OTTO-VON-GUERICKE-UNIVERSITÄT MAGDEBURG

Fakultät für Informatik
Institut für Technische Informationssysteme

Diplomarbeit

Eignungsuntersuchung des ODMG-93 Objektmodells als Objektmodell einer Föderierungsschicht zur Integration heterogener, autonomer Datenbanksysteme

Verfasser:
Eyk Hildebrandt

Betreuer der Arbeit:
Prof. Dr. habil G. Saake
Dipl. Inf. I. Schmitt

Hildebrandt, Eyk:

Eignungsuntersuchung des ODMG-93 Objektmodells als Objektmodell einer Föderierungsschicht zur Integration heterogener, autonomer Datenbanksysteme: Diplomarbeit. - Otto-von-Guericke-Universität Magdeburg, 1995.

Vorwort

Die vorliegende Arbeit wurde im Institut für Technische Informationssysteme der Fakultät für Informatik der Otto-von-Guericke-Universität Magdeburg im Sommersemester 1995 angefertigt. Ziel der Arbeit war die Eignungsuntersuchung des Objektmodells des ODMG-93 Standards für objektorientierte Datenbankmanagementsysteme als Datenmodell einer Föderierungsschicht zur Integration heterogener, autonomer Datenbanksysteme. Zum Erreichen dieses Zieles war zunächst eine Einarbeitung in die Themenbereiche ODMG-93 Standard und föderierte Datenbanksysteme notwendig. Eine umfangreiche Aufarbeitung von Literatur über die Realisierung der Integration führte zur Erarbeitung von Anforderungen an das Datenmodell einer Föderierungsschicht. Auf der Basis dieser Anforderungen wurde die Eignung des ODMG-93 Objektmodells untersucht und bewertet.

Danksagung

Hiermit möchte ich mich bei meinem Betreuer Dipl.-Inf. Ingo Schmitt für die geduldige und hilfreiche Unterstützung danken. Meinen Kommilitonen Guido Grohmann und Uwe Scholz danke ich für die unterstützenden Diskussionen. Herrn Dipl.-Phys. Frank Dennhardt danke ich für die sprachliche Unterstützung. Abschließend sei auch Herrn Prof. Dr. habil Gunter Saake gedankt.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

In diesem Kapitel werden die grundlegenden Motive, die zur Erstellung dieser Arbeit geführt haben, dargelegt. Damit soll das Verständnis für die Ziele, die sich aus der Aufgabenstellung ergeben, geschaffen werden. Die Grundlagen der Arbeit werden eingegrenzt und umrissen. Ebenfalls wird der prinzipielle Aufbau der Arbeit erläutert und begründet.

1.1 Motivation

Die stürmische Entwicklung in der Informatik der letzten Jahrzehnte und speziell in der Datenbankwelt haben bewirkt, daß die Heterogenität in verschiedenen Formen Einzug in die Datenverwaltung genommen hat und den Blick auf die Gesamtheit verstellt. Betrachtet man heutige Datenbestände in Betrieben und Institutionen, so läßt sich erkennen, daß eine Vielzahl von unterschiedlichen historisch gewachsenen Konzepten und Systemen zu ihrer Verwaltung eingesetzt werden. So werden in Unternehmen die in den späten 60er Jahren entwickelten hierarchischen Systeme gleichberechtigt mit den marktüblichen relationalen Systemen und modernen objektorientierten Systemen zur Datenhaltung eingesetzt. Der Drang nach Rationalisierung der Arbeitsabläufe erfordert, alle Daten des Unternehmens unter einem Blickwinkel zu betrachten und systemübergreifende Anwendungen zu etablieren. Die eine Möglichkeit, dies zu erreichen, wäre der völlige Neuaufbau der Datenverwaltung des Unternehmens unter Kontrolle eines modernen Datenbankmanagementsystems. Dem sprechen mehrere Gründe entgegen. Auf der einen Seite entstehen hohe Kosten durch Anschaffung von Soft- und Hardware sowie durch Arbeitsausfall während der Umstellungsperiode. Auf der anderen Seite stehen viele gewachsene Anwendungsprogramme, deren Umstellung erheblichen Aufwand erfordert.

In den letzten Jahren wurde die Idee der sogenannten föderierten Datenbanksysteme [SL90] entwickelt, die einerseits die Anforderungen nach einer systemübergreifenden Sicht erfüllen und andererseits die obengenannten Probleme vermeiden soll. Ein vollwertiges Datenbankmanagementsystem koordiniert dabei das Zusammenspiel der heterogenen autonom bleibenden Datenbanksysteme und fungiert als sogenannte Föderierungsschicht. Vielfältige Forschungsarbeit wurde auf diesem Gebiet in den letzten Jahren geleistet. Dabei wurden sowohl Prototypen als auch erste einsatzfähige Produkte entwickelt. Unterschiedliche Ansätze wurden hinsichtlich des zu wählenden Datenmodells für die Föderierungsschicht verfolgt. Das Datenmodell ist ein entscheidender Faktor für den erfolgreichen Einsatz dieser Systeme, da es der Träger eines einheitlichen Schemas über alle verwalteten Daten ist.

Eine der rasantesten Entwicklungen der letzten Jahre ist der Einzug der Objektorientierung in fast alle Bereiche der Informatik. Auch im Datenbankbereich brachten objektorientierte Systeme neue Dynamik in die Entwicklung. Sie erwiesen sich als geeignet, spezielle Probleme, wie die Abbildung komplexer Zusammenhänge, zu lösen und in neue Bereiche wie Multimedia vorzustoßen. Trotz unbestreitbarer Vorteile in verschiedenen Bereichen konnten die objektorientierten Datenbanksysteme noch nicht die Akzeptanz erhalten, wie sie zum Beispiel relationale Systeme besitzen. Ein Grund dafür war das Fehlen eines einheitlichen Objektmodells und eines Sprachstandards, wie es in relationalen Systemen durch das relationale Modell und den SQL-Standard gegeben ist. Mit der Entwicklung des ODMG-93 Standards [Cat94] für Objektorientierte Datenbanksysteme wurde ein Ansatz geschaffen, dieses Problem zu beheben.

Der Einsatz von Datenbankmanagementsystemen mit einem objektorientierten Datenmodell als Föderierungsschicht eines föderierten Datenbanksystems wird als geeignet angesehen, die

Heterogenität in der Datenverwaltung zu überwinden [McL93]. Es ist deshalb eine lohnenswerte Aufgabe zu untersuchen, ob und wie sich das Objektmodell des ODMG-93 Standards als Datenmodell einer Föderierungsschicht eignet.

1.2 Aufbau der Arbeit

Zwei Schwerpunkte bilden die Grundlagen dieser Arbeit, der ODMG-93 Standard für objektorientierte Datenbankmanagementsysteme und die Integration heterogener, autonomer Datenbanksysteme durch ein föderiertes Datenbanksystem. Es ist die Aufgabe dieser Arbeit zu untersuchen, ob und wie auf der Basis des Objektmodells des ODMG-93 Standards eine solche Integration durchgeführt werden kann. Zur Verwirklichung dieser Aufgabe wurde die im folgenden beschriebene Gliederung der Arbeit erstellt.

In Kapitel 2 erfolgt eine Einführung in den ODMG-93 Standard. Es werden die Bestandteile des Standards, insbesondere das für die Arbeit wichtige Objektmodell, vorgestellt und beschrieben. Eine graphische Notation für Schemabeschreibungen basierend auf dem Objektmodell wird eingeführt. Ebenso werden die wichtigsten Sprachkonstrukte der zum Standard gehörenden Schemadefinitionssprache (ODL) vorgestellt. Damit können in der weiteren Arbeit Beispiele in graphischer und ODL-Notation beschrieben werden.

In Kapitel 3 werden die Grundlagen föderierter Datenbanksysteme erarbeitet. Dabei werden die wichtigsten Begriffe geklärt, die Aufgaben und die Realisierung föderierter Datenbanksysteme umrissen. Es wird ein Referenzmodell vorgestellt, daß eine wichtige Grundlage für die Untersuchung darstellt. Eine Einordnung und Abgrenzung föderierter Datenbanksysteme rundet das Kapitel ab.

In den Kapiteln 4 und 5 werden die Voraussetzungen für die Eignungsuntersuchung geschaffen. So wird in Kapitel 4 der Begriff des Konflikts bei der Integration heterogener, autonomer Datenbanksysteme eingeführt, denn das Erkennen und das Auflösen von Konflikten ist die Basis für eine erfolgreiche Integration. Darum werden in diesem Kapitel die wichtigsten Konflikte beschrieben und klassifiziert. In Kapitel 5 werden dann Anforderungen erarbeitet, anhand derer die Eignung des ODMG-93 Objektmodells nachgewiesen werden soll. Basis für diese Anforderungen sind vier wichtige Prozesse, die zusammen die Integration realisieren.

In Kapitel 6 folgt die Eignungsuntersuchung des ODMG-93 Objektmodells. Orientiert an den in Kapitel 5 beschriebenen Prozessen wird das ODMG-93 Objektmodell auf Erfüllung der Anforderungen untersucht. Dabei werden Vorschläge für die Realisierung einer Integration gemacht, sowie existierende Ansätze auf ihre Umsetzbarkeit überprüft.

Im letzten Kapitel werden die Ergebnisse der Arbeit aufgearbeitet und zusammengefaßt. Ein Ausblick schließt die Arbeit ab.

2 Einführung in den ODMG-93 Standard

Dieses Kapitel dient dazu, den ODMG-93 Standard für objektorientierte Datenbankmanagementsysteme, der eine wichtige Grundlage für diese Arbeit darstellt, vorzustellen. Die Kenntnis der elementaren Prinzipien sowie der Darstellungsmittel des Standards ist notwendig, um die angestrebte Untersuchung durchführen zu können. Dazu werden in einem ersten Abschnitt die Hintergründe und Ziele des Standards erläutert. Anschließend folgt die Beschreibung der wichtigsten Grundsätze des Objektmodells. Dabei wird ein graphische Notation für Schemabeschreibungen mit dem Objektmodell eingeführt. In weiteren Abschnitten werden die auf dem Objektmodell basierenden Sprachen vorgestellt. Insbesondere wird dabei auf Sprachkonstrukte der Schemadefinitionssprache ODL eingegangen, um später Beispiele für Schemata präsentieren zu können. Ein Abschnitt über zukünftige Erweiterungen des Standards schließt das Kapitel ab.

2.1 Grundlagen

Die von fünf führenden Herstellern objektorientierter Datenbankmanagementsysteme als Untergruppe der Object Management Group (OMG) gegründete Object Database Management Group (ODMG) verabschiedete 1993 die erste Fassung eines Standards (ODMG-93) für objektorientierte Datenbankmanagementsysteme (ODBMSe). Ein Ziel des Standards ist es, das Schreiben portierbarer Applikationen für ODBMSe zu ermöglichen und damit die Akzeptanz der Systeme zu erhöhen. Außerdem soll Interoperabilität zwischen den Systemen ermöglicht werden, um z. B. die Föderation autonomer Datenbanken zu erleichtern. Die Beschreibung des Standards folgt [Cat94].

Kernstück des Standards ist das Objektmodell, das auf der Basis des OMG Objektmodells entwickelt wurde. Es wurde erweitert, um den speziellen Anforderungen von Datenbanksystemen zu genügen. Als Datendefinitionssprache zur Beschreibung der Schemata auf Grundlage des Objektmodells dient die Object Definition Language (ODL). Desweiteren wurde eine deklarative Anfragesprache Object Query Language (OQL) in Anlehnung an den relationalen Standard SQL geschaffen. Zur Manipulation persistenter Objekte in ODBMSen wurden Sprachanbindungen als Object Manipulation Language (OML) für die zwei verbreitetsten objektorientierten Programmiersprachen C++ und Smalltalk definiert.

2.2 Objektmodell

Das Objektmodell dient als Grundlage zur Modellierung konzeptioneller Datenbankschemata in jedem ODMG-93 konformen ODBMS. Grundlegendes Modellierungsprimitiv ist das Objekt. Objekte können zu Typen auf der Basis gleicher Struktur und gleichen Verhaltens kategorisiert werden. Die Struktur ergibt sich aus einer Menge von Eigenschaften (Attribute und Beziehungen), deren Werte den Zustand eines Objektes ergeben. Das Verhalten wird durch eine Menge von Operationen, die an einem Objekt ausgeführt werden können, definiert. Ein Typ besitzt ein Interface, d. h. die externe Beschreibung der Struktur und des Verhaltens seiner Instanzen, sowie eine oder mehrere Implementationen, die Datenstrukturen und Methoden zur physikalischen Repräsentation und Unterstützung des extern sichtbaren Zustandes und Verhaltens bereitstellen. Die Kombination eines Interfaces und einer für den Typ definierten Implementation wird als Klasse bezeichnet.

Typen sind selbst Objekte und besitzen eigene Eigenschaften. Typen werden in Hierarchien von Sub- und Supertypen angeordnet, wobei ein Subtyp alle Charakteristiken (Eigenschaften

und Operationen) seiner Supertypen erbt, sowie neue Charakteristiken für seine Instanzen definieren kann. Die Instanz eines Subtyps ist indirekte Instanz all seiner Supertypen [BFN94]. Die durch die multiple Vererbung möglicherweise entstehenden Namenskonflikte müssen durch Redefinition behandelt werden. Das Objektmodell unterscheidet zwischen instantiierbaren und abstrakten Typen, die keine Implementation besitzen. Als speziell auf den Datenbankbereich zugeschnittene Typeigenschaften unterstützt das Objektmodell Extensionen und wertbasierte Schlüssel. Die explizite Deklaration der Extension, d. h. die Menge aller Instanzen eines Typs im Interface, erlaubt einen schnellen mengenorientierten Zugriff auf Objekte der Datenbank. Zwischen Typ und Extension besteht ein direkter Zusammenhang, d. h. die Instanz eines Typs ist automatisch Mitglied der Extension des Typs, und die Extension eines Subtyps ist eine Untermenge der Extension des Supertyps. Ebenfalls im Interface können Schlüsselbedingungen, einfache oder zusammengesetzte, bestehend aus Eigenschaften des Typs, definiert werden.

Eine vordefinierte Typhierarchie als Modellierungsgrundlage beschreibt die grundlegenden Konzepte des Objektmodells. Nachfolgend werden die Basistypen der Hierarchie gezeigt und die damit verbundenen Konzepte erläutert:

- Denotable_Object
  - Object
  - Literal
- Characteristic
  - Operation
  - Property
    - Attribute
    - Relationship

In der aktuellen Revision gibt es Einschränkungen bezüglich der Typeigenschaften. Benutzerdefinierte Subtypen mit Interfacebeschreibung dürfen nur unterhalb des Typs *Object* erstellt werden. Extensionen, Schlüsselbedingungen und benutzerdefinierte Implementationen sind nur für Instanzen des Typs *Object*, d. h. Objekttypen, möglich.

Der Typ *Denotable_Object* ist der abstrakte Supertyp für veränderbare Objekte (Typ *Object*) und unveränderbare Objekte (Typ *Literal*), die jeweils atomar oder strukturiert sein können. Alle Instanzen der Subtypen von *Denotable_Object* haben eine eigene Identität, die jedoch unterschiedlich repräsentiert und gehandhabt wird.

Benutzerdefinierte Objekttypen bilden die Grundlage jeder Schemadefinition mit Hilfe des ODMG-93 Objektmodells. Die Instanzen der Objekttypen, die Objekte, sind die Elemente einer objektorientierten Datenbank. Wie bereits erläutert, können Objekttypen als Subtypen eines oder mehrerer anderer Objekttypen definiert werden. Abbildung 2.1 zeigt einen Objekttyp A als Subtyp eines Objekttyps B.

