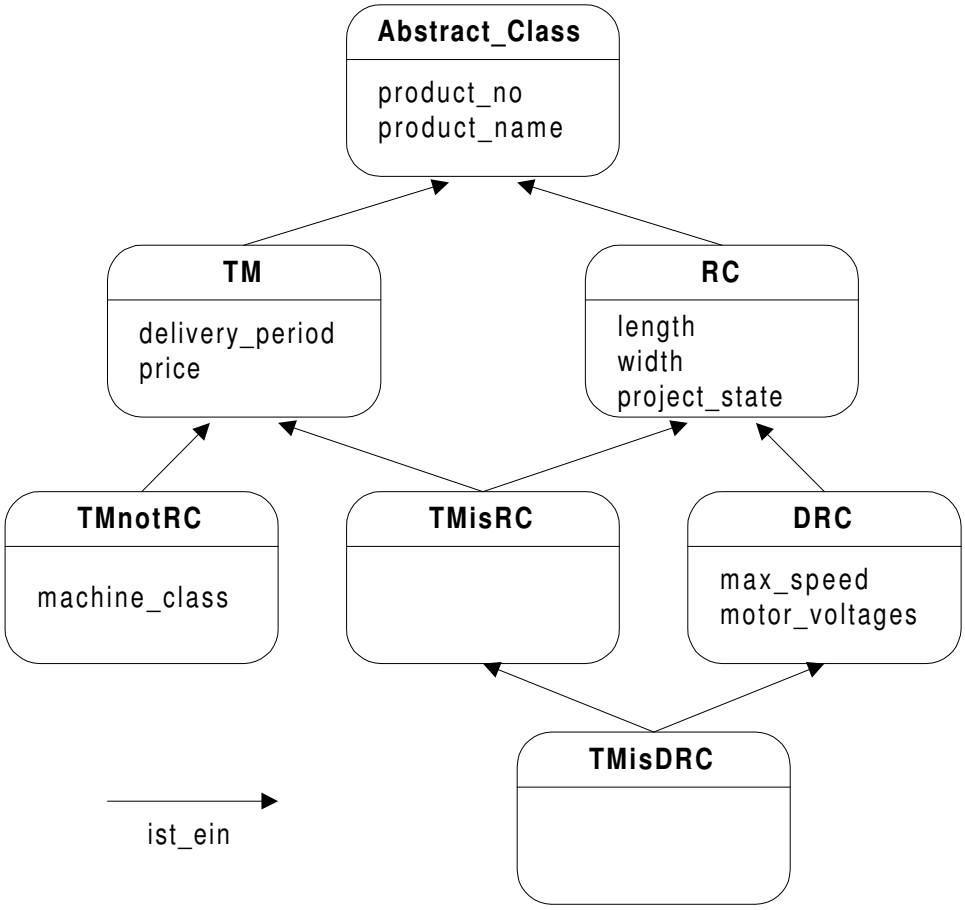


Abbildung 8-3: Klassenhierarchie des integrierten Schemas

Die Integration der lokalen Ausgangsschemata zu dem in Abbildung 8-3 dargestellten integrierten Gesamtschema kann [CHJ+96] entnommen werden.

Die allgemeinste Klasse des Gesamtschemas ist `Abstract_Class`. Hierbei handelt es sich, wie schon der Name vermuten läßt, um eine abstrakte Klasse. Dies bedeutet, daß ihre flache Extension leer ist. Von dieser Klasse wird die Klasse `TM` abgeleitet, deren flache Extension alle Rollenförderer enthält, die nicht in der Fertigungsdatenbank enthalten sind. Ihre tiefe Extension repräsentiert die Menge aller Transportmaschinen der Vertriebsdatenbank.

Die Klasse `TM` wird weiterhin zu `TMnotRC` und `TMisRC` spezialisiert. Die Klasse `TMnotRC` repräsentiert alle Transportmaschinen, die keine Rollenförderer sind, wogegen die Klasse `TMisRC` gerade diese abbildet. Es sei darauf hingewiesen, daß die Klasse `TMisRC` neben `TM` auch die Klasse `RC` als Superklasse besitzt (Mehrfachspezialisierung).

Die Klasse `RC` geht aus der Klasse `Abstract_Class` hervor. Sie repräsentiert die Menge aller Rollenförderer der Fertigungsdatenbank und besitzt neben

TMisRC die Klasse DRC der motorbetriebenen Rollenförderer der Fertigungsdatenbank als Subklassen.

Schließlich wird aus den Klassen TMisRC und DRC durch Mehrfachspezialisierung die Klasse TMisDRC der motorgetriebenen Rollenförderer, die auch in der Vertriebsdatenbank enthalten sind, abgeleitet.