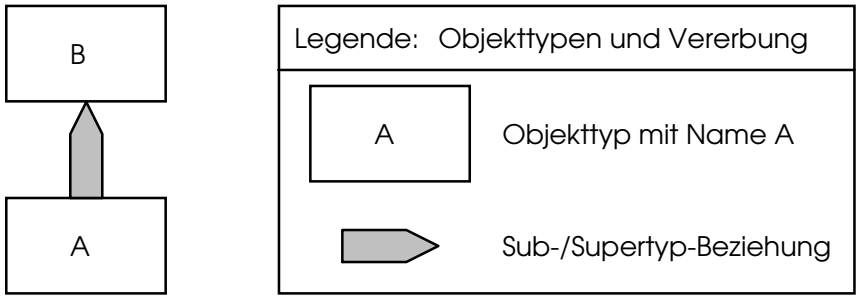

Abbildung 2.1: Vererbungsbeziehung zwischen zwei Objekttypen

Grundlegendes Konzept für Instanzen von Objekttypen ist die Objektidentität, die sich im Gegensatz zu den Werten der Charakteristiken eines Objekts während seiner Lebenszeit nicht ändern kann. Die Objektidentität für Objekte wird durch einen implementationsabhängigen *object identifier* (OID) repräsentiert und unterscheidet jedes Objekt innerhalb einer Datenbank von allen anderen Objekten der Datenbank. Unabhängig von der Objektidentität können Objekte mit einem oder mehreren Namen belegt werden, die zur Laufzeit eine eindeutige Identifikation eines Objektes ermöglichen. Für Objekttypen sind die Operationen *create*, zum Erzeugen eines Objektes, und *delete*, zum Löschen eines Objektes, vordefiniert. Ein weiteres Konzept für Instanzen von Objektypen ist die Lebenszeit, wobei zwischen transienten Objekten innerhalb einer Prozedur (*coterminus_with_procedure*) oder eines Prozesses (*coterminus_with_process*) und persistenten Objekten innerhalb einer Datenbank (*coterminus_with_database*) unterschieden wird. Die Lebenszeit eines Objektes wird bei seiner Erstellung festgelegt und kann danach nicht mehr verändert werden.

Neben atomaren Objekttypen gibt es auch strukturierte Objekttypen, die sich in Mengen (*Collection*) von Objekten (*Set*, *Bag*, *List* und *Array*) und Objektstrukturen (*Structure*) unterteilen. Strukturierte Objekttypen dienen nicht der Schemadefinition, ihre Instanzen sollen die Ergebnisse von Anfragen mit der OQL (siehe Abschnitt 2.4) aufnehmen.

Literale, d. h. Instanzen von Subtypen vom Typ *Literal*, sind Objekte, deren Wert nicht verändert werden kann, d. h. ihre Identität ergibt sich aus ihrem Wert. Als instanzierbare atomare Subtypen werden *Integer*, *Float*, *Boolean* und *Character* unterstützt, deren Präzisierung von der gegebenen Sprachanbindung abhängt. Desweiteren werden auch strukturierte Literaltypen wie String-, Datums- und Aufzählungstypen angeboten.

Zu jedem Objekttyp werden eine Menge von Instanzen der Subtypen *Attribute* und *Relationship* des Typs *Property* definiert, deren Instanzen den Zustand eines Objekts bilden. Attribute, d. h. Instanzen vom Typ *Attribute*, sind Eigenschaften einzelner Objekttypen, deren Wertebereiche Literaltypen sind. Die Werte eines Attributs können ermittelt und manipuliert werden, wofür die Operationen *get_value* und *set_value* vordefiniert sind. Instanzen vom Typ *Relationship*, im folgenden Beziehungen genannt, sind Eigenschaften, die zwischen genau zwei Objekttypen definiert sind, wobei die Möglichkeit einer rekursiven Beziehung auf einem Objekttyp besteht. Beziehungen werden durch sogenannte *traversal paths* in der Interfacebeschreibung bekanntgemacht. Mit Hilfe der *inverse* Klausel kann dem System das Zusammengehören zweier *traversal paths* als bidirektionale Beziehung bekanntgegeben werden. Beziehungen im Objektmodell beinhalten das Konzept der referentiellen Integrität. Damit ist die explizite Definition von 1:1-, 1:n-, m:1- und m:n-Beziehungen möglich. Mehrwertige Beziehungen können geordnet sein. N-äre Beziehungen werden nicht unterstützt. Operationen zum Erzeugen und Löschen von Beziehungsinstanzen sowie zum Traversieren entlang von Beziehungen sind vordefiniert. Abbildung 2.2 zeigt eine rekursive 1:1-Beziehung auf einem Objekttypen A sowie eine m:n-Beziehung zwischen Objekttypen B und C, wobei RA1, RA2, RC, und RB die zugehörigen *traversal paths* bezeichnen.

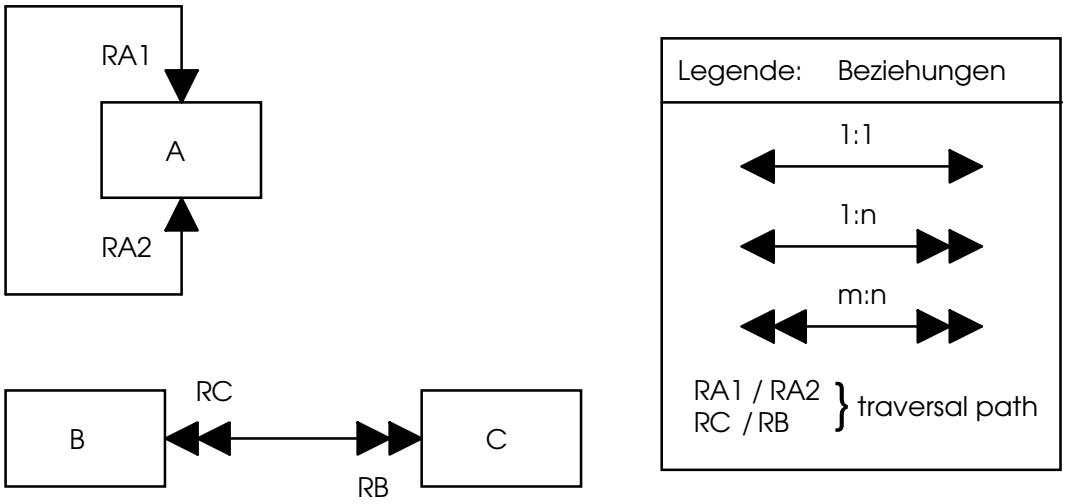

Abbildung 2.2: Beziehungen zwischen Objekttypen

Das Verhalten von Objekttypen wird durch eine Menge von Instanzen vom Typ *Operation*, den Operationen, bestimmt. In der Interfacebeschreibung ist jede Operation durch die *operation signature* vertreten, die Argumente, Rückgabewerte und mögliche Ausnahmebedingungen enthält. Als Argumente und Rückgabewerte können alle Subtypen von *Denotable_Object* fungieren.

Das Objektmodell unterstützt ein Transaktionsmodell zur Manipulation von persistenten Objekten, d. h. der Zugriff, das Einfügen, Ändern und Löschen von persistenten Objekten kann nur innerhalb einer Transaktion vorgenommen werden. Dazu wird der vordefinierte Typ *Transaction* verwendet. Die vordefinierten Operationen *begin*, *commit* und *abort* gewährleisten die Atomarität und die Dauerhaftigkeit. Transaktionen können geschachtelt werden, wobei das Isolationsprinzip innerhalb der Schachtelung aufgehoben wird. Mit der vordefinierten Operation *abort_to_top_level* können Subtransaktionen zur obersten Transaktion abbrechen, ohne daß diese ebenfalls abbrechen muß. Damit wird ein offen geschachteltes Transaktionsmodell unterstützt. Das unterstützte Concurrency Control Protokoll beruht auf einem pessimistischen Ansatz basierend auf Lese- und Schreibsperren auf Objekten. Transaktionen gegenüber mehreren Datenbanken werden in der aktuellen Revision nicht unterstützt.

Ein weiterer vordefinierter Typ ist der Typ *Database*, dessen Instanzen die Datenbanken sind. Zu jeder Datenbank gehört ein Schema mit Typdefinitionen, und die Datenbank enthält die Instanzen dieser Typen. Zum Typ *Database* sind die Operationen *open* zum Öffnen der Datenbank, *close* zum Schließen der Datenbank, sowie die booleschen Operationen *contains_object?* zur Überprüfung des Enthaltenseins von Objekten in der Datenbank und *lookup_object* zum Zugriff auf Objekte, vordefiniert.

2.3 Object Definition Language

Die ODL ist eine Spezifikationssprache zur Schemadefinition auf Basis des Objektmodells. Die Syntax ist unabhängig von bisherigen Sprachansätzen, jedoch so entwickelt, daß Definitionen in objektorientierte Programmiersprachen wie C++ oder Smalltalk (siehe Abschnitt 2.5) sowie in Datendefinitionssprachen existierender ODBMSe portiert werden können. In der aktuellen Version des Standards erlaubt die ODL nur die Definition von Interfacebeschreibungen von Objekttypen. Es folgt eine kurze Beschreibung der wichtigsten Sprachkonstrukte der ODL, soweit sie in der weiteren Arbeit verwendet werden.

Für die Sprachbeschreibung werden folgende Festlegungen getroffen:

⇒ **symbol**	- Schlüsselwörter und Symbole der ODL
⇒ <symbol>	- nichtterminierende Symbole
⇒ <*symbol*>	- benutzerdefinierte Symbole
⇒ [symbol]	- optionales Symbol
⇒ symbol1 | symbol2	- entweder symbol1 oder symbol2 verwenden.

Danach läßt sich folgende nichtvollständige Spezifikation einer Interfacebeschreibung eines Objekttyps ausdrücken.

```
<type definition>   ::= interface <type_name>
                                  [:<supertype_list>]
                       {
                       [<type_property_list>]
                       [<property_list>]
                       [<operation_list>]
                       };
<supertype_list>    ::= <type_name>
                       |<type_name>,<supertype_list>
<type_property_list>::= <type_property>;
                       |<type_property> <type_property_list>
<type_property>     ::= extent <extent_name>
                       |key[s] <key_list>
<key_list>          ::= <key_spec>|<key_spec>,<key_list>
<key_spec>          ::= <property_name>
                       |(<property_name_list>)
<property_name_list>::= <property_name>
                       |<property_name>,<property_name_list>
<property_name>     ::= <attribute_name>
                       |<traversal_path_name>
<property_list>     ::= <property_spec>;
                       |<property_spec> <property_list>
<property_spec>     ::= <attribute_spec>|<relationship_spec>
<attribute_spec>    ::= [attribute] <domain_type>
                       [[<integer>]] <attribute_name>
<domain_type>       ::= <atomic_literal>|<structured_literal>
                       |<collection of literals>
<relationship_spec> ::= [relationship]
                       [<col_type>]<target_type><t_p_name_1>
                       inverse <target_type>::<t_p_name_2>
                       [{order_by <attribute_list>}]
<col_type>          ::= SET|LIST
<attribute_list>    ::= <attribute_name>
                       |<attribute_name> <attribute_list>
<operation_list>    ::= <operation_spec>;
                       |<operation_spec> <operation_list>
<operation_spec>    ::= <return-type> <operation_name>
                       ([<argument_list>])
```

Für <atomic_literal> und <structured_literal> werden die entsprechend vordefinierten, in Abschnitt 2.2 behandelten, Literaltypen verwendet. Auf eine genaue Beschreibung von <argument_list> wird verzichtet.

Es folgt ein Beispiel für ein konzeptionelles Datenbankschema basierend auf dem Objektmodell und ausgedrückt in der ODL.

Beispiel 2.1: Das Beispielschema modelliert die Personalabteilung einer Firma. Abbildung 2.3 zeigt das Schema in graphischer Notation. Daran schließt sich die ODL-Notation an.

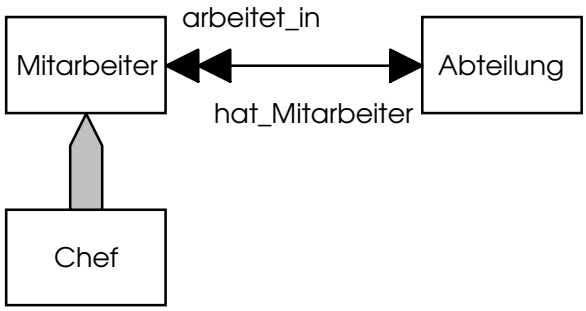

Abbildung 2.3: Schema der Personalabteilung einer Firma

```
interface Mitarbeiter {
  extent Personal;
  key (name,geb_dat);
  attribute string name;
  attribute date geb_dat;
  attribute struct<
    struct<string<50> str_name, integer[3] nummer> strasse,
    string<50> stadt,
    string<5> plz,
    string<3> land> adresse;
  relationship abteilung arbeitet_in
    inverse abteilung::hat_Mitarbeiter  };

interface chef:mitarbeiter {
  extent chefs;
  attribute integer parkplatznr; };

interface abteilung {
  extent abteilungen;
  keys anr,name;
  attribute integer anr;
  attribute string name;
  relationship List<Mitarbeiter> hat_Mitarbeiter
    inverse Mitarbeiter::arbeitet_in
    {order_by name,geb_dat} };
```

2.4 Object Query Language

Die OQL ist eine deklarative Anfragesprache auf der Basis des Objektmodells. Sie soll sowohl navigierenden als auch mengenmäßigen Zugriff auf die in einer Datenbank enthaltenen Objekte ermöglichen. Die Syntax und Semantik der OQL kann über Funktionen in objektorientierte Sprachen eingebunden werden (siehe Abschnitt 2.5). Der navigierende Zugriff erfolgt über benannte Objekte, die als Eintrittspunkte in die Datenbank fungieren. Für den mengenmäßigen Zugriff wird eine SQL-ähnliche Syntax, nämlich eine *select-from-where* Klausel mit Gruppierungs-, Sortiertungs- und Aggregatfunktionen, bereitgestellt. Desweiteren werden arithmetische Ausdrücke (*not*, +, - u. a.), Quantifikatoren (*forall*, *exists*), Vergleichs-

operationen (=, <, >, *in* u. a.) und Konvertierungsausdrücke (*listtoset*, *flatten* u. a.) unterstützt. Grundlage für den mengenmäßigen Zugriff sind die benannten Extensionen der Objekttypen.

Ergebnisse von Anfragen an Datenbanken basierend auf dem Objektmodell können sehr unterschiedlicher Art sein. So können alle in einer Datenbank existierenden Objekte Ergebnis einer Anfrage sein. Es können jedoch durch Anfragen auch neue Objekte eines existierenden Typs erzeugt werden. Durch die strukturierten Objekttypen ist es möglich, die Ergebnisse mengenmäßiger Anfragen in die Typhierarchie einzubinden.

Die OQL bietet durch die *select-from-where*-Klausel einen Ansatz für ein Sichtenkonzept des ODMG-93 Standards ähnlich dem des relationalen Modells. So könnten virtuelle Objekttypen auf der Basis einer Anfrage mit dieser Klausel aus bestehenden Objekttypen abgeleitet werden. Zur Zeit sieht der Standard jedoch keine explizite Definition von externen Schemata (Views) vor.

2.5 Programmiersprachenanbindung

Da die Datenmodelle und Datenbanksprachen sehr vieler existierender ODBMSe auf weiterentwickelten objektorientierten Programmiersprachen beruhen, ist eine Anbindung des Standards an diese Sprachen notwendig. In der aktuellen Version existiert diese Anbindung für C++ und Smalltalk. Weitere Anbindungen sollen folgen. Am Beispiel der C++-Anbindung sollen die Grundprinzipien erläutert werden.

Die C++ ODL beschreibt ein Datenbankschema als eine Menge von Objektklassen im syntaktischen Stil des Deklarationsteils eines C++-Programms. Dafür müssen nur wenige C++-Erweiterungen definiert werden. Dazu gehört die Aufnahme des Schlüsselwortes *inverse* für bidirektionale Beziehungen. Beziehungen werden über die Template Klasse *Ref* ausgedrückt. Da Extensionen und wertbasierte Schlüssel nicht explizit von C++ unterstützt werden, müssen sie über Datenstrukturen und Methoden implementiert werden.

Die C++ OML beschreibt die Manipulation persistenter Objekte. Prinzipiell werden dabei die in C++ vorhandenen Operationen zur Manipulation transienter Objekte genutzt, die entsprechend den besonderen Anforderungen überladen werden. Eine Klasse *Pobject* wird eingeführt, von der alle Klassen erben, für die persistente Instanzen erlaubt werden. Die Operationen der Template Klasse *Ref* dienen zum Auf- und Abbau von Beziehungen. Für Transaktionen wird die Klasse *Transaction* mit den in Abschnitt 2.2 erläuterten Operationen eingeführt. Ebenso wird eine Klasse *Database* für Datenbanken eingeführt.

. Die C++ OQL besteht aus zwei Funktionen, die als ein Argument einen String zur Aufnahme der Anfrage besitzen. Zur Laufzeit übersetzen diese Funktionen die Anfrage, werten sie aus und legen das Ergebnis in einem weiteren Argument ab. Die eine Funktion *query* ist eine Methode der Template Klasse *Collection*, die zur Aufnahme von Objektmengen eingeführt wird. Das Ergebnis einer Anfrage mit dieser Funktion ist immer eine Instanz der Klasse *Collection*. Die andere Funktion *oql* ist eine freistehende Funktion, die dem Programmierer den Zugang zur vollen Funktionalität der OQL sicherstellt. Der Ergebnistyp der Anfrage sind Referenzen auf Variablen. Ebenso kann die Anfrage Parameter enthalten, die erst zur Laufzeit eingegeben werden.

2.6 Zukünftige Erweiterungen

Der ODMG Standard für objektorientierte Datenbankmanagementsysteme liegt derzeit in seiner ersten Version ODMG-93 vor. Diese sogenannte Basisebene soll in weiteren

Standardisierungsschritten sukzessive um zusätzliche Funktionalität erweitert werden. Einiges von der Funktionalität, die für die nächste Version vorgesehen ist, wurde in [Cat94] vorgestellt. Davon sollen einige Beispiele, die wichtige wünschenswerte Funktionalität beinhalten, an dieser Stelle genannt werden.

Objekte können direkte Instanzen von mehreren Typen sein. Ein Objekt darf dynamisch einem Typ zugeordnet werden und darf ihn dynamisch wieder verlieren. Diese Eigenschaften sind unter dem Begriff des Rollenkonzeptes bekannt. Die Lebenszeit von Objekten kann dynamisch geändert werden.

Das Transaktionsmodell wird erweitert, um die Effizienz des parallelen Zugriffs auf eine Datenbank zu erhöhen. Eine einzelne Transaktion erlaubt den Zugriff auf Objekte in mehr als einer Datenbank. Mehrere Prozesse können an einer einzelnen Transaktion teilnehmen.

Die Typen *Schema* und *Subschema* werden eingeführt. Zu jeder Datenbank gehört ein Schema, verschiedene Datenbanken können dasselbe Schema nutzen. Ein Subschema ist eine Untermenge der Typen genau eines Schemas. Eigenschaften von Objekttypen eines Schemas können in einem Subschema weggelassen werden. Die Frage, ob in einem Subschema die Definition von neuen aus verschiedenen Objekttypen des Schemas abgeleiteten Objekttypen erlaubt werden soll, ist offen. Zugriffskontrolle und Datensicherungsmechanismen werden auf Subschemaebene definiert.

Schemainformationen werden wie Objekte behandelt, d. h. Daten und Metadaten besitzen die gleiche Struktur. Metadaten können mit den Sprachkonstrukten der OML manipuliert werden, so daß eine Schemaevolution möglich ist.

3 Einführung in die Föderierungsproblematik

Nach der Einführung in den ODMG-93 Standard soll nun die zweite Grundlage der Arbeit, die Föderierung heterogener, autonomer Datenbanksysteme näher betrachtet werden. Da es in der weiteren Arbeit notwendig sein wird, verschiedene Aspekte dieser Thematik tiefgründiger zu beleuchten, wird mit diesem Kapitel ein Ausgangspunkt für die weitere Beschäftigung mit diesem Gebiet gelegt. Dazu werden in einem ersten Abschnitt die grundlegenden Begriffe geklärt und definiert. Der zweite Abschnitt dient dazu, die Aufgaben und Einsatzmöglichkeiten föderierter Datenbanksysteme zu erläutern. Im dritten Abschnitt wird ein Referenzmodell vorgestellt, an dem sich die durchzuführende Untersuchung orientiert. Am Ende dieses Kapitels steht eine erforderliche Aufgliederung dieses Gebietes sowie eine Abgrenzung von anderen Datenbanksystemen.

3.1 Begriffsbildung

Das wachsende Bedürfnis nach Kooperation zwischen unterschiedlichen Datenhaltungssystemen hat eine Anzahl von Realisierungsansätzen und damit einhergehenden neuen Begriffen geschaffen. Diese Arbeit beschäftigt sich mit dem Ansatz einer sogenannten *Föderation* heterogener autonomer Datenbanksysteme. Dazu gilt es die wichtigsten damit verbundenen Begriffe zu klären.

Ein *föderiertes Datenbanksystem* (FDBS) ist nach [SL90] ein Datenbanksystem (DBS), das Nutzern einheitliche Operationen auf autonomen und möglicherweise heterogenen Datenbanksystemen durch Schaffung einer homogenisierten Betrachtungsweise auf die in den einzelnen Datenbanken verwalteten Daten ermöglicht. Diese Datenbanksysteme werden in diesem Zusammenhang als *Komponentendatenbanksysteme* (CDBSe) bezeichnet. Die Software, die die Zusammenarbeit der CDBSe innerhalb einer Föderation koordiniert, heißt nach [SL90] *föderatives Datenbankmanagementsystem* (FDBMS). Das Datenmodell, durch das die homogenisierte Betrachtungsweise mittels eines einheitlichen Schemas auf die Datenbestände der von den CDBSen verwalteten Datenbanken ermöglicht werden soll, nennt man *gemeinsames Datenmodell* (GDM). Abbildung 3.1 zeigt ein Beispiel eines FDBSs mit heterogenen CDBSen.

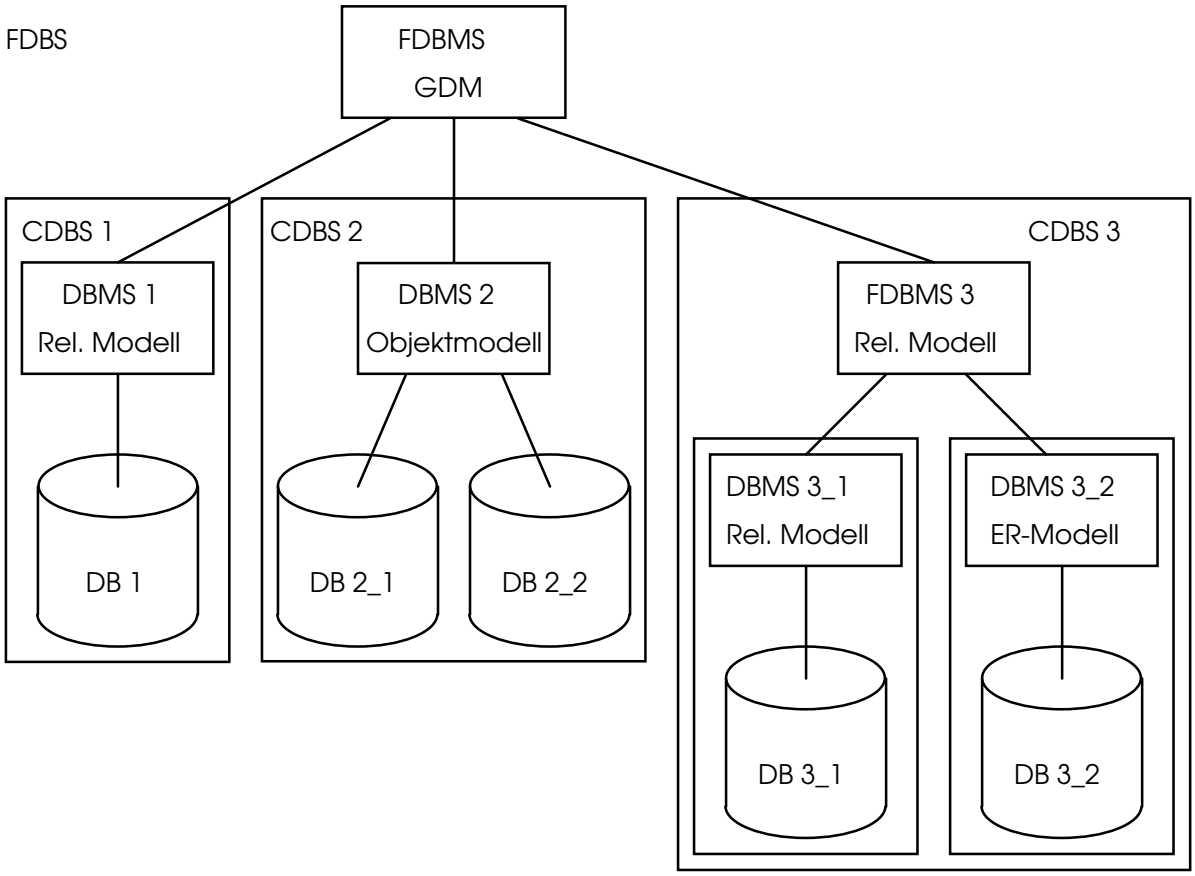

Abbildung 3.1: Ein FDBS mit heterogenen Komponenten

3.2 Charakterisierung und Aufgaben föderierter Datenbanksysteme

Föderierte Datenbanksysteme stellen eine Möglichkeit dar, unterschiedliche Datenbanksysteme unternehmensweit zu integrieren. In diesem Abschnitt wird untersucht, welche Charakteristiken föderierte Datenbanksysteme kennzeichnen, welche Einsatzmöglichkeiten es für diese Systeme gibt und welche Aufgaben das FDBMS übernimmt.

3.2.1 Verteilung, Autonomie und Heterogenität

Drei orthogonale Dimensionen charakterisieren nach [SL90] Systeme, die aus mehreren Datenbanksystemen bestehen, wozu auch FDBSe zählen. Dies sind Verteilung, Autonomie und Heterogenität. Anhand dieser Dimensionen lassen sich die Hauptaufgaben von FDBSen herleiten.

Die Daten eines FDBSs werden in verschiedenen Datenbanken unter der Kontrolle verschiedener CDBSe verwaltet und gespeichert, d. h. sie sind verteilt. Ein wichtiges Ziel eines FDBSs ist es, dem Nutzer die Verteilung und Herkunft der Daten vorzuenthalten.

Autonomie bedeutet in diesem Zusammenhang, daß die CDBSe die Kontrolle über die verwalteten Datenbanken behalten. Eine besondere Forderung an ein FDBS ist in diesem Bereich die Unabhängigkeit der lokalen Applikationen der CDBSe gegenüber dem FDBS, d. h. alle Anwendungen eines CDBSs werden von der Teilnahme am FDBS nicht beeinträchtigt und können weiterlaufen. Durch weitgehenden Schutz der Autonomie und der lokalen An-

wendungen sollte der Betrieb eines CDBSs nahezu unabhängig von seiner Teilnahme an einem FDBS möglich sein. Vier Arten von Autonomie werden von [SL90] dabei unterschieden:

1. Designautonomie - Kontrolle über Daten und deren Repräsentation,
2. Kommunikationsautonomie - Kontrolle über Kommunikation mit anderen CDBSen,
3. Ausführungsautonomie - Kontrolle über Ausführung aller Operationen,
4. Assoziationsautonomie - Kontrolle über Teilung der Ressourcen und der Funktionalität.

Letzteres bedeutet dabei, daß ein CDBS entscheiden kann, welche der von ihm verwalteten Daten es dem FDBS zur Verfügung stellt und welche Operationen darauf erlaubt werden. Die weitgehende Wahrung aller Autonomiearten ist oftmals eine Voraussetzung für die Teilnahme eines Datenbanksystems als CDBS eines FDBSs, obwohl dadurch Konflikte mit anderen Anforderungen auftreten können. So kann z. B. eine strikte Ausführungs- und Kommunikationsautonomie ein korrektes und effizientes Transaktionsmanagement innerhalb des föderierten Systems behindern [Rah94]. Deshalb ist es notwendig, genau abzuwägen, wie und inwieweit die Autonomiearten geschützt werden müssen.

Die CDBSe weisen in einem FDBS zumeist verschiedene Grade von Heterogenität auf. Die CDBSe können zentrale, verteilte oder wieder föderierte Datenbanksysteme, aber auch einfache Datenhaltungs- oder Filesysteme sein. Bei den Datenbanksystemen können unterschiedliche Datenmodelle (relational, hierarchisch, objektorientiert u. a.) zum Einsatz kommen. Datenbankmanagementsysteme mit demselben Datenmodell können sich, wenn vorhanden, zum Beispiel hinsichtlich Anfragesprache und Transaktionsmanagementfähigkeiten unterscheiden. Dieser Bereich wird als syntaktische Heterogenität bezeichnet. Ein weiterer Bereich ist die semantische Heterogenität der konzeptionellen Schemata. Sie entsteht durch die Abstraktion, Modellierung und Definition von Daten durch die Betreiber der CDBSe. Dabei können unterschiedliche Weltausschnitte oder verschiedene Sichten auf denselben Weltausschnitt dargestellt werden. Die Überwindung der Heterogenität ist das wichtigste Ziel eines FDBSs, da dadurch einem Nutzer der homogene Zugriff auf die Daten ermöglicht wird.

Zusammenfassend lassen sich anhand der drei Dimensionen Verteilung, Autonomie und Heterogenität die Hauptaufgaben von föderierten Datenbanksystemen aufstellen:

- Überwindung von syntaktischer und semantischer Heterogenität der CDBSe,
- Transparenz der Datenverteilung für den FDBS-Nutzer,
- Schutz der Autonomiearten und damit der lokalen Anwendungen der CDBSe.

3.2.2 Weitere Einsatzmöglichkeiten

Neben den in Abschnitt 3.2.1 erläuterten Hauptaufgaben können FDBSe weitere Einsatzzwecke erfüllen. Zwei Möglichkeiten sollen an dieser Stelle angerissen werden.

In heutigen Unternehmen trifft man oft Datenhaltungssysteme der verschiedensten Generationen gleichzeitig an. Dabei entsteht oft der Wunsch, ältere Systeme aus dem aktiven Betrieb zu nehmen und ihre Daten auf ein moderneres System zu übertragen. Dies erfordert jedoch normalerweise Unterbrechung des laufenden Betriebes und aufwendige Rekodierungen. Sind aber sowohl das zu eliminierende System (Quellsystem) als auch das Zielsystem der Übertragung CDBSe eines FDBSs, das über Objektmigrationsfähigkeiten verfügt [RS94b], so können Daten und Metadaten vollständig und mit wenig Aufwand übertragen und das Quellsystem entfernt werden. Objektmigration als Dienst des FDBSs bedeutet, daß Daten sowohl mit als auch ohne zugehörige Schemainformationen von einem CDBS auf ein anderes

CDBS kopiert oder bewegt werden können. Damit können neben der schrittweisen Reduzierung von Datenbanksystemen [RS94a] weitergehende Ziele, wie z. B. der Wechsel der Klassenzugehörigkeit von Objekten über CDBSe hinweg zur Darstellung von dynamischen Zusammenhängen, erreicht werden. Ein flexibler Ansatz zur Objektmigration innerhalb eines FDBSs auf der Basis eines objektorientierten GDMs wird von [RS94b] gegeben. Dieser interessante Aspekt föderierter Datenbanksysteme wird aus Zeitgründen in dieser Arbeit nicht weiter behandelt, d. h. inwieweit der ODMG-93 Standard hierfür Unterstützung bietet, wird nicht untersucht.

Eine weitere Aufgabe, die mit föderierten Datenbanksystemen erfüllt werden kann, ist die Realisierung eines objektorientierten Datenbankmanagementsystems auf der Basis eines relationalen Datenbanksystems. Voraussetzung dafür wäre ein FDBMS mit einem geeigneten objektorientierten GDM. Ein solches System würde die Vorteile relationaler Systeme, wie die schnelle Anfrageausführung, mit denen objektorientierter Systeme, wie dauerhafte Objektidentität, verbinden. Dies wäre eine mögliche Alternative zu bisherigen Ansätzen für eine Verbindung von relationalen Datenbanksystemen mit einer objektorientierten Umgebung, wie sie z. B. in [KPP+94] beschrieben wurde.

3.2.3 Die Aufgaben des FDBMSs

Das FDBMS übernimmt in einem FDBS die Aufgabe, dem Föderationsnutzer den Zugriff und die Manipulation der von den CDBSen zur Verfügung gestellten Daten bei Überbrückung der syntaktischen und semantischen Heterogenität zu ermöglichen, ohne die Autonomie der CDBSe zu verletzen. Um die Heterogenität zu überbrücken, muß das FDBMS Schemainformationen über die von den CDBSen verwalteten Daten in der Weise speichern, daß ein FDBS-Nutzer über ein homogenes Schema eine gesamtheitliche Betrachtungsweise über die von den CDBSen zur Verfügung gestellten Daten gewinnt. Das FDBMS soll alle Fähigkeiten eines herkömmlichen Datenbankmanagementsystems besitzen, damit die normalen Datenbankfunktionen wie z. B. Anfrageausführung, Datensicherheit, Transaktionsmanagement unter Berücksichtigung der besonderen Anforderungen gewährleistet werden können.