Betrachtet man noch einmal die extensionale Überlappung der beiden Datenbanken in Abbildung 8-2, so kann man leicht erkennen, daß der überlappende Anteil gerade durch die tiefe Extension der Klasse TMisRC beziehungsweise durch die Vereinigung der flachen Extensionen von TMisRC und TMisDRC verkörpert wird.

Aus diesem Grund muß für diese beiden Klassen eine Identifikation der semantisch äquivalenten Objekte und deren Integration zu einem globalen Objekt realisiert werden.

In der globalen Datenbank werden die soeben erläuterten Klassen als virtuelle UniSQL/M-Klassen angelegt. Sie bilden eine globale integrierte Sicht auf die Datenbestände der KDBSe, die den Gesamtdatenbestand der beiden lokalen Systeme globalen Applikationen zur Verfügung stellt.

8.3 Registrierung der KDBSe

Wie bereits im letzten Abschnitt dargestellt, besteht der erste Schritt zur Einbindung von KDBSen darin, die betreffenden Datenbanken in der globalen Datenbank zu registrieren. Für das Anwendungsbeispiel lauten die hierzu erforderlichen Anweisungen für beide KDBSe:

```
register ldb vertrieb

  name 'first'
  type 'oracle'
  host 'pluto'
  user 'scott'
  password 'tiger'
;

register ldb fertigung

  name 'tom'
  type 'unisql'
  host 'pluto'
  user 'public'
  password ''
;
```

In einer Registrierungsanweisung wird zunächst der Name festgelegt, mit dem die lokale Datenbank global angesprochen werden soll (zum Beispiel Vertrieb). Außerdem enthält die Anweisung Informationen, die den Aufbau einer Verbindung zum betreffenden KDBS ermöglichen. Diese Informationen beinhalten den Na-

men der lokalen Datenbank im lokalen System (zum Beispiel first), ihren Typ (zum Beispiel oracle), den Host, auf dem der Zugriff realisiert werden soll (zum Beispiel pluto) und den Datenbank-Account, festgelegt durch Nutzerbezeichnung und Paßwort, der für den Zugriff aus dem MDBS heraus verwendet werden soll.

8.4 Erstellen der Proxies

Mit Hilfe von Proxies werden lokale Objekt-Typen in die globale Datenbank integriert. Sie überführen die betreffenden lokalen Objekt-Typen in das Datenmodell der globalen Datenbank und wählen gleichzeitig den Anteil des lokalen Objekt-Typs selektiv und projektiv aus. Man kann sagen, Proxies „vertreten“ lokale Objekt-Typen in der globalen Datenbank.

Zur Verdeutlichung wurde unten die Definition des Proxys `roller_conveyor_proxy` aus dem Anwendungsbeispiel angeführt, durch den der lokale Objekt-Typ `roller_conveyor` der Fertigungsdatenbank global verfügbar wird. Ein vollständiges Verzeichnis aller Proxy-Definitionen des Anwendungsbeispiels ist im Anhang zu finden.

```
create proxy roller_conveyor_proxy
  on ldb fertigung (

  product_no char(6),
  product_name varchar(20),
  length integer,
  width integer,
  project_state varchar(15))

  as
  select product_no,product_name,length,
    width,project_state
  from roller_conveyor
;
```

Neben dem Namen des Proxys wird in der Definition die lokale Datenbank angegeben, in der sich der betreffende lokale Objekt-Typ befindet (hier Fertigung). Außerdem erfolgt eine Auflistung der Attribute des Proxys mit ihren (globalen) Datentypen.

Die Proxy-Definition wird von einer Select-Anweisung abgeschlossen. Durch sie lassen sich die lokalen Datenbankobjekte, die global „sichtbar“ sein sollen projektiv und selektiv auswählen. Die lokalen Attribute in der Select-Anweisung korrespondieren dabei entsprechend ihrer Reihenfolge mit den Attributen des Proxys.

Für unser Anwendungsbeispiel erfolgt durch die obige Anweisung die Auswahl aller lokalen Objekte mit allen lokalen Attributen. Der lokale Objekt-Typ wird also vollständig exportiert.

8.5 Anlegen der virtuellen Klassen

Aufbauend auf den Proxies der lokalen Objekt-Typen läßt sich nun das globale integrierte Schema in Form von virtuellen Klassen definieren. Dabei muß auch die Integration von lokalen Objekt-Typen mit überlappenden Extensionen erfolgen.