3.3 Referenzarchitektur für föderierte Datenbanksysteme

Neben der Betrachtung der externen Realisierung der Aufgaben eines FDBSs durch das FDBMS sind Aussagen über die interne Realisierung der Aufgaben eines FDBSs notwendig. Diese interne Realisierung eines FDBS, d. h. die Verwirklichung der Integration der CDBSe, bildet den Ausgangspunkt für die Eignungsuntersuchung des ODMG-93 Objektmodells. Die hier im folgenden vorgestellte Referenzarchitektur von [SL90] als konsequente Weiterentwicklung der herkömmlichen Datenbankarchitektur eignet sich als Grundlage für die interne Realisierung. Die Referenzarchitektur gliedert sich in das 5-Schemaebenen Modell und die darauf aufbauende Systemarchitektur.

3.3.1 Das 5-Schemaebenen-Modell

[SL90] entwickelten in Anlehnung an das ANSI/X3/SPARC 3-Schichten-Modell für Datenbanken [TK78] ein 5-Schemaebenen-Modell, welches die Grundlage einer anerkannten Referenzarchitektur für die Föderation autonomer heterogener Datenbanken bildet. Das Modell basiert auf den für ein FDBS notwendigen Schemata, die in 5 Ebenen hierarchisch angeordnet sind. Es folgt eine kurze Einführung in das Modell. Die auf dem Modell basierenden Prozesse werden in nachfolgenden Abschnitten als Grundlage für die zu erarbeitenden Anforderungen an das GDM dienen.

- **Lokale Schemata und Komponentenschemata**

  Ausgangspunkt des Modells und unterste Schicht sind die konzeptionellen Schemata der CDBSe, die in diesem Zusammenhang als lokale Schemata bezeichnet werden. Durch Übersetzung der lokalen Schemata in das GDM des FDBMSs entstehen die sogenannten Komponentenschemata, wodurch die syntaktische Heterogenität überbrückt werden kann [HK95]. Dieser Prozeß der Schematransformation muß zusätzlich mit einer semantischen Anreicherung versehen werden, wenn das Datenmodell, in dem das lokale Schema erstellt wurde, nicht die Ausdrucksfähigkeiten des GDMs besitzt. Dann muß zusätzliche Semantik, die durch einen Prozeß der Wissensaneignung gewonnen werden kann, dem Komponentenschema hinzugefügt werden. Bei Verwendung eines GDMs mit geringeren Ausdrucksfähigkeiten als die Datenmodelle der CDBSe kann ein semantischer Verlust entstehen, der entsprechend aufgefangen werden muß. Die mit dem Prozeß der Schematransformation verbundenen Anforderungen an das GDM werden in Abschnitt 5.1 ausführlich behandelt.

- **Exportschemata**

  Im nächsten Prozeß werden aus den Komponentenschemata die Informationen gefiltert, die der Föderation zur Verfügung gestellt werden sollen. Ebenfalls werden alle auf diesen Informationen zugelassenen Operationen bestimmt, sowie Zugriffsrechte spezifiziert. Die daraus resultierenden Schemata werden als Exportschemata bezeichnet und entstehen durch Verhandlung zwischen den Datenbankadministratoren (DBAen) der CDBSe und dem DBA des FDBSs. Dieser Prozeß begründet sich durch die Assoziations- und Kommunikationsautonomie der CDBSe.

- **Föderierte Schemata**

  Der folgende Prozeß der Schemaintegration führt zu einem oder mehreren föderierten Schemata und dient zur Überwindung der semantischen Heterogenität. Die föderierten Schemata bieten einen einheitlichen Zugriff auf die in unterschiedlichen Datenbanksystemen verwalteten Daten. Das Schemamodell bietet zwei Möglichkeiten:

  1. ein föderiertes Schema - gesamtheitliche Sicht auf Daten
  2. mehrere föderierte Schemata - Sichten für verschiedene Nutzergruppen

  Letztere Möglichkeit ist ein Kennzeichen lose gekoppelter Systeme, während in eng gekoppelten Systemen ein einheitliches globales Schema propagiert wird. Die mit dem Prozeß der Schemaintegration verbundenen Aufgaben und die Anforderungen, die sich daraus an das GDM ergeben, werden in Abschnitt 5.2 ausführlich behandelt.

- **Externe Schemata**

  Aus der globalen Sicht der föderierten Schemata können wie im ANSI/X3/SPARC Modell externe Schemata zur Spezifizierung von Benutzersichten, zur Addition zusätzlicher Integritätsbedingungen oder zur speziellen Zugriffskontrolle abgeleitet werden. Sollen die externen Schemata in einem anderen Datenmodell als dem GDM formuliert werden, so muß das föderierte Schema, aus dem die Sicht abgeleitet werden soll, in das entsprechende Datenmodell überführt werden, d. h. eine weitere Schematransformation wäre nötig. In Abschnitt 5.3 werden mit der Ableitung externer Schemata verbundene Aufgaben behandelt.

In einer konkreten Realisierung der Referenzarchitektur können aus Redundanzgründen Schichten des Modells wegfallen. Drei Fälle sind hier möglich:

1. GDM als Datenmodell eines CDBSs - Lokales Schema = Komponentenschema

2. Export aller Daten - Komponentenschema = Exportschema
3. mehrere föderierte Schemata - Föderiertes Schema = Externes Schema

Eine denkbare Veränderung gegenüber dem Referenzmodell wäre ein Reihenfolgetausch zwischen Komponentenschema und Exportschema, der eine Aufwandsersparnis bringen würde, da nur die gefilterten Schemata transformiert werden müßten, wobei allerdings der Filterungsprozeß uneinheitlich verlaufen würde.

Die oben beschriebenen Prozesse bauen aufeinander auf und bilden zusammen einen *Bottom-Up* Entwicklungsprozeß. Dieser wird verwendet, um verschiedene Datenhaltungssysteme in einem neuen FDBS zu integrieren. Eine Remodellierung eines bestehenden FDBSs durch neue Benutzeranforderungen kann durch einen ebenfalls von [SL90] beschriebenen *Top-Down* Entwicklungsprozeß erfolgen.

Das 5-Schemaebenen-Modell gilt sowohl für lose als auch für eng gekoppelte Systeme. Abbildung 3.2 zeigt eine mögliche Entwicklung der Schemata eines FDBSs entsprechend dem Referenzmodell.

3.3.2 Systemarchitektur

Auf dem 5-Schemaebenen-Modell bauen [SL90] eine Systemarchitektur für FDBSe auf, die die grundlegenden Systemkomponenten und Abläufe innerhalb eines FDBSs darstellt, ohne eine konkrete Realisierung vorzugeben. Neben den Schemata der Schemaebenenhierarchie sind Daten, Befehle, Prozessoren und Mappings die wichtigsten Systemkomponenten. Mappings sind Funktionen, die Schemaobjekte zweier Schemata verbinden. Sie entstehen bei den Überführungsprozessen bei der Entwicklung der Schemahierarchie, so daß die folgenden drei unterschiedlichen Ausprägungen entstehen:

1. Transformationen - 1:1-Abbildung zweier Schemata,
2. Filter - Abbildung Schema - Subschema,
3. Konstruktionen - 1:n-Abbildung von Schemata.

Die Abbildungen sollten bei Transformationen und Konstruktionen eineindeutig sein, da Befehle und Daten entlang beider Richtungen der Schemahierarchie übertragen werden müssen. Die Prozessoren sind Softwaremodule, die die inneren Abläufe eines FDBSs steuern, indem sie Befehle und Daten entlang der Schemahierarchie verarbeiten. Folgende vier grundlegenden Arten von Prozessoren mit einigen ihrer Aufgaben werden in diesem Rahmen unterschieden:

1. Transformationsprozessor - Befehle und Daten in andere Datenmodelle übersetzen
2. Filterprozessor - Filtern von Befehlen und Daten, Zugriffskontrolle
3. Konstruktionsprozessor - Anfragen an föderiertes Schema zerlegen, Daten integrieren
4. Zugriffsprozessor - Zugriff auf Daten von Datenhaltungssystemen

Einige der Prozessoren benötigen die Mappings als Arbeitsgrundlage. Dafür gibt es zwei Möglichkeiten. In einem Fall werden die Mappings direkt in die Prozessoren hineincodiert, was spätere Schemaänderungen erschwert. Im zweiten Fall werden die Mappings als eigenständige Datenstruktur gespeichert, so daß die Prozessoren darauf zugreifen können.

Zusammenfassend läßt sich feststellen, daß die Referenzarchitektur sich als ein geeignetes Mittel zur Realisierung von föderierten Datenbanksystemen erwiesen hat. Dies begründet sich vor allem durch den logischen Aufbau des Schemaebenen-Modells.

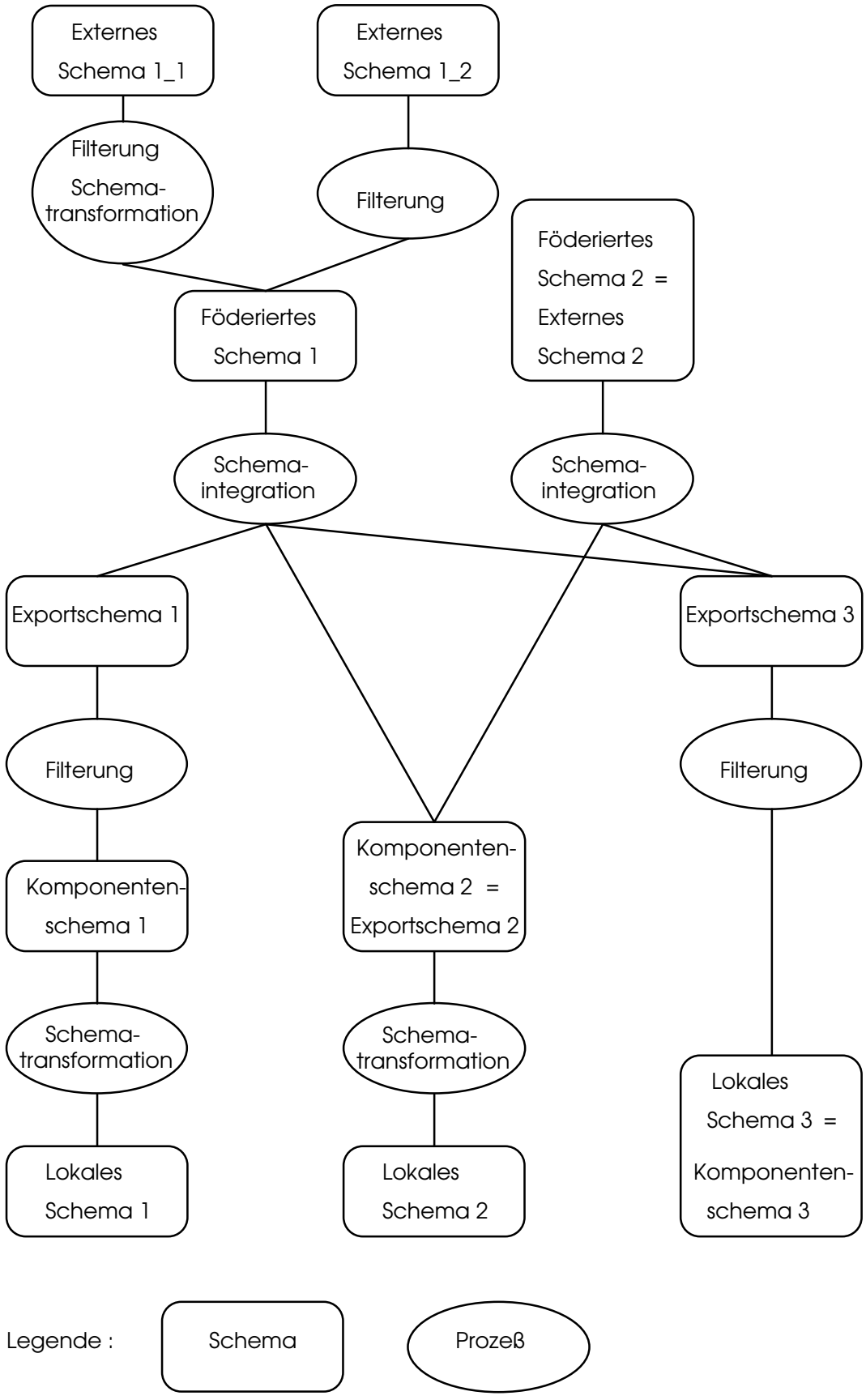

Abbildung 3.2: Entwicklung eines FDBSs nach dem 5-Schemaebenen-Modell von [SL90]

3.4 Einteilung und Einordnung föderierter Datenbanksysteme

Die Referenzarchitektur gibt nur Rahmenrichtlinien für den Aufbau eines FDBSs. Für verschiedene Einsatzzwecke können unterschiedliche Varianten von FDBSen entwickelt werden. Zwei grundsätzliche Systemkonzepte werden in diesem Abschnitt vorgestellt. Eine Einordnung der FDBSe in den Bereich der Datenbanksysteme wird ebenfalls vorgenommen.

3.4.1 Zwei Systemkonzepte föderierter Datenbanksysteme

Nach [SL90] kann man zwischen zwei Systemkonzepten für die Realisierung von föderierten Datenbanksystemen unterscheiden, den *lose gekoppelten* Systemen und den *eng gekoppelten* Systemen. Die Unterscheidung zwischen diesen Systemen erfolgt hinsichtlich des Erfüllungsgrades gegenüber den in Abschnitt 3.2.1 erläuterten Hauptaufgaben von FDBSen.

Bei lose gekoppelten Systemen übernehmen die FDBS-Nutzer die Verantwortung für die Erstellung und Aufrechterhaltung der Föderation. Die CDBSe kooperieren dabei nur soweit wie notwendig, um einen koordinierten Zugriff auf ihre Daten zu ermöglichen. Schutz der Autonomiearten steht im Vordergrund. Eine umfassende Verteilungstransparenz und die vollständige Überwindung der syntaktischen und semantischen Heterogenität wird nicht angestrebt.

Bei den eng gekoppelten Systemen übernimmt die Föderation selbst die Verantwortung für ihre Erstellung und Aufrechterhaltung. Eng gekoppelte Systeme streben eine gesamtheitliche Sicht auf die von den CDBSen bereitgestellten Daten an. Das bedeutet einerseits, daß die syntaktische und semantische Heterogenität überwunden werden muß, und andererseits, daß eine hohe Verteilungstransparenz erreicht werden muß. Um einen hohen Kooperationsumfang zu erreichen, müssen dabei Kompromisse beim Schutz der Autonomiearten eingegangen werden.

Eng gekoppelte Systeme sollten realisiert werden, wenn die Heterogenität und Verteilung der Datenbestände für den Nutzer verborgen werden soll. Lose gekoppelte Systeme könnten ihren Einsatz beim Information Retrieval über große Datenbanken auf der Basis einer schnellen Anfragesprache [Rah94] haben. Ein weiterer Gesichtspunkt bei der Auswahl eines der beiden Systemkonzepte ist die Kooperationsbereitschaft der einzelnen CDBSe.

3.4.2 Abgrenzung von anderen Datenbanksystemen

Nach [Rah94] lassen sich FDBSe von anderen Datenbanksystemen hinsichtlich der drei Grundkonzepte Verteilung, Autonomie und Heterogenität abgrenzen. Abbildung 3.3 zeigt ein Spektrum von Datenbanksystemen einschließlich lose und eng gekoppelter FDBSe entlang dieser Dimensionen. Es wird begrenzt von isolierten DBSen mit keiner Kooperation und Verteilungstransparenz und maximaler Heterogenität einerseits und verteilten und parallelen DBSen mit den umgekehrten Werten andererseits. In diesem Spektrum findet sich ein weiterer Ansatz zum Zugriff auf heterogene Datenbestände, die *Verteilten Transaktionssysteme*. Diese ermöglichen eine verteilte Transaktionsverarbeitung unter der Kontrolle von Monitoren außerhalb der heterogenen Datenbanksysteme [Rah94]. Sie bilden eine Alternative zu FDBSen, wenn ein geringer Kooperationsumfang zwischen den beteiligten DBSen angestrebt wird oder der Aufwand für die Realisierung eines FDBSs zu groß gegenüber dem erstrebten Nutzen erscheint.