Das Anlegen von virtuellen Klassen für unser Anwendungsbeispiel soll mit Hilfe der virtuellen Klasse `TMisDRC` verdeutlicht werden. Sie enthält alle motorbetriebenen Rollenförderer, die auch in der Vertriebsdatenbank gespeichert sind und somit einen Teil der extensionalen Überlappung der lokalen Datenbanken.

```
create vclass tmisdrc
  as subclass of tmisrc,drc

  as
  select t.product_no,t.product_name,
    t.delivery_period,t.price,
    d.length,d.width,d.project_state,
    d.max_speed,d.motor_voltages
  from transport_machine_proxy t,
    driven_roller_coveyor_proxy d
  where t.product_no=d.product_no
;
```

Die Definition der virtuellen Klasse wird durch die Angabe ihres Namens und der möglicherweise vorhandenen virtuellen Superklassen eingeleitet. Im Beispiel erbt die virtuelle Klasse `TMisDRC` von den virtuellen Klassen `TMisRC` (Rollenförderer, die auch in der Vertriebsdatenbank vorhanden sind) und `DRC` (motorbetriebene Rollenförderer). Es findet also eine Mehrfachvererbung statt. Die neue virtuelle Klasse besitzt allerdings keine zusätzlichen Attribute, sondern nur jene, die von den virtuellen Superklassen geerbt werden.

Die Auflösung von Attributkonflikten, die durch die Mehrfachvererbung entstehen ist in diesem Fall nicht notwendig, da alle betreffenden Attribute (zum Beispiel `product_no`) bereits von einer gemeinsamen virtuellen Superklasse der virtuellen Superklassen der neuen virtuellen Klasse geerbt wurden und in UniSQL/M in einem derartigen Fall die betreffenden Attribute generell nur einmal vererbt werden.

Die (virtuelle) Extension der neuen virtuellen Klasse wird mit Hilfe einer (oder mehrerer) Select-Anweisungen bestimmt. Die Integration der Daten beider KDBSe wird hier dadurch erreicht, daß Attribute aus beiden Proxies der lokalen Objekt-Typen selektiert und in der neuen virtuellen Klasse vereinigt werden. Dabei erfolgt die Identifikation der semantisch äquivalenten Objekt durch einen Verbund über die Produktnummer.

Unter Verwendung des zusätzlichen Wissens aus der Schemaintegration, daß nur solche Transportmaschinen der Vertriebsdatenbank auch in der Fertigungsdatenbank enthalten sein können, die als Maschinenklasse (`machine_class`) den Wert `'roller_conveyor'` besitzen, ließe sich der Identifikationsprozeß noch

effizienter gestalten. Hierzu könnte die Where-Klausel der Select-Anweisung durch die zusätzliche Bedingung `t.machine_class = 'roller_conveyor'` konjunktiv erweitert werden. Damit ließe sich ein beträchtlicher Anteil an Transportmaschinen der Vertriebsdatenbank von der vergleichsweise aufwendigen Berechnung des Verbundes ausschließen.

Eine vollständige Auflistung der Definitionen aller virtuellen Klassen des Anwendungsbeispiels kann dem Anhang entnommen werden.

9 Realisierungsmöglichkeiten für Identifikationsverfahren in UniSQL/M

Bezugnehmend auf die in Abschnitt 5 vorgestellten Konzepte zur Identifikation semantisch äquivalenter Datenbankobjekte, sollen in diesem Abschnitt Möglichkeiten aufgezeigt werden, wie sich diese Ansätze in dem objektrelationalen MDBS UniSQL/M realisieren lassen.

9.1 Verwendung einer Lookup-Tabelle

Soll eine Lookup-Tabelle verwendet werden, um semantisch äquivalente Objekte zweier lokaler Datenbanken zu identifizieren, so kann dies durch eine zusätzliche Klasse in der globalen Datenbank erreicht werden, die die OIDen der semantisch äquivalenten Objekte enthält.

Für zwei lokale Objekt-Typen A und B mit extensionaler Überlappung, deren globale Proxies a und b heißen, könnte diese zusätzliche Klasse, dargestellt als SQL/X-Anweisung, folgendermaßen aufgebaut sein:

```
create class lookup_a_b (
  a_ref a not null,
  b_ref b not null,
  unique(a_ref,b_ref))
```

Die dargestellte Klasse enthält die Attribute `a_ref` vom Typ `a` und `b_ref` vom Typ `b`. Das heißt, es handelt sich um Referenzattribute für Objekte der Proxies a und b. Ihre Werte sind OIDen der Objekte aus den betreffenden Proxies.

Damit die Informationen der Lookup-Tabelle auch tatsächlich zur Integration der semantisch äquivalenten lokalen Objekte zu einem globalen Objekt verwendet werden, muß die Definition der globalen integrierten (virtuellen) Klasse angepaßt werden. Die folgende SQL/X-Anweisung zur Definition der globalen integrierten (virtuellen) Klasse soll dies an einem Beispiel verdeutlichen:

```
create vclass integration_of_a_and_b (
  att1 att1_type,
  att2 att2_type,
  att3 att3_type,
  att4 att4_type)

as
select local_att1, local_att2, na, na from a
where a.identity not in
(select a_ref from lookup_a_b)

union all
select na, na, local_att3, local_att4 from b
```

```
where b.identity not in
(select b_ref from lookup_a_b)

union all
select a.loacal_att1, a.local_att2, b.local_att3,
b.local_att4
from a, b, lookup_a_b l
where a.identity=l.a_ref and b.identity=l.b_ref
```

In diesem Beispiel wurde davon ausgegangen, daß die integrierte (virtuelle) Klasse insgesamt vier Attribute, zwei aus a (`att1` und `att2`) und zwei aus b (`att3` und `att4`), enthält. Die (virtuelle) Extension der (virtuellen) Klasse wird durch drei Select-Anweisungen definiert. Die ersten beiden Anweisungen selektieren die Objekte aus a und b, die nicht Bestandteil der extensionalen Überlappung sind und deren OIDen somit auch nicht in der Lookup-Tabelle vorkommen. Die Angabe `na` (Nullwert) wird für globale Attribute verwendet, die im betreffenden lokalen Objekt-Typ nicht vorhanden sind.

Mit Hilfe der dritten Select-Anweisung werden die semantisch äquivalenten Objekte unter Nutzung der Lookup-Tabelle identifiziert und ihre Eigenschaften in einem gemeinsamen globalen (virtuellen) Objekt vereinigt. Hierzu wird ein Verbund der Informationen der Proxies mit den Informationen der Lookup-Tabelle berechnet. Nur die Objekte sind äquivalent, deren OIDen als Paar in der Lookup-Tabelle verzeichnet sind.

9.2 Gemeinsamer Schlüssel (common key)

Existiert ein gemeinsamer Schlüssel, so kann eine Identifikation und Integration äquivalenter Objekte relativ einfach mit Hilfe einer globalen Verbundsicht erfolgen. Dabei werden innerhalb der Where-Klausel der Select-Anweisung die Schlüsselattribute (`key_att`) miteinander verglichen.

Die folgende SQL/X-Anweisung verdeutlicht diese Situation am Beispiel aus dem vorangegangenen Unterabschnitt:

```
create vclass integration_of_a_and_b
  key  key_type,
  att2 att2_type,
  att3 att3_type)

as
select a.key_att, a.local_att2, b.local_att3
from a, b
where a.key_att=b.key_att
```

Der Nachteil der Verwendung einer Verbundsicht besteht darin, daß keine schreibenden Zugriffe zulässig sind. Ist es aber notwendig derartige Zugriffe zu realisieren, so können hierfür die betreffenden Proxies direkt angesprochen werden. Je-

doch bedeutet dies eine Verletzung der Verteilungstransparenz, die in der globalen Datenbank angestrebt wird.

9.3 Erweiterter Schlüssel (extended key)

Wie bereits in Abschnitt 5 zum Konzept der Verwendung eines erweiterten Schlüssels ausgeführt wurde, beruht die Erkennung der semantisch äquivalenten Objekte hierbei auf dem Vergleich eines erweiterten Primärschlüssels. Fehlende Eigenschaften der betreffenden Objekt-Typen, die Bestandteil des erweiterten Schlüssels sind, werden mit Hilfe von Zusatzwissen in Form spezieller Regeln (ILFDs) berechnet.

Hinsichtlich der Realisierung dieses Verfahrens in UniSQL/M kann man von der Idee gemäß [LSP+93] ausgehen, daß sich ILFDs, die sich auf gleiche Eigenschaften beziehen in Form von Tabellen beziehungsweise Klassen in einer Datenbank abbilden lassen.

Zum Beispiel läßt sich die Menge aller ILFDs der Form E.PLZ=plz → E.Ort='ort', die ausgehend von der Postleitzahl eines Realweltobjektes (aus einer Menge von Realweltobjekten E) die Eigenschaft Ort des Realweltobjektes bestimmen, als Klasse ILFD_PLZ_ORT darstellen. Jedes Objekt dieser Klasse entspricht genau einer konkreten ILFD. Die Definition dieser Klasse als SQL/X-Anweisung lautet:

```
create class ilfd_plz_ort (
  plz number(5) not null,
  ort varchar(50) not null,
  unique(plz))
```

Analog wird nun für jedes zu berechnende Attribut beider zu untersuchender Objekt-Typen eine Klasse angelegt, die das notwendige Wissen zur Berechnung des betreffenden Attributes enthält. Alternativ lassen sich möglicherweise auch bestehende Datenbanken (zum Beispiel Postleitzahlen-Verzeichnis der Telekom) nutzen, indem sie als weitere Komponenten in das MDBS einbezogen werden.