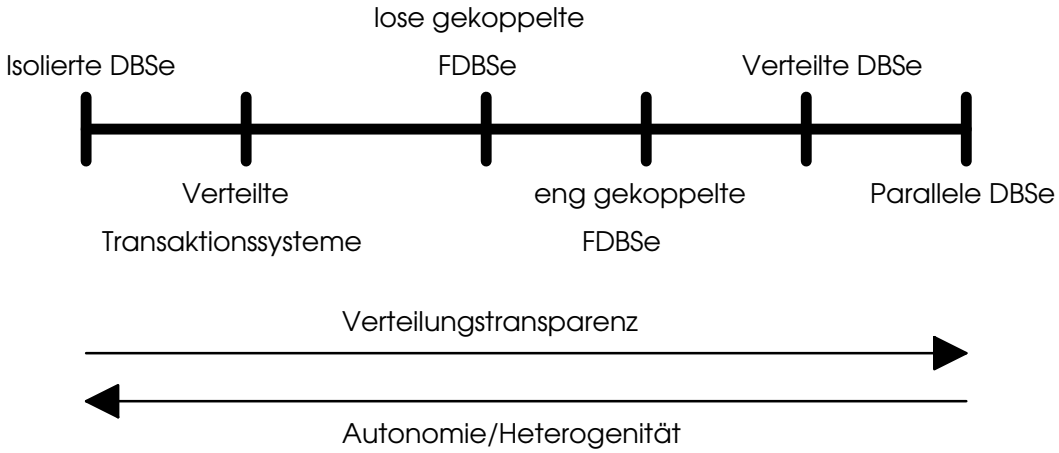

Abbildung 3.3: Einordnung von FDBSen nach [Rah94]

Eine zweite Einordnung föderierter Datenbanksysteme folgt nach [SL90]. Danach sind föderierte Datenbanksysteme eine Teilmenge der sogenannten *Multidatenbanksysteme*[1] (MDBSe), die sich von zentralisierten DBSen mit einem DBMS und einer Datenbank auf demselben Computersystem und verteilten DBSen mit einem verteilten DBMS und mehreren Datenbanken auf verschiedenen Computersystemen unterscheiden. MDBSe ermöglichen Operationen auf verschiedenen beteiligten Datenbanksystemen. Zu den MDBSen gehören noch die *Nichtföderierten Datenbanksysteme*, eine Integration nichtautonomer CDBSe. Abbildung 3.4 zeigt die Einordnung nach [SL90]. Zentralisierte DBSe entsprechen den isolierten DBSen und verteilte DBSe den verteilten und parallelen DBSen von [Rah94]. Verteilte Transaktionssysteme werden von dieser Einordnung nicht berücksichtigt.

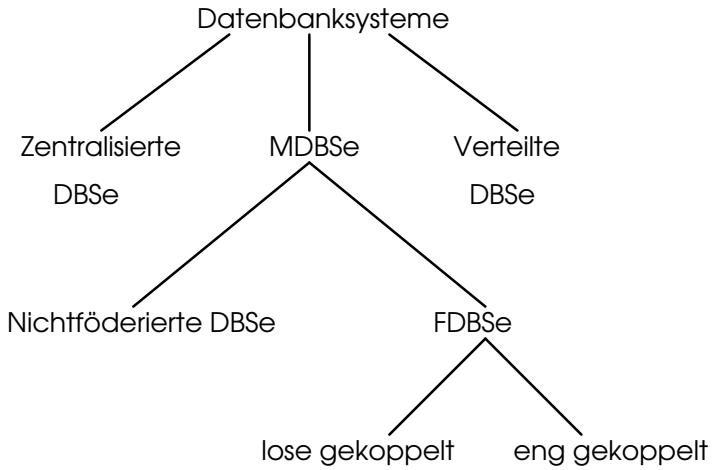

Abbildung 3.4: Einordnung von FDBSen nach [SL90]

[1] Der Begriff der Multidatenbanksysteme wird in der Literatur unterschiedlich verwendet, so bezeichnet z. B. [Rah94] lose gekoppelte FDBSe als Multi-DBSe.

4 Konflikte bei der Integration

Konflikte im Kontext der Föderierung heterogener, autonomer Datenbanksysteme sind konkrete Ausprägungen der syntaktischen und semantischen Heterogenität der CDBSe. Die Auflösung der Konflikte ebnet den Weg zu einer erfolgreichen Integration der heterogenen, autonomen Datenbanksysteme innerhalb eines FDBSs. Ein besonderes Verständnis für die Vielfalt und Komplexität der bei der Integration auftretenden Konflikte ist notwendig, um Anforderungen an ein GDM in Hinblick auf die Integrationsprozesse definieren zu können. In der Literatur [KS91, SP91, Dup94] wird versucht, Konflikte zu verallgemeinern und zu klassifizieren, um daraus Ansätze abzuleiten, die die Lösung der Konflikte erleichtern. Deshalb wird in diesem Kapitel eine Klassifizierung und Beschreibung aller der Konflikte vorgenommen, die für die nachfolgende Untersuchung der Integrationsprozesse und der Eignung des ODMG-93 Objektmodells als GDM von Bedeutung sind. Im ersten Abschnitt werden die Grundlagen zur Klassifikation durch Einführung wichtiger Begriffe gelegt, sowie eine erste Einteilung vorgenommen. In den nachfolgenden Abschnitten erfolgt die Beschreibung der durch Klassifizierung gewonnenen Konfliktarten. Zum Abschluß des Kapitels folgt eine zusammenfassende Klassifikation sowie eine Aufarbeitung der sich ergebenden Erkenntnisse.

4.1 Grundlagen der Klassifikation von Konflikten

Zur Klassifizierung von Konflikten lassen sich drei Ebenen mit ihren Ausprägungen und zugehörigen Elementen unterscheiden, die in Metabeziehung zueinanderstehen. Tabelle 4.1 zeigt die Ebenen, die Ausprägungen, die Elemente und ein Beispiel mit ODMG-93.

Ebene	Ausprägungen	Elemente	Beispiel
Datenmodell-ebene	Datenmodelle	Strukturelemente	ODMG-93 **type object**, *attribute*
Schemaebene	Schemata	Schemaelemente	Schema1: interface **person** { attribute *string name*; };
Datenbank-ebene	Datenbanken	Datenelemente	DB1: **john**{"*John Doe*"}

Tabelle 4.1: Ebenen zur Konfliktklassifikation

Die Datenbankebene ist die unterste Ebene, die Schemaebene ist die Metaebene und die Datenmodellebene die Metametaebene. Jede Ebene hat ihre Ausprägungen, und zu jeder Ausprägung gehört eine bestimmte Menge von Elementen. Die Ausprägungen der Datenmodellebene sind Datenmodelle mit den zugehörigen Strukturelementen. Die Ausprägungen der Schemaebene sind Schemata mit den zugehörigen Schemaelementen. Die Ausprägungen der Datenbankebene sind Datenbanken mit den zugehörigen Datenelementen. Die Ausprägungen und Elemente der unterschiedlichen Ebenen stehen in Instanzbeziehung. Ein Schema ist Instanz eines bestimmten Datenmodells, und zu jedem Schema gehören bestimmte Datenbanken. In gleicher Beziehung stehen die Elemente der verschiedenen Ebenen. Jedes Schemaelement ist Instanz eines bestimmten Strukturelementes, so ist *person* Instanz von *type object*. Instanzen von Schemaelementen sind bestimmte Datenelemente, so ist *john* Instanz von *person*. Die im Datenmodell festgelegten Zusammenhänge zwischen Strukturelementen werden auf die instanziierten Elemente der niederen Ebenen übertragen. So ist im ODMG-93 Datenmodell ein Attribut immer Bestandteil eines Objekttyps, das bedeutet für obiges Beispiel,

name ist an *person* genauso gekoppelt wie "*John Doe*" an *john*. Strukturelemente, die nicht Bestandteil anderer Strukturelemente sind, bezeichnet man als Basisstrukturelemente. Im ODMG-93 Objektmodell ist der Objekttyp das einzige Basisstrukturelement.

Zum weiteren Verständnis für Konflikte werden die Begriffe des Realweltkonzepts und der Realweltklasse eingeführt. Unter einem Realweltkonzept versteht man die Bedeutung eines Schemaelementes in der realen Welt. Eine Realweltklasse repräsentiert die Menge von Objekten der realen Welt, die durch ein Schemaelement abstrahiert wird. Diese Menge kann, muß aber nicht, mit der Menge aller Instanzen dieses Schemaelementes auf Datenebene übereinstimmen. Ein weiterer an dieser Stelle einzuführender Begriff ist der der semantischen Vergleichbarkeit. Schemaelemente unterschiedlicher Schemata sind semantisch vergleichbar, wenn sie Realweltkonzepte abbilden, die entweder äquivalent oder sich einschließende, überlagernde oder disjunkte Teilkonzepte eines übergeordneten Realweltkonzepts sind.

Drei grundsätzliche Arten von Konflikten entsprechend des Zeitpunktes ihrer Lösung lassen sich unterscheiden:

- **Datenmodellkonflikt** - Schematransformation,
- **Schemakonflikte** - Schemaintegration,
- **Datenkonflikte** - Datenintegration.

Diese erste Einteilung der Konflikte beruht also auf den drei Prozessen, die die entscheidende Rolle bei der Integration spielen.

Der Datenmodellkonflikt ist eine Ausprägung der syntaktischen Heterogenität der CDBSe. Er entsteht, wenn die konzeptionellen Schemata der CDBSe mit unterschiedlichen Datenmodellen modelliert werden, d. h. die Schemata sind Instanzen verschiedener Datenmodelle. Dieser Konflikt wird nach dem 5-Schemaebenen-Modell durch die Transformation der lokalen Schemata in das GDM gelöst, so daß die entstehenden Schemata Instanzen eines einzigen Datenmodells sind. Der Datenmodellkonflikt wird nicht weiter klassifiziert.

Schema- und Datenkonflikte sind Ausprägungen der semantischen Heterogenität. Sie entstehen, wenn sich die Diskursbereiche (UoD - *universe of discours*) der konzeptionellen Schemata der CDBSe überlappen [SP91]. Sie werden in den nachfolgenden Abschnitten weiter klassifiziert.

4.2 Schemakonflikte

Schemakonflikte bestehen zwischen semantisch vergleichbaren Schemaelementen der zu integrierenden Exportschemata, wenn für die Abstraktion und Modellierung der entsprechenden Realweltkonzepte unterschiedliche Mittel und Methoden verwendet wurden. Nach dem 5-Schemaebenen-Modell werden Schemakonflikte während der Schemaintegration gelöst, so daß alle in den Exportschemata abgebildeten Realweltkonzepte durch Schemaelemente eines föderierten Schemas repräsentiert werden. Nachfolgend wird ein Klassifikationsansatz für Schemakonflikte entwickelt. Basierend auf diesem Ansatz werden in den darauffolgenden Abschnitten die Schemakonflikte beschrieben.

4.2.1 Klassifikationsansatz

Nach [SP91, Dup94] lassen sich folgende vier Grundtypen von Schemakonflikten unterscheiden:

- **Namenskonflikte,**

- **Semantische Konflikte,**
- **Beschreibungskonflikte,**
- **Strukturkonflikte.**

Zwischen Schemaelementen verschiedener Schemata können Schemakonflikte aller vier Grundtypen gleichzeitig auftreten. Die Mehrzahl der Schemakonflikte ist unabhängig vom verwendeten GDM. Einige Schemakonflikte treten jedoch nur bei bestimmten Datenmodellen auf, sie entstehen durch spezifische Ausdrucksfähigkeiten der Datenmodelle.

Die weitere Klassifikation und Beschreibung aller vier Grundtypen von Schemakonflikten in den folgenden Abschnitten erfolgt auf der Basis des ODMG-93 Objektmodells, d. h. es wird versucht, alle datenmodellunabhängigen Schemakonflikte sowie alle für das ODMG-93 Objektmodell spezifischen Schemakonflikte zu klassifizieren. Alle verwendeten Beispiele benutzen entweder die in Abschnitt 2.2 eingeführte graphische Notation des Objektmodells oder basieren auf der in Abschnitt 2.3 eingeführten Object Definition Language. Wie in der Literatur [SP91, Dup94] üblich werden nur Schemakonflikte zwischen genau zwei unterschiedlichen Schemata betrachtet.

Schemakonflikte sind sehr komplex und vielfältig; deshalb ist es notwendig, Hilfsmittel zur abstrakten Beschreibung von Schemakonflikten zu verwenden. Auf der Basis einer solchen Beschreibung kann die Konfliktlösung während der Schemaintegration erleichtert werden. Ein solches Hilfsmittel sind die sogenannten *correspondence assertions* wie sie in [SP91, SP92, SPD92, Dup94] verwendet werden. Diese beschreiben semantische Konflikte als Mengenvergleich der zugehörigen Realweltklassen und Beschreibungskonflikte als Wertebereichsvergleich von Attributen und Beziehungen. Zur Beschreibung von Schemakonflikten wird in den folgenden Abschnitten ein an die *correspondence assertions* angelehntes Modell eingeführt. Dazu werden jeweils Formeln für die Grundtypen aufgestellt, erläutert und in Beispielen verwendet. Für Namenskonflikte wird keine besondere Formel verwendet. Es wird die Grundannahme getroffen, daß zwei Exportschemata ES_1 und ES_2, ausgedrückt im ODMG-93 Objektmodell, existieren.

4.2.2 Namenskonflikte

Namenskonflikte, das sind Konflikte zwischen den Bezeichnungen von Schemaelementen unterschiedlicher Schemata, sind die einfachsten Schemakonflikte, die sich in folgende zwei elementare Arten teilen lassen:

- **Synonyme** - Unterschiedliche Namen für semantisch vergleichbare Schemaelemente,
- **Homonyme** - Derselbe Name für semantisch nicht vergleichbare Schemaelemente.

Diese Konflikte können alle Schemaelemente betreffen, die durch den Modellierer mit einer Bezeichnung versehen werden können. Das anschließende Beispiel zeigt sowohl Synonyme als auch Homonyme.

Beispiel 4.1: Zwei Schemata ES_1 und ES_2 in ODMG-93 ODL:

```
interface person {                interface person {
  attribute string name;            attribute string name;
  attribute date geb_dat;           attribute date geboren_am;
  attribute bool partner; };        attribute string partner; };
```

Bei den Attributen *geb_dat* und *geboren_am* handelt es sich offenbar um Synonyme, während die Attribute *partner* Homonyme sind, da das boolesche Attribut angibt, ob eine Person einen Partner hat oder nicht, während das stringwertige Attribut den Namen eines Partners enthält.

4.2.3 Semantische Konflikte

Semantische Konflikte existieren zwischen semantisch vergleichbaren Schemaelementen, wenn die Extensionen der zugehörigen Realweltklassen nicht übereinstimmen. Das Erkennen von semantischen Konflikten erfordert genaue Informationen über den von den einzelnen Schemata abgebildeten Weltausschnitt. Ausprägungen von semantischen Konflikten werden zwischen Instanzen von Basisstrukturelementen, das bedeutet für ODMG-93 zwischen Objekttypen, betrachtet.

Zum Aufstellen einer Formel zur Darstellung semantischer Konflikte treffen wir folgende Annahmen, die die Allgemeingültigkeit nicht beschränken. X sei ein beliebiger Objekttyp in ES_1 und Y ein beliebiger Objekttyp in ES_2, beide seien semantisch vergleichbar. Dann gilt folgende Formel:

(1) X *mop* Y; mit *mop* $\in (\equiv_S, \subseteq_S, \supseteq_S, \cap_S, \varnothing_S)$.

Für jedes Paar semantisch vergleichbarer Objekttypen verschiedener Schemata läßt sich durch die Mengenvergleichsoperatoren *mop* darstellen, ob sie semantisch äquivalent ($\equiv_S$) sind, wenn die Extensionen der zugehörigen Realweltklassen übereinstimmen, oder ob zwischen ihnen einer der folgenden Typen semantischer Konflikte existiert:

- **Inklusion** - Extensionen stehen in Teilmengenbeziehung ($\subseteq_S$, $\supseteq_S$),
- **Überlappung** - Extensionen überlappen sich teilweise ($\cap_S$),
- **Disjunktheit** - Extensionen sind disjunkt ($\varnothing_S$).

Die folgenden drei Beispiele verdeutlichen die drei Typen semantischer Konflikte sowie den Unterschied zwischen Realweltkonzept und Realweltklasse.

Beispiel 4.2: Zwei Schemata ES_1 und ES_2 in ODMG-93 ODL:

```
interface student {..};        interface student {..};
```

Schema ES_1 bildet eine Universität ab, während Schema ES_2 eine Fakultät derselben Universität abbildet. Daraus ergibt sich folgender Konflikt:
ES_1.Student $\supseteq_S$ ES_2.Student.