Durch eine Folge von relationalen Operationen (für Details siehe [LSP+93]) kann nun eine Tabelle (sogenannte matching-Tabelle) berechnet werden, die die OIDen der semantisch äquivalenten Objekte der betrachteten Objekt-Typen paarweise enthält. Sie entspricht der im Unterabschnitt 9.1 dargestellten Lookup-Tabelle. Während jedoch dort durch eine Lookup-Tabelle „manuell" festlegt wird, welche Objekte semantisch äquivalent sind, soll eine matching-Tabelle nur verhindern, daß der Aufwand zur automatischen Identifikation der äquivalenten Objekte bei jeder Anfrage neu entsteht.

Bei der Berechnung der matching-Tabelle wird so vorgegangen, daß zunächst die Attribute, die den beiden betrachteten Objekt-Typen, zum Beispiel R und S, hinsichtlich des erweiterten Schlüssels fehlen, unter Nutzung der ILFD-Klassen abgeleitet werden. Ergebnis sind zwei temporäre Klassen R' und S', die die ursprünglichen Schlüsselattribute und die fehlenden Attribute des erweiterten

Schlüssels enthalten. Die matching-Tabelle kann nun durch einen Verbund über den erweiterten Schlüssel berechnet werden.

Da sich die (virtuelle) Extension der integrierten (virtuellen) Klasse auf die Informationen der matching-Tabelle stützt, gleicht ihre Struktur der in Unterabschnitt 9.1 dargestellten integrierten (virtuellen) Klasse. Es wird jedoch eine Methode, die die Ausführung der beschriebenen Operationen zur Berechnung der matching-Tabelle realisiert, hinzugefügt. Diese Methode muß in geeigneten Intervallen aufgerufen werden, um eine hinreichende Aktualität der Informationen in der matching-Tabelle zu gewährleisten.

9.4 Kompatible Attributpaare

Als Voraussetzung für die Realisierung einer Identifikation semantisch äquivalenter Datenbankobjekte auf der Basis gemeinsamer Attribute in dem MDBS UniSQL/M ist eine Erweiterung der Metadaten über die beiden betrachteten Objekt-Typen notwendig. Diese Metadaten beinhalten Informationen darüber, welche Attribute kompatibel sind, welchem Datentyp sie angehören, welches Gewicht sie besitzen und ob sie dominant sind. Weiterhin werden Informationen darüber benötigt, wie verläßlich die Daten der beiden beteiligten lokalen Datenbanken sind, wie groß die zulässigen Abweichungen für nicht dominante Attribute (maximum difference distance, MDD) sind und ab welcher Grenzwahrscheinlichkeit α Objekte als semantisch äquivalent gelten.

Für die Verwaltung der genannten Metadaten wird eine zusätzliche Klasse in der globalen Datenbank vorgeschlagen, die für jedes vorhandene kompatible Attributpaar ein Objekt enthält. Hierdurch wird eine Änderung bestimmter Metadaten, zum Beispiel der Attributgewichte, erleichtert. Die Grenzwahrscheinlichkeit α und die Verläßlichkeitskoeffizienten der lokalen Datenbanken werden als Klassenattribute implementiert. Damit hat die vorgeschlagene Zusatzklasse mit Metadaten für zwei lokale Objekt-Typen R und S folgende Struktur:

```
create class komp_att_of_r_and_s
  class attribute (
    alpha float default 0,
    r1 float,
    r2 float)
  (
  r_att varchar(30) not null unique,
  s_att varchar(30) not null unique,
  type varchar(20) not null,
  weight float not null,
  domination integer not null,
  mdd varchar(20)
  )
```

Die Definition der integrierten (virtuellen) Klasse wird, wie auch schon im letzten Abschnitt, um eine Methode ergänzt, die den Identifikationsalgorithmus enthält.

Innerhalb der Methode werden die Metadaten der obigen Klasse gelesen und eine Identifikation der äquivalenten Objekte durchgeführt. Eine grobe Darstellung des Algorithmus' kann [CTK96] entnommen werden.

Die Ergebnisse der Methode in Form von Paaren semantisch äquivalenter Objekte werden wiederum in eine matching-Tabelle geschrieben.

Sollen bei der Ausführung von Anfragen gegenüber der integrierten (virtuellen) Klasse widersprüchliche Angaben über bestimmte (nicht dominante) Eigenschaften von semantisch äquivalenten Objekten in Form von wahrscheinlichkeitsbasierten partiellen Werten integriert werden, so ist es notwendig, die Anfragebearbeitungskomponente des UniSQL/M-Systems dahingehend zu modifizieren.

Zusammenfassend kann festgestellt werden, daß das MDBS UniSQL/M in der aktuellen Version 3.5 systemseitig keine Konzepte speziell zur Lösung des Problems der Identifikation semantisch äquivalenter Objekte anbietet.

Wie die Ausführungen in diesem Abschnitt verdeutlicht haben, ist es jedoch möglich, mit Hilfe allgemeiner Konzepte des MDBSs eine Identifikation, Integration und Verwaltung äquivalenter Objekte auf der Basis verschiedener Identifikationsansätze zu implementieren. Hierfür wurden folgende Möglichkeiten genutzt:

- Erweiterung des Metadatenbestandes
- Verbundsichten
- nutzerdefinierte Methoden

10 Zusammenfassung

In diesem abschließenden Abschnitt sollen die Ergebnisse dieser Arbeit noch einmal zusammenfassend dargestellt werden. Ausgangspunkt der Untersuchungen war das Problem der Identifikation semantisch äquivalenter Objekte, welches im Zusammenhang mit dem Aufbau von MDBSen auftritt. Werden Realweltobjekte in mehrere Datenbanken abgebildet, so können überlappende Extensionen der betreffenden Datenbanken entstehen. Sollen diese Datenbanken dann in ein MDBS einbezogen werden, so ist es für eine semantisch korrekte Ausführung globaler Datenbank-Operationen notwendig, eine Identifikation der semantisch äquivalenten Objekte durchzuführen, um diese Objekte dann zu einem globalen Objekt integrieren zu können.

Die Untersuchung des Problems der Identifikation semantisch äquivalenter Datenbankobjekte in Abschnitt 3 zeigte, daß dieses Problem für den allgemeinen Fall algorithmisch unentscheidbar ist. Jedoch können für spezielle Fälle Konzepte zur Identifikation äquivalenter Objekte gefunden werden. In Abschnitt 5 wurden eine Reihe von Identifikations-Konzepten vorgestellt und analysiert. Es zeigte sich hierbei, daß die Voraussetzungen für den Einsatz der vorgestellten Konzepte sehr unterschiedlich sind. Jedes der Konzepte ist nur für spezielle Anwendungssituation geeignet, die maßgeblich durch die Struktur der zu untersuchenden Objekte und die zur Verfügung stehenden Zusatzinformationen über diese Objekte bestimmt wird.

Aus diesem Grund wurde in Abschnitt 6 eine Methode erarbeitet, die die Auswahl eines geeigneten Identifikationskonzeptes auf der Basis eines Entscheidungsgraphen ermöglicht. Weiterhin wurde in diesem Abschnitt untersucht, welche funktionalen Anforderungen an MDBSe gestellt werden müssen, damit eine Identifikation semantisch äquivalenter Objekte ermöglicht werden kann. Hierzu wurde eine systemunabhängige Architektur für Identifikations-Komponenten eines MDBSs entworfen.

Neben der Auswahl und Analyse von Konzepten zur Identifikation semantisch äquivalenter Objekte, war es Ziel dieser Arbeit zu untersuchen, wie sich derartige Konzepte in dem objektrelationalen MDBS UniSQL/M realisieren lassen. In Abschnitt 7 wurde hierzu das MDBS UniSQL/M vorgestellt. Schwerpunkt war dabei zu analysieren, welche Möglichkeiten in UniSQL/M zur Identifikation und Integration semantisch äquivalenter Objekte existieren. Es mußte allerdings festgestellt werden, daß keine Konzepte vorhanden sind, die speziell der Identifikation semantisch äquivalenter Objekte dienen. Dennoch konnte anhand einer Beispielanwendung mit zwei KDBSen in Abschnitt 8 und bei der Untersuchung der Realisierungsmöglichkeiten der Konzepte aus Abschnitt 5 in Abschnitt 9 gezeigt werden, daß eine Identifikation und Integration semantisch äquivalenter Objekte in UniSQL/M realisierbar ist. Eine Implementierung von Konzepten zur Identifikation, Integration und Verwaltung semantisch äquivalenter Objekte als Bestandteil des MDBMSs wäre jedoch für künftige UniSQL/M-Versionen wünschenswert.

A Verzeichnisse

Literaturverzeichnis

[CTK96] A. L. P. Chen, P. S. M. Tsai, and J.-L. Koh. Identifying Object Isomerism in Multidatabase Systems. *Distributed and Parallel Databases*, 4(2):143-168, April 1996.

[CHJ+96] S. Conrad, M. Höding, S. Janssen, G. Saake, I. Schmitt, and C. Türker. Integrity Constraints in Federated Database Design. Preprint 2, Fakultät für Informatik, Universität Magdeburg, April 1996.