Beispiel 4.3: Zwei Schemata ES_1 und ES_2 in ODMG-93 ODL:

```
interface kunde {..};          interface kunde {..};
```

Beide Schemata modellieren unterschiedliche Firmen, die übereinstimmende Kunden aufweisen können, woraus sich folgender Konflikt ergibt:
ES_1.Kunde $\cap_S$ ES_2.Kunde.

Beispiel 4.4: Zwei Schemata ES_1 und ES_2 in ODMG-93 ODL:

```
interface personal {..};       interface personal {..};
```

Beide Schemata modellieren unterschiedliche Firmen, woraus sich folgender Konflikt ergibt:
ES_1.Personal $\varnothing_S$ ES_2.Personal.

In jedem der drei Beispiele wird von den beteiligten Schemaelementen dasselbe Realweltkonzept abgebildet, so in Beispiel 4.3 das Konzept "Kunde einer Firma". Die zugehörigen Realweltklassen ergeben sich aus Erkenntnissen über die zugrundeliegenden Schemata.

Semantische Konflikte können nicht nur zwischen genau zwei Objekttypen auftreten, sondern auch zwischen einem Objekttyp eines Schemas und einer Gruppe von Objekttypen eines anderen Schemas (1:n) sowie zwischen Gruppen von Objekttypen verschiedener Schemata (m:n), jedoch lassen sie sich alle auf eine Menge von einzelnen Konflikten zwischen

genau zwei Objekttypen zurückführen. Diese erweiterten Konflikte ergeben sich dadurch, daß Realweltkonzepte beliebig in Teilkonzepte zerlegt sein können. Nach [Dup94] werden sie als **semantische Fragmentierung** bezeichnet. Um eine optimale Schemaintegration zu erreichen, sollte diese Fragmentierung erkannt werden. Für die Darstellung dieser komplexen Konflikte treffen wir folgende Annahmen. $X_1..X_m$ seien Objekttypen in ES_1, die mit Objekttypen $Y_1..Y_n$ in ES_2 semantisch vergleichbar sind. Dann läßt sich die semantische Fragmentierung gemäß folgender Formel darstellen:

(2) $(X_1\ op_{11}\ X_2\ ..\ X_{m-1}\ op_{1m-1}\ X_m)\ mop\ (Y_1\ op_{21}\ Y_2\ ..\ Y_{n-1}\ op_{2n-1}\ Y_n)$;
mit $op_{1i}, op_{2j} \in (\cup, \setminus)$, i := 1..m-1, j := 1..n-1; $mop \in (\equiv_S, \subseteq_S, \supseteq_S, \cap_S)$.

Die Ausdrücke $\cup$ und \ bezeichnen Operationen zum Vereinigen ($\cup$) und Zerlegen (\) von Realweltkonzepten und Realweltklassen von Objekttypen, um komplexe Zusammenhänge innerhalb einer Gruppe von Objekttypen herzustellen. Zum Erkennen von semantischer Fragmentierung und dem Darstellen semantischer Konflikte zwischen Gruppen von Objekttypen gemäß Formel (2) geht man gemäß folgenden Schritten vor:

1. Finde eine Äquivalenz ($\equiv_S$) und stelle sie dar, sonst
2. finde eine Inklusion ($\subseteq_S$, $\supseteq_S$) und stelle sie dar, sonst
3. finde eine Überlappung ($\cap_S$) und stelle sie dar, sonst
4. stelle jeweils semantische Konflikte zwischen genau zwei Objekttypen dar.

Das Finden bedeutet die optimale Anwendung der Operationen $\cup$ und \. Die Arten semantischer Fragmentierung hängen von den Fähigkeiten der Datenmodelle ab, Realweltkonzepte sowohl disjunkt als auch nicht disjunkt zerlegt abbilden zu können. Mit dem ODMG-93 Objektmodell lassen sich demnach folgende zwei Arten von semantischer Fragmentierung erkennen, die im Anschluß genauer beschrieben werden:

- **Klassifikationskonflikt [Dup94],**
- **Vererbungskonflikt.**

Beim Klassifikationskonflikt ergibt sich die semantischer Fragmentierung dadurch, daß ein Realweltkonzept durch eine Gruppe disjunkter Teilkonzepte abgebildet wird. Diese Fragmentierung läßt sich erkennen, indem man die Realweltkonzepte semantisch und die Realweltklassen von Objekttypen mengenmäßig vereinigt. In der Formel (2) wird dies durch die Vereinigungsoperation $\cup$ ausgedrückt. Das folgende Beispiel zeigt einen Klassifikationskonflikt sowie die zugrundeliegenden semantischen Konflikte.

Beispiel 4.5: Zwei Schemata ES_1 und ES_2 in ODMG-93 ODL:

```
interface personal {..};      interface angestellte {..};
                              interface arbeiter {..};
```

Beide Schemata modellieren dieselbe Firma. Folgender Klassifikationskonflikt ergibt sich:
ES_1.Personal $\equiv_S$ (ES_2.Angestellte $\cup$ ES_2.Arbeiter).
Folgende semantischen Konflikte liegen zugrunde:
ES_1.Personal $\supseteq_S$ ES_2.Angestellte und ES_1.Personal $\supseteq_S$ ES_2.Arbeiter.

Eine besondere Form der semantischen Fragmentierung kann sich durch die Möglichkeiten des Vererbungsprinzips ergeben, das auch im ODMG-93 Objektmodell Anwendung findet. Bei der Vererbung ergibt sich die semantische Fragmentierung derart, daß ein Realweltkonzept sowohl in disjunkte Teilkonzepte zerlegt sein kann, als auch selbst Teil eines größeren Konzeptes sein kann. Zum Erkennen des letzteren muß man Realweltkonzepte semantisch in

Teilkonzepte zerlegen und die Realweltklassen subtrahieren. Dazu dient in Formel (2) die Differenzoperation \. Das folgende Beispiel zeigt einen Vererbungskonflikt sowie die zugrundeliegenden semantischen Konflikte.

Beispiel 4.6: Abbildung 4.1 zeigt zwei Schemata ES_1 und ES_2 in graphischer Notation des ODMG-93 Objektmodells:

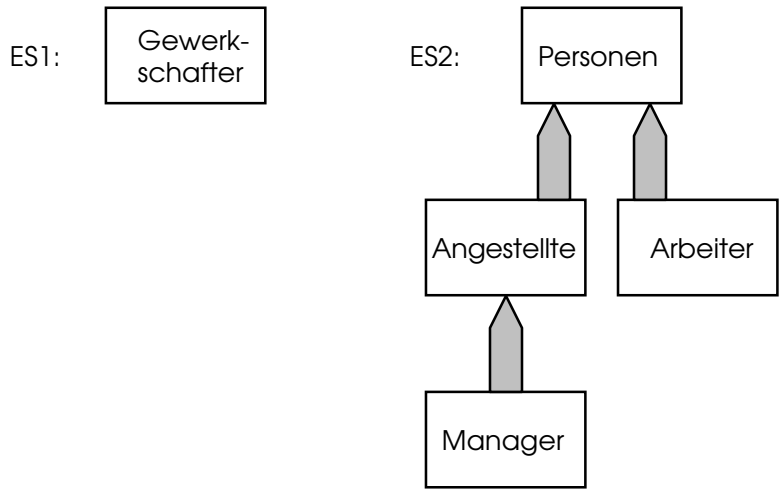

Abbildung 4.1: Beispielschemata in ODMG-93

Schema ES_2 modelliert das Personal einer Firma, während Schema ES_1 die Gewerkschaftsmitglieder derselben Firma abbildet. Es ergibt sich folgender Vererbungskonflikt:
ES_1.Gewerkschafter $\subseteq_S$ ((ES_2.Angestellte \ ES_2.Manager) $\cup$ ES_2.Arbeiter).
Folgende semantische Konflikte liegen zugrunde:
ES_1.Gewerkschafter $\subseteq_S$ ES_2.Angestellte, ES_1.Gewerkschafter $\subseteq_S$ ES_2.Arbeiter und
ES_1.Gewerkschafter $\varnothing_S$ ES_2.Manager.

4.2.4 Beschreibungskonflikte

Beschreibungskonflikte zwischen semantisch vergleichbaren Objekttypen entstehen, wenn die dazugehörigen Eigenschaftsmengen sowie die Eigenschaften untereinander nicht übereinstimmen. Die Ursache für Beschreibungskonflikte sind einerseits unterschiedliche Blickwinkel der Modellierer auf denselben Weltausschnitt und andererseits die Verwendung unterschiedlicher Darstellungsmittel. Während semantische Konflikte den extensionalen Teil einer Beziehung zwischen semantisch vergleichbaren Objekttypen abbilden, stellen Beschreibungskonflikte den intensionalen Teil dar.

Beschreibungskonflikte werden zwischen genau zwei semantisch vergleichbaren Objekttypen und den in ihnen enthaltenen Eigenschaften dargestellt. Um eine Formel zur Darstellung von Beschreibungskonflikten aufzustellen, treffen wir folgende Annahmen. Es existieren zwei semantisch vergleichbare Objekttypen ES_1.X{$x_1..x_f$, $rx_1..rx_g$, $kx_1..kx_h$} und $ES2$.Y{$y_1..y_u$, $ry_1..ry_v$, $ky_1..ky_w$} mit jeweils allen Attributen ($x_1..x_f$, $y_1..y_u$), Beziehungen ($rx_1..rx_g$, $y_1..ry_v$) und Schlüsseln ($kx_1..kx_h$, $ky_1..ky_w$). Beschreibungskonflikte können gemäß folgender Formel dargestellt werden:

(3) X *bop* Y
{ $x_i \equiv_A (y_{j1} + y_{j2} .. y_{jt-1} + y_{jt})$;
x_i *aop* y_j;
$rx_k \leftrightarrow_{RK} ry_l$;

$x_i \leftrightarrow_{AM} Y$
$kx_1..kx_h \leftrightarrow_{SB} ky_1..ky_w$ };
mit *bop* ∈ ($\equiv_B$, $\subseteq_B$, $\supseteq_B$, $\cap_B$, $\varnothing_B$); *aop* ∈ ($\equiv_A$, $\leftrightarrow_{AI}$, $\leftrightarrow_{AW}$, $\leftrightarrow_{AL}$);
i ∈ (1..f); j, j1, j2, .. jt-1, jt ∈ (1..u); k ∈ (1..g); l ∈ (1..v).

Anhand der Strukturelemente, die im ODMG-93 Objektmodell einen Objekttyp bilden, lassen sich folgende fünf Typen von Beschreibungskonflikten ermitteln, die durch entsprechende Vergleichsoperatoren in der Formel (3) dargestellt werden:

- **Eigenschaftsmengenkonflikte** (*bop*),
- **Attributkonflikte** (*aop*),
- **Kardinalitätskonflikt zwischen Beziehungen** ($\leftrightarrow_{RK}$),
- **Attribut versus Metainfo** ($\leftrightarrow_{AM}$),
- **Schlüsselkonflikt** ($\leftrightarrow_{SB}$).

Im folgenden werden diese Typen weiter klassifiziert und beschrieben.

Durch Vergleich der Eigenschaftsmengen, das sind die Mengen aller Attribute und Beziehungen, kann für alle Paare von semantisch vergleichbaren Objekttypen festgestellt und mit Hilfe der Operatoren *bop* dargestellt werden, ob sie beschreibungsäquivalent ($\equiv_B$) sind, wenn alle Eigenschaften übereinstimmen, oder ob sie in einem der drei folgenden Eigenschaftsmengenkonflikte stehen:

- **Inklusion** - Eigenschaften stehen in Teilmengenbeziehung ($\subseteq_B$, $\supseteq_B$),
- **Überlappung** - Eigenschaften überlappen sich teilweise ($\cap_B$),
- **Disjunktheit** - Eigenschaften sind disjunkt ($\varnothing_B$).

Eigenschaften stimmen überein, wenn sie semantisch vergleichbar sind. Diese Eigenschaften werden als gemeinsame Eigenschaften der beiden Objekttypen bezeichnet. Die folgenden drei Beispiele verdeutlichen die drei Fälle von Eigenschaftsmengenkonflikten.

Beispiel 4.7: Zwei Schemata ES_1 und ES_2 in ODMG-93 ODL:

```
interface buch {                interface buch {
  attribute string titel;         attribute string titel;
  attribute string autor;         attribute string autor; };
  attribute string verlag; };
```

Folgender Konflikt ergibt sich: ES_1.Buch $\supseteq_B$ ES_2.Buch.

Beispiel 4.8: Zwei Schemata ES_1 und ES_2 in ODMG-93 ODL:

```
interface auto {                interface auto {
  attribute string typ;           attribute string typ;
  relationship firma prod; };     attribute integer ps; };
```

Folgender Konflikt ergibt sich: ES_1.Auto $\cap_B$ ES_2.Auto.

Beispiel 4.9: Zwei Schemata ES_1 und ES_2 in ODMG-93 ODL:

```
interface student {               interface student {
  attribute string name;            attribute integer st_nr;
  relationship Set<vorl> hört };  attribute string fach };
```

Folgender Konflikt ergibt sich: ES_1.Student $\varnothing_B$ ES_2.Student.

Zwischen gemeinsamen Attributen von zwei Objekttypen können Attributkonflikte auftreten, die nach [Dup94] auch als technische Konflikte bezeichnet werden. Diese ergeben sich dadurch, daß die Modellierer einerseits unterschiedliche Techniken zur Darstellung von Attributen von Objekttypen verwenden und andererseits Attributwerte unterschiedlich berechnet und interpretiert werden. Vier Arten von Attributkonflikten werden klassifiziert. Es folgt eine kurze Beschreibung der Konfliktarten, die in Formel (3) durch die Attributvergleichsoperatoren *aop* dargestellt werden, mit anschließendem Beispiel.

- **1 Attribut versus n Attribute**

  Bei diesem Konflikt ist ein Attribut des einen Objekttyps äquivalent ($\equiv_A$) zu einer Kombination (+ Operator) von n Attributen des anderen Objekttyps, d. h. durch Zusammenfügung der Wertebereiche der n Attribute ergibt sich der Wertebereich des einen Attributs. Die Art der Zusammenfügung hängt von der Art der verwendeten Literaltypen ab.

- **Interpretationskonflikt**

  Bei diesem Konflikt haben zwei semantisch vergleichbare Attribute denselben Literaltyp als Wertebereich, die Attributwerte der Objekte werden jedoch in den einzelnen Schemata unterschiedlich interpretiert oder berechnet ($\leftrightarrow_{AI}$). Um Ausprägungen dieses Konflikts zu erkennen, müssen weitergehende Informationen über die Semantik der beteiligten Attribute gewonnen werden. Eine der bekanntesten Ausprägungen dieses Konflikts ist das sogenannte *meal-cost* Problem [GB91] (siehe Beispiel 4.10). Weitere Differenzierungen bei dieser Konfliktart wären möglich, sind aber für die weitere Untersuchung nicht notwendig.

- **Wertebereichskonflikt**

  Bei diesem Konflikt haben zwei semantisch vergleichbare Attribute denselben Literaltyp als Wertebereich, jedoch sind die Wertebereiche unterschiedlich begrenzt. ($\leftrightarrow_{AW}$).

- **Literaltypenkonflikt**

  Bei diesem Konflikt haben zwei semantisch vergleichbare Attribute unterschiedliche Literaltypen als Wertebereiche ($\leftrightarrow_{AL}$).

Beispiel 4.10: Zwei Schemata ES_1 und ES_2 in ODMG-93 ODL:

```
interface adresse {              interface adresse {
  attribute string<30> str;        attribute string<50> str;
  attribute string nr;             attribute integer nr;
  attribute string plz_ort; };     attribute string plz;
                                   attribute string ort; };
interface restaurant {           interface restaurant {
  attribute float kosten; };       attribute float kosten; };
```

Als zusätzliche Information ist bekannt, daß das Attribut *kosten* in ES_1 die Durchschnittskosten einer Mahlzeit inklusive Steuer und Trinkgeld abbildet, während das vergleichbare Attribut mit demselben Namen in ES_2 die Durchschnittskosten ohne Steuer und Trinkgeld abbildet. Folgende Konflikte ergeben sich:

ES_1.adresse $\equiv_B$ ES_2.adresse

{str $\leftrightarrow_{AW}$ str; nr $\leftrightarrow_{AL}$ nr; plz_ort $\equiv_A$ (plz + ort)};

ES_1.restaurant $\equiv_B$ ES_2.restaurant

{preis $\leftrightarrow_{AI}$ preis}.

Zwischen gemeinsamen Beziehungen von zwei Objekttypen kann ein Kardinalitätskonflikt ($\leftrightarrow_{RK}$) auftreten, d. h. eine Realweltbeziehung wird in einem Schema einwertig und in einem anderen Schema mehrwertig abgebildet. Das folgende Beispiel verdeutlicht diesen Konflikt.

Beispiel 4.11: Zwei Schemata ES_1 und ES_2 in ODMG-93 ODL:

```
interface auto {                    interface auto {
  relationship                        relationship
    person besitzer; };                 Set<person> besitzer; };
```

Folgender Konflikt ergibt sich: ES_1.auto $\equiv_B$ ES_2.auto
{besitzer $\leftrightarrow_{RK}$ besitzer}.

Eine besonderer Typ eines Beschreibungskonfliktes ist der Konflikt zwischen Attribut und Metainfo. Dieser Konflikt entsteht durch die unterschiedliche Beschreibung von Realweltkonzepten an sich, d. h. während ein Realweltkonzept in einem Schema durch den Namen des zugehörigen Objekttyps beschrieben wird, wird es im anderen Schema durch den Wert eines Attributs eines Objekttyps, in dem es ein Teilkonzept eines übergeordneten Realweltkonzepts abbildet, beschrieben ($\leftrightarrow_{AM}$). Dieses Attribut wird nach [HK95] als Diskriminante bezeichnet. Das folgende Beispiel zeigt eine Ausprägung dieses Konfliktes, die sich im Zusammenhang mit dem Klassifikationskonflikt aus Beispiel 4.5 ergeben kann.

Beispiel 4.12: Zwei Schemata ES_1 und ES_2 in ODMG-93 ODL:

```
interface personal {                  interface angestellte {..};
  attribute enum{ang,arb} a/a; }; interface arbeiter {..};
```

Folgende Konflikte ergeben sich: ES_1.personal *bop* ES_2.angestellte
{a/a $\leftrightarrow_{AM}$ *angestellte*},
ES_1.personal *bop* ES_2.arbeiter
{a/a $\leftrightarrow_{AM}$ *arbeiter*}.

Heutige Datenmodelle erlauben die Definition von Integritätsbedingungen über Schemaelemente verschiedener Größe. Neben einfachen Bedingungen auf Wertebereichs- und Attributebene gibt es auch komplexe Bedingungen über mehrere Entitytypen hinweg. Zwei Möglichkeiten zur Definition von Integritätsbedingungen sind prinzipiell möglich. Zum einen können Integritätsbedingungen prozedural formuliert werden, so im ODMG-93 Objektmodell über die zu den Objekttypen gehörenden Operationen. Daraus möglicherweise entstehende Konflikte werden an dieser Stelle aufgrund der großen Komplexität nicht behandelt. Zum zweiten bieten viele Datenmodelle Möglichkeiten zur deklarativen Definition von Integritätsbedingungen. Die einzige im ODMG-93 Objektmodell hierfür vorgesehene Möglichkeit ist die Definition von Schlüsselbedingungen für Objekttypen, die eine eindeutige Identifikation eines Objekts innerhalb der Extension über die Werte der Eigenschaften, die in einem Schlüssel des Objekttyps definiert wurden, erlauben. Zwischen zwei semantisch vergleichbaren Objekttypen kann sich daher ein Schlüsselkonflikt, der durch die Definition unterschiedlicher Schlüsselbedingungen entsteht, ergeben ($\leftrightarrow_{SB}$). Dieser Konflikt kann verschiedene Formen annehmen. So kann eine Eigenschaft eines Objekttyps ein einzelner Schlüssel sein, während die semantisch korrespondierende Eigenschaft eines anderen Objekttyps kein Schlüssel oder Teil eines zusammengesetzten Schlüssels ist.

Beispiel 4.13: Zwei Schemata ES_1 und ES_2 in ODMG-93 ODL:

```
interface buch {                    interface buch {
  key(isbn);                          key(titel,autor);
  attribute string isbn;              attribute string titel;
  attribute string titel; };          attribute string autor; };
```

Folgender Konflikt ergibt sich: $ES_1.\text{buch} \cap_B ES_2.\text{buch}$
$\{\text{key(isbn)} \leftrightarrow_{SB} \text{key(titel,autor)}\}$.

4.2.5 Strukturkonflikte

Strukturkonflikte ergeben sich, wenn äquivalente Realweltkonzepte durch Schemaelemente abgebildet werden, die Instanzen unterschiedlicher Strukturelemente sind. Daher sind die Arten von möglichen Strukturkonflikten abhängig von den Strukturelementen, die die Datenmodelle zur Verfügung stellen. Im folgenden werden die Strukturkonflikte beschrieben, die mit dem ODMG-93 Objektmodell auftreten können.

Mit Instanzen von drei Strukturelementen des ODMG-93 Objektmodells können Realweltkonzepte abgebildet werden, mit Objekttypen, Attributen und Beziehungen. Daraus ergeben sich folgende drei mögliche Arten von Strukturkonflikten:

- **Objekttyp versus Attribut(e),**
- **Beziehung versus Attribut(e),**
- **Objekttyp/Beziehungs-Konflikt.**

Die ersten beiden Arten sind untrennbar miteinander verbunden, so daß sie nachfolgend im Zusammenhang beschrieben werden.

Zur Darstellung der drei Strukturkonflikttypen mit Hilfe von Formeln werden folgende Annahmen getroffen. Es existiere in ES_1 ein Objekttyp X_a sowie ein Objekttyp $X_b\{rx_{ba}\}$ mit einer Beziehung rx_{ba} zu X_a. In ES_2 existiere ein mit X_b semantisch vergleichbarer Objekttyp $Y\{y_1..y_k\}$ mit bestimmten Attributen $y_1..y_k$, die dasselbe Realweltkonzept wie X_a abbilden. Desweiteren existieren die jeweils semantisch vergleichbaren Objekttypen $ES_1.X_c$, $ES_2.Y_p$ und $ES_1.X_d$, $ES_2.Y_q$. In ES_1 existiere ein Objekttyp $X_e\{rx_{ec}, rx_{ed}\}$ mit Beziehungen rx_{ec} zu X_c und rx_{ed} zu X_d. In ES_2 existiere eine Beziehung $Y_p\{ry_{pq}\}/Y_q\{ry_{qp}\}$. Dann gelten folgende Formeln:

(4) $X_a \leftrightarrow_{SOA} Y_p\{y_{p1} .. y_{pz}\}$;

(5) $X_b\{rx_{ba}\} \leftrightarrow_{SRA} Y_p\{y_{p1} .. y_{pz}\}$;

(6) $Xe\{rx_{ec}, rx_{ed}\} \leftrightarrow_{SOR} Y_p\{ry_{pq}\}/Y_q\{ry_{qp}\}$.

Der Konflikt Objekttyp versus Attribut(e) entsteht, wenn ein Realweltkonzept in einem Schema als Objekttyp und in einem anderen Schema durch ein oder mehrere Attribute abgebildet wird. Dabei können diese Attribute mit Attributen des Objekttyps semantisch vergleichbar sein. Dieser Konflikt kann gemäß Formel (4) dargestellt werden. Semantische und Eigenschaftsmengenkonflikte zwischen Objekttyp und den Attributen lassen sich fixieren, indem die Attribute zu einem virtuellen Objekttyp vereinigt werden. In Zusammenhang mit der Existenz eines solchen Konfliktes kann ein Konflikt Beziehung versus Attribut(e) bestehen. Dieser Konflikt entsteht immer dann, wenn in zwei semantisch vergleichbaren Objekttypen ein Realweltkonzept zum einen durch eine Beziehung und zum anderen durch ein oder mehrere Attribute ausgedrückt wird. Wenn die Attribute und die Beziehungen mengenwertig sind, spricht man nach [Dup94] von einem Aggregationskonflikt. Die Darstellung dieses Konfliktes kann gemäß Formel (5) erfolgen. Das folgende Beispiel verdeutlicht den Zusammenhang zwischen beiden Konflikten.

Beispiel 4.14: Zwei Schemata ES_1 und ES_2 in ODMG-93 ODL:

```
interface buch {                interface buch {
  attribute string titel;         attribute string titel;
  attribute string autor;         attribute string verlag; };
```

```
  relationship verlag verlegt_von
    inverse verlag::verlegte_bücher };
interface verlag {
  attribute string vname;
  relationship Set<buch> verlegte_bücher
    inverse buch::verlegt_von };
```

Folgende Strukturkonflikte ergeben sich:
ES_1.verlag $\leftrightarrow_{SOA}$ ES_2.buch{verlag}; ES_1.buch{verlegt_von} $\leftrightarrow_{SRA}$ ES_2.buch{verlag}

Ein Objekttyp/Beziehungs-Konflikt entsteht, wenn Realweltbeziehungen zwischen zwei Realweltkonzepten in einem Schema durch eine Beziehung zwischen den die Realweltkonzepte abbildenden Objekttypen und in einem anderen Schema durch einen eigenen Objekttyp mit Beziehungen zu den entsprechenden Objekttypen dargestellt werden. Bei Darstellung von Realweltbeziehungen mittels eines Objekttyps ergibt sich die Möglichkeit, zusätzliche beschreibende Attribute hinzuzufügen, was bei einer reinen Beziehung nicht möglich ist. Der daraus möglicherweise resultierende Eigenschaftsmengenkonflikt sowie ein existierender semantischer Konflikt läßt sich fixieren, indem die Beziehung einem virtuellen Objekttyp zugeordnet wird. Dieser Konflikt kann gemäß Formel (6) dargestellt werden. Das folgende Beispiel zeigt einen Konflikt zwischen einem Objekttyp und einer Beziehung.

Beispiel 4.15: Abbildung 4.2 zeigt zwei Schemata ES_1 und ES_2 in graphischer Notation des ODMG-93 Objektmodells:

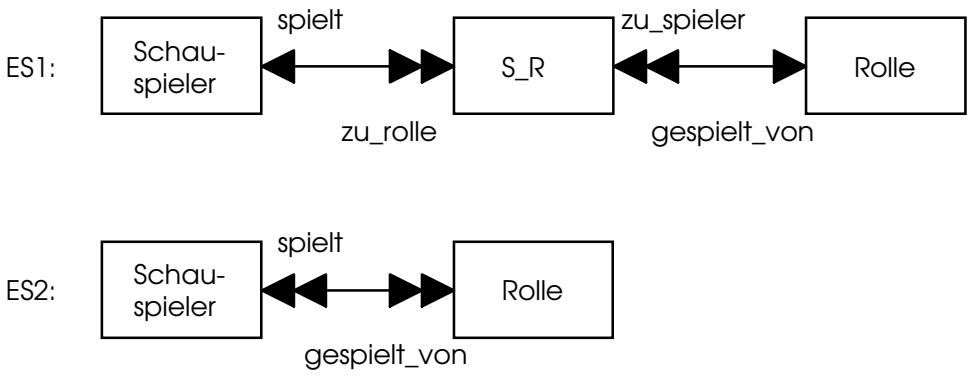

Abbildung 4.2: Beispielschemata in ODMG-93

Folgender Konflikt ergibt sich:
ES_1.S_R{zu_rolle, zu_spieler} $\leftrightarrow_{SOR}$ ES_2.schauspieler{spielt}/rolle{gespielt_von}.

Steht ein Objekttyp mit mehreren Beziehungen in Konflikt, so spricht man nach [Dup94] von einem Dekompositionskonflikt oder **struktureller Fragmentierung**. Mit dem ODMG-93 Objektmodell kann sich dieser Konflikt dadurch ergeben, daß Realweltbeziehungen zwischen mehreren Realweltkonzepten in einem Schema durch einen Objekttyp und in einem anderen Schema durch mehrere binäre Beziehungen abgebildet werden. Zur Darstellung dieses Konflikttyps mit einer entsprechenden Formel werden folgende Annahmen getroffen. In ES_1 existiere ein Objekttyp $X\{rx_1..rx_n\}$ mit Beziehungen $rx_1..rx_n$ zu Objekttypen $X_1..X_n$. In ES_2 existieren mit $X_1..X_n$ jeweils semantisch vergleichbare Objekttypen $Y_1..Y_n$, zwischen denen Beziehungen der Form $Y_i\{ry_{ij}\}/Y_j\{ry_{ji}\}$, $i \neq j$, existieren. Eine strukturelle Fragmentierung läßt sich gemäß folgender Formel darstellen:

(7) $X\{rx_1..rx_n\} \leftrightarrow_{SOR} Y_1\{ry_{12}\}/Y_2\{ry_{21}\} \mathbb{X} Y_2\{ry_{23}\}/Y_3\{ry_{32}\} \mathbb{X} .. \mathbb{X} Y_n\{ry_{n1}\}/Y_1\{ry_{1n}\}$.

Durch Anwendung der Operation eines Joins $\mathbb{X}$ über die Realweltbeziehungen kann die strukturelle Fragmentierung erkannt und dargestellt werden. Das folgende Beispiel soll dies verdeutlichen.

Beispiel 4.16: Abbildung 4.3 zeigt zwei Schemata ES_1 und ES_2 in graphischer Notation des ODMG-93 Objektmodells:

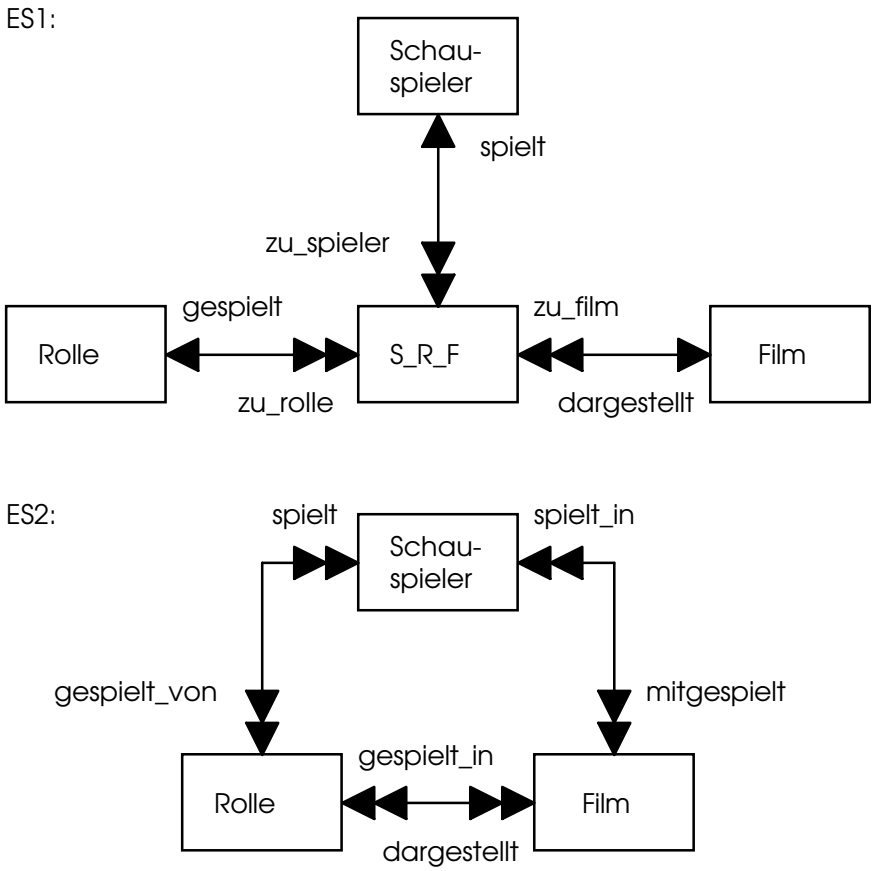

Abbildung 4.3: Beispielschemata in ODMG-93

Folgender Konflikt ergibt sich:

ES_1.S_R_F{zu_rolle, zu_spieler, zu_film}
$\leftrightarrow_{SOR}$ ES_2.rolle{gespielt_in}/film{dargestellt} $\mathbb{X}$
ES_2.film{mitgespielt}/schauspieler{spielt_in} $\mathbb{X}$
ES_2.schauspieler{spielt}/rolle{gespielt_von}.

4.3 Datenkonflikte

Datenkonflikte sind Konflikte zwischen Datenelementen, die Instanzen von semantisch vergleichbaren Schemaelementen verschiedener Schemata sind und dasselbe Realweltobjekt repräsentieren. Diese Konflikte auf der Datenebene können auf der Schemaebene nicht erkannt werden [Rah94]. Folgende zwei Arten von Datenelementen lassen sich abgrenzen:

1. elementare Datenelemente, im ODMG-93 Objektmodell repräsentiert durch Attributwerte als Instanzen von Attributen, sowie
2. übergeordnete Datenelemente, im ODMG-93 Objektmodell repräsentiert durch Objekte mit OID als Instanzen von Objekttypen und Referenzen auf Objekte als Instanzen von Beziehungen.

Danach können nach [LSPR93] die zwei folgenden Typen von Datenkonflikten klassifiziert werden:

- **Attributwertekonflikt,**
- **Objektidentifizierungskonflikt.**

Datenkonflikte werden während der Datenintegration gelöst.

Ein Attributwertekonflikt ergibt sich, wenn Attributwerte semantisch vergleichbarer Attribute von Objekten verschiedener Datenbanken, die dasselbe Realweltobjekt abbilden, nicht übereinstimmen [LSPR93]. Verschiedene Gründe lassen sich nach [KS91, Rah94] für das Auftreten von Ausprägungen dieses Konfliktes angeben, die hier in aller Kürze folgen:

a) Falsche Daten, d. h. inkorrekte Eingabedaten, überholte Daten durch unterschiedliche Änderungszeitpunkte,

b) Unterschiedliche Repräsentationen, d. h. keine festen Regelungen für Datenrepräsentation existieren.

Attributwertekonflikte sind die direkte Erweiterung der Attributkonflikte auf Datenbankebene. Das folgende Beispiel verdeutlicht mögliche Ausprägungen dieses Konfliktes.

Beispiel 4.17: Zwei Datenbanken DB_1 und DB_2 basierend auf den Schemata ES_1 und ES_2 aus Beispiel 4.7 enthalten die folgenden beiden Objekte, die auf demselben Realweltobjekt basieren:
DB_1.faust{"Faust - Eine Tragödie", "J.W. v. Goethe", "A-Verlag"},
DB_2.faust{"Faust - Eine Komödie", " von Goethe, Johann W."}.
Der Wert des Attributs *titel* in DB_2.faust beruht offenbar auf einer falschen Eingabe, die Werte der Attribute *autor* enthalten willkürliche Repräsentationen desselben Namens.

Objektidentifizierungskonflikte entstehen, wenn die Eigenschaftswerte keine zuverlässige Identifizierung von Objekten unterschiedlicher Datenbanken, die dasselbe Realweltobjekt abbilden, erlauben. Die zwei folgenden Möglichkeiten für einen Objektidentifizierungskonflikt können sich ergeben:

1. Objekte, die dasselbe Realweltobjekt abbilden, weisen unterschiedliche Eigenschaftswerte für gemeinsame Eigenschaften auf.
2. Objekte, die nicht dasselbe Realweltobjekt abbilden, weisen identische Eigenschaftswerte für alle gemeinsamen Eigenschaften auf.

Die erste Möglichkeit läßt sich auf eine Menge von Attributwertekonflikten zurückführen. Ein Grund für die zweite Möglichkeit besteht darin, daß nur sehr wenige gemeinsame Eigenschaften der zugehörigen Objekttypen existieren.

4.4 Zusammenfassende Klassifikation

Die in diesem Kapitel vorgenommene Beschreibung und Klassifizierung der bei der Integration heterogener, autonomer Datenbanksysteme in einem FDBS auftretenden Konflikte wird an dieser Stelle mit einigen Erkenntnissen und einer Zusammenfassung der Ergebnisse des Klassifizierungsprozesses abgeschlossen. Diese Ergebnisse bilden eine Grundlage für die weitere Untersuchung.

Als das zentrale Problem für die Integration erweist sich die semantische Heterogenität vor allem in Form der Schemakonflikte. Sie lassen sich in ihrer Vielfalt und Komplexität nur sehr schwer in eindeutige Kategorien einordnen. So werden auch in der Literatur zumeist nur Teilaspekte dieses Problems behandelt. Daraus und aus der Verwendung unterschiedlicher Datenmodelle ergeben sich oft unterschiedliche Ansatzpunkte für eine Einordnung der Konflikte. Bei der hier erfolgten Beschreibung und Klassifizierung stand deshalb auch nicht das Streben nach vollständiger Erfassung aller Konfliktmöglichkeiten im Vordergrund. Es wurde versucht, die Schemakonflikte zu finden und zu beschreiben, die bei Verwendung des ODMG-93 Objektmodells als GDM eines FDBSs auftreten können. Trotz dieser Beschränkung konnten bestimmte Konflikte, z. B. solche, die sich durch Fragmentierung ergeben, nicht genauer untersucht werden. Diese Spezialfälle können Anlaß für weitergehende Analysen bieten; für das in dieser Arbeit zu untersuchende Problem sollten die beschriebenen Konflikte als Ausgangsbasis ausreichen. Als Einstiegspunkte in die Systematisierung dienten die Vorgaben von [SP91, Dup94]. Es muß festgestellt werden, daß die sich ergebende Klassifikation keine vollständige Aufteilung der Konflikte in voneinander unabhängige Arten ergeben hat.

Beim ODMG-93 Objektmodell lassen sich sowohl Vor- als auch Nachteile hinsichtlich des Auftretens bestimmter Konfliktarten erkennen. Ein wesentlicher Vorteil ergibt sich dadurch, daß jedes Realweltobjekt als direkte Instanz genau eines Objekttyps modelliert wird. Die semantische Fragmentierung wird dadurch gegenüber anderen Datenmodellen, wie dem relationalen Modell oder objektorientierten Modellen nichtdisjunkter Vererbung, erheblich eingeschränkt. In solchen Modellen kann sich eine Realweltklasse als Schnittmenge der Realweltklassen mehrerer Relationen oder Objekttypen ergeben. Im ODMG-93 Objektmodell muß eine solche Schnittmenge zwangsläufig durch einen Subtyp mehrerer Supertypen abgebildet werden. Nachteilig gegenüber anderen Datenmodellen zeigen sich die Strukturkonflikte, die sich durch Konzepte wie mengenwertige Attribute oder die besondere Art der Assoziation durch Beziehungen ergeben. So werden im Relationenmodell keine mengenwertigen Attribute erlaubt, so daß kein Aggregationskonflikt auftreten kann, und Beziehungen werden über Attribute aufgebaut, so daß sich kein Konflikt Beziehung versus Attribut(e) ergeben kann.

Ausprägungen der beschriebenen Konfliktarten lassen sich oft nur durch zusätzliche Erkenntnisse über die zugrundeliegenden Weltausschnitte erkennen. Die folgende Abbildung zeigt eine zusammenfassende Klassifikation der Konflikte, wie sie in diesem Kapitel vorgenommen wurde.

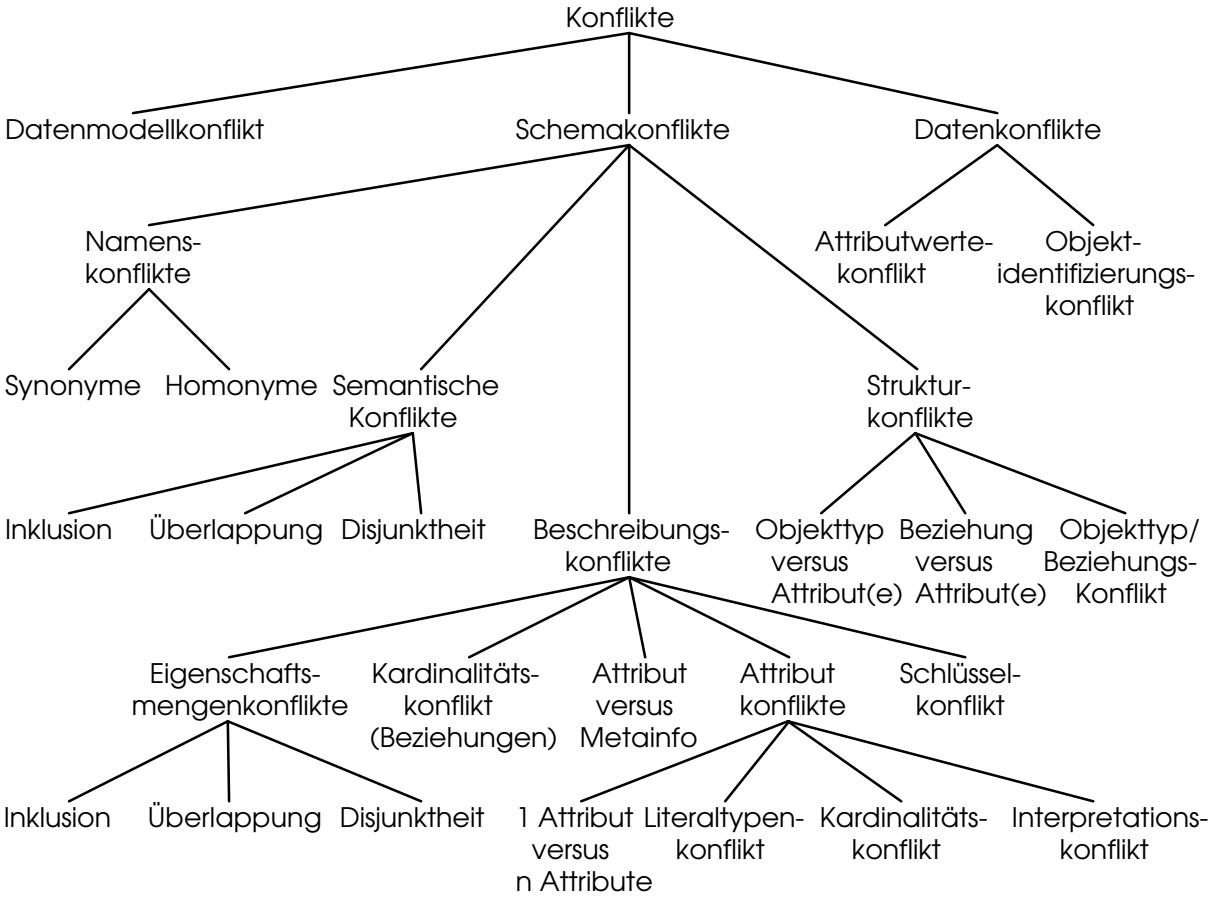

Abbildung 4.4: Klassifikation von Konflikten

5 Integrationsprozesse

In diesem Kapitel sind Anforderungen zu erarbeiten, anhand derer die Eignung des ODMG-93 Objektmodells als GDM zur Integration heterogener, autonomer Datenbanksysteme in einem FDBS untersucht werden kann. Grundlage der Integration ist der Aufbau der Schemaebenenhierarchie (siehe Abschnitt 3.3.1) sowie die Lösung der in Kapitel 4 klassifizierten Konflikte. Daraus werden vier Prozesse abgeleitet, die wesentlich für eine erfolgreiche Integration verantwortlich sind. Basis ist dabei die Realisierung eines eng gekoppelten Systems mit einem föderierten Schema, da hier eine Integration in engerem Sinne vollzogen wird. Diese Prozesse werden in den folgenden Abschnitten beschrieben, Lösungsansätze und daraus resultierende Anforderungen an das GDM werden dabei diskutiert. Abschließend werden alle Anforderungen zusammengefaßt.

5.1 Schematransformation

Die Transformation oder Übersetzung eines Schemas von einem Datenmodell in ein anderes Datenmodell wird innerhalb der Föderierung autonomer CDBSe nach dem Referenzmodell von [SL90] bei der Überführung der lokalen Schemata in die Komponentenschemata und der Überführung des föderierten Schemas zur Ableitung eines externen Schemas notwendig. Der Schritt der Transformation der lokalen Schemata in das GDM bildet die Voraussetzung für den Schemaintegrationsprozeß. Dabei wird der Datenmodellkonflikt als Ausprägung der syntaktischen Heterogenität gelöst. [SP91] diskutieren Möglichkeiten für eine Lösung im Rahmen der Schemaintegration. Dadurch würde sich allerdings die Anzahl der auftretenden Schemakonflikte erhöhen, da Stukturelemente verschiedener Datenmodelle in Konflikt geraten können. Der Prozeß der Schematransformation wird auch als *Homogenisierung* [HK95] bezeichnet.

Für die Schematransformation ist die Wahl eines geeigneten GDMs besonders wichtig, da hier das Fundament für eine erfolgreiche Schemaintegration gelegt wird. Zwei verschiedene Ansätze werden hierbei verfolgt, wobei die grundlegende Frage für die Schematransformation die nach der Größe der Ausdrucksfähigkeiten des GDMs ist. Nach [SCG91] müssen die Ausdrucksfähigkeiten (*expressiveness*) eines GDMs größer oder gleich denen eines jeden Datenmodells der CDBSe sein, um alle in den lokalen Schemata vorhandene Semantik in die Komponenten-schemata übertragen zu können. Ein zweiter Ansatz propagiert ein Datenmodell mit möglichst einfachen Ausdrucksfähigkeiten im Wortsinne eines kanonischen Datenmodells, damit möglichst wenige Schemakonflikte auftreten und die Schemakonfliktlösung wesentlich erleichtert wird. Wie der entstehende semantische Verlust bei der Schematransformation aufgefangen wird, ist Gegenstand derzeitiger Forschungsarbeit. Da sich diese Arbeit mit der Eignung des ODMG-93 Objektmodells als GDM beschäftigt und es sich dabei um ein Datenmodell mit zumindest vielfältigen Ausdrucksfähigkeiten handelt, wird der erste Ansatz weiterverfolgt.

Nach [SCG91] ergeben sich die Ausdrucksfähigkeiten eines Datenmodells daraus, wie direkt und in natürlicher Weise es ein Konzept der realen Welt darstellen kann, egal wie komplex das Konzept ist. Die Ausdrucksfähigkeiten teilen sich in einen strukturellen und in einen verhaltensmäßigen Teil. [SCG91] erarbeiteten eine Reihe von Charakteristiken, die ein GDM ihrer Meinung nach als Ausdrucksfähigkeiten besitzen sollte. Folgende Charakteristiken der strukturellen Ausdrucksfähigkeiten eines GDMs werden von [SCG91] aus den klassischen Abstraktionskonzepten hergeleitet:

- Charakteristik 1: Klassifikation/Instantiierung

- Bedingung: Trennung von Klassen und Individuen
- Empfehlung: Unterstützung von Meta- und Metametaklassen

- Charakteristik 2: Generalisierung/Spezialisierung
  - Bedingung: Unterstützung vieler Ebenen von Superklassen/Subklassen
  - Empfehlung: Unterstützung von multipler Vererbung und verschiedener Arten der Spezialisierung (exklusiv, nicht exklusiv)
- Charakteristik 3: Aggregation/Dekomposition und Assoziation
  - Bedingung: Aggregation und Assoziation zur Konstruktion komplexer Objekte gefordert
  - Empfehlung: Unterstützung anderer Formen (Listen, Bags u. a.)

Zum verhaltensmäßigen Teil wurde folgende Charakteristik angegeben:

- Charakteristik 4: Verhaltensmodellierung
  - Bedingung: Unterstützung der Definition von Operationen und Integritätsbedingungen
  - Empfehlung: Einkapselung von Operationen

Wie in Kapitel 4 gezeigt, erhöht sich die Anzahl und Komplexität der Schemakonfliktmöglichkeiten mit der Anzahl der unterstützten Konzepte. Deshalb sollte bei der Schematransformation wenn möglich darauf geachtet werden, Schemakonflikte bereits im Ansatz zu vermeiden. Als weiteres Problem bei der Schematransformation sowie in Hinblick auf die Schemaintegration sehen [SCG91] Datenmodelle als GDM, die mehr als ein Basisstrukturelement besitzen, wie z. B. das Entity-Relationship-Modell. Dabei werden bei der Schematransformation Entscheidungen gefordert, ob ein Schemaelement des lokalen Schemas z. B. als Entity oder als Relationship in das Komponentenschema übertragen wird. Ebenso ergibt sich eine Erhöhung der Anzahl von zu lösenden Strukturkonflikten, so daß folgende Charakteristik abgeleitet wird:

- Charakteristik 5
  - Bedingung: nur ein Basisstrukturelement.

Zusammenfassend läßt sich sagen, daß die Forderung nach hohen Ausdrucksfähigkeiten des GDMs sowohl Vor- als auch Nachteile mit sich bringt. Vorteile bieten sich durch die semantisch verlustfrei mögliche Übertragung der lokalen Schemata sowie durch die Möglichkeiten einer semantischen Anreicherung (siehe unten). Demgegenüber stehen die Schwierigkeiten, die sich in Hinblick auf die Schemaintegration ergeben. Aufgrund der zur Zeit