[Dem89] L. DeMichiel. Resolving Database Incompatibility: An Approach to Performing Relational Operations over Mismatched Domains. *IEEE Transactions on Knowledge and Data Engineering*, 1(4):485-493, December 1989.

[KAA+93] W. Kent, R. Ahmed, J. Albert, M. Ketabchi, and M.-C. Shan. Object Identification in Multidatabase Systems. In D. K. Hsiao, E. J. Neuhold, and R. Sacks-Davis, editors, *Proc. of the IFIP WG 2.6 Working Conf. on Interoperable Database Systems (DS-5), Victoria, Australia, November, 1992*, pages 313-330, Amsterdam, 1993. North-Holland.

[KC86] S. Khoshafian, and G. P. Copeland. Object Identity. In N. Meyrowitz, editor, *Proc. of the 1st Int. Conf. on Object Oriented Programming Systems, Languages and Applications (OOPSLA '86), Portland, Oregon, SIGPLAN Notices 21(11)*, pages 406-416. ACM Press, November 1986.

[LSP+93] E.-P. Lim, J. Srivastva, S. Prabhakar, and J.Richardson. Entity Identification in Database Integration. In A. Elmagarmid and E. Neuhold, editors, *Proc. of the 9th IEEE Int. Conf. on Data Engineering, ICDE'93, Vienna, Austria, 19-23 April 1993*, pages 294-301, Los Alamitos, CA, 1993. IEEE Computer Society Press.

[Pu91] C. Pu. Key Equivalence in Heterogeneous Databases. In Y. Kambayashi, M. Rusinkiewicz, and A. Sheth, editors, *Proc. of the 1st Int. Workshop on Interoperability in Multidatabase Systems (IMS'91), Kyoto, Japan*, pages 314-316. IEEEComputer Society Press, April 1991.

[SST97] G. Saake, I. Schmitt, und C. Türker. *Objektdatenbanken - Konzepte, Sprachen, Architekturen.* International Thomson Publishing, Bonn, 1997.

[SS95] I. Schmitt and G. Saake. Managing Object Identity in Federated Database Systems. In M. Papazoglou, editor, *OOER'95: Object Oriented and Entity-Relationship Modeling, Proc. of the 14th Int. Conf., Gold Coast, Australia, December 1995*, volume 1021 of *Lecture Notes in Computer Science*, pages 400-411, Berlin, 1995. Springer-Verlag.

[SL90] A. P. Sheth and J. A. Larson. Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases. *ACM Computing Surveys*, 22(3):183-236, September 1990.

[Uni96a] *UniSQL/X User's Manual Vol. 1.* Release 3.5, UniSQL, Inc., 1996.

[Uni96b] *Database Administrators Guide.* Release 3.5, UniSQL, Inc., 1996.

[Uni96c] *UniSQL/M User's Manual.* Release 3.5, UniSQL, Inc., 1996.

[WM89] Y. Wang and S. Madnick. The Inter-Database Instance Identification Problem in Integrating Autonomous Systems. In *Proc. of the 5th IEEE Int. Conf. on Data Engineering, Los Angeles*, pages 46-55, February 1989.

[ZHK+95] G. Zhou, R. Hull, R. King, and J.-C. Franchitti. Using Object Matching and Materialization to Integrate Heterogeneous Databases. In S. Laufmann, S. Spaccapietra, and T. Yokoi, editors, *Proc. of the 3rd Int. Conf. on Cooperative Information Systems (CoopIS'95), Vienna, Austria*, pages 4-18, May 1995.

Abbildungsverzeichnis

Tabellenverzeichnis

B Selbständigkeitserklärung

Hiermit erkläre ich, daß die vorliegende Arbeit von mir selbst und nur unter Nutzung der angegebenen Hilfsmittel angefertigt wurde.

C Thesen

1. FDBSe stellen ein Integrationskonzept dar, welches es erlaubt, bestehende autonome DBSe zu einem Gesamtsystem zu integrieren.

2. Die Identifikation semantisch äquivalenter Datenbankobjekte ist für den allgemeinen Fall algorithmisch unentscheidbar.

3. Die semantisch korrekte Ausführung von globalen Datenbank-Operationen in einem MDBS erfordert die Integration, und somit als Voraussetzung hierfür auch die Identifikation, semantisch äquivalenter Datenbankobjekte.

4. Der im Rahmen dieser Arbeit entwickelte Entscheidungsgraph ermöglicht die Auswahl geeigneter Identifikationskonzepte (aus der Menge der innerhalb dieser Arbeit analysierten Identifikationskonzepte) für konkrete Anwendungsfälle.

5. Das objektrelationale MDBS UniSQL/M bietet in der aktuellen Version 3.5 keine Konzepte an, die speziell der Identifikation semantisch äquivalenter Datenbankobjekte dienen.

6. Die im Rahmen dieser Arbeit analysierten Konzepte zur Identifikation semantisch äquivalenter Datenbankobjekte lassen sich in dem objektrelationalen MDBS UniSQL/M durch eine Erweiterung des Metadatenbestandes, die Nutzung von Verbundsichten und nutzerdefinierter Methoden realisieren.

D Anhang

Dieser Anhang enthält ein vollständiges Verzeichnis aller SQL/X-Anweisungen, die für die Implementation der Beispielanwendung aus Abschnitt 8 in UniSQL/M notwendig sind.

Anlegen der Vertriebsdatenbank

```
create table transport_machine (

  product_no char(6) primary key,
  product_name varchar2(40),
  price number(10,2),
  delivery_period number(3),
  machine_class varchar2(30)
);
```

Anlegen der Fertigungsdatenbank

```
create class roller_conveyor (

  product_no char(6) not null unique,
  product_name varchar(20),
  length integer,
  width integer,
  project_state varchar(15)
);

create class driven_roller_conveyor
  as subclass of roller_conveyor (

  max_speed float,
  motor_voltages set(float)
);
```

Registrierung der KDBSe

```
register ldb vertrieb

  name 'first'
  type 'oracle'
  host 'pluto'
  user 'scott'
  password 'tiger'
;

register ldb fertigung

  name 'tom'
  type 'unisql'
  host 'pluto'
  user 'public'
  password ''
;
```

Erstellen der Proxies

```
create proxy transport_machine_proxy
  on ldb vertrieb (

  product_no char(6),
  product_name varchar(40),
  price numeric(10,2),
  delivery_period numeric(3),
  machine_class varchar(30))

  object_id(product_no)

  as
  select product_no,product_name,price,
    delivery_period,machine_class
  from transport_machine
;
```

```
create proxy roller_conveyor_proxy
  on ldb fertigung (

  product_no char(6),
  product_name varchar(20),
  length integer,
  width integer,
  project_state varchar(15))

  as
  select product_no,product_name,length,
    width,project_state
  from roller_conveyor
;

create proxy driven_roller_conveyor_proxy
  on ldb fertigung
  under roller_conveyor_proxy (

  max_speed float,
  motor_voltages set(float))

  as
  select product_no,product_name,length,
    width,project_state,
    max_speed,motor_voltages
  from driven_roller_conveyor
;
```

Erstellen der virtuellen Klassen

```
create vclass abstract_class

  (
  product_no char(6),
  product_name varchar(20)
  )

  as
  select product_no,product_name
  from transport_machine_proxy

  union all
  select product_no,product_name
  from roller_conveyor_proxy
```

```
  union all
  select product_no,product_name
  from driven_roller_conveyor_proxy
;

create vclass tm
  as subclass of abstract_class

  (
  delivery_period,
  price
  )

  as
  select product_no,product_name,
    delivery_period,price
  from transport_machine_proxy
;

create vclass rc
  as subclass of abstract_class

  (
  length,
  width,
  project_state
  )

  as
  select product_no,product_name,
    length,width,project_state
  from roller_conveyor_proxy

  union all
  select product_no,product_name,
    length,width,project_state
  from driven_roller_conveyor_proxy
;
```

```
create vclass tmnotrc
  as subclass of tm

  (
  machine_class
  )

  as
  select product_no,product_name,
    delivery_period,price,
    machine_class
  from transport_machine_proxy
  where machine_class<>'roller_conveyor'

  with check option
;

create vclass tmisrc
  as subclass of tm,rc

  as
  select t.product_no,t.product_name,
    t.delivery_period,t.price,
    r.length,r.width,r.project_state
  from transport_machine_proxy t,
    roller_coveyor_proxy r
  where t.product_no=r.product_no

  union all
  select t.product_no,t.product_name,
    t.delivery_period,t.price,
    d.length,d.width,d.project_state
  from transport_machine_proxy t,
    driven_roller_coveyor_proxy d
  where t.product_no=d.product_no
;
```

```
create vclass drc
  as subclass of rc

  (
  max_speed,
  motor_voltages
  )

  as
  select product_no,product_name,
    length,width,project_state,
    max_speed,motor_voltages
  from driven_roller_conveyor_proxy
;

create vclass tmisdrc
  as subclass of tmisrc,drc

  as
  select t.product_no,t.product_name,
    t.delivery_period,t.price,
    d.length,d.width,d.project_state,
    d.max_speed,d.motor_voltages
  from transport_machine_proxy t,
    driven_roller_coveyor_proxy d
  where t.product_no=d.product_no
;
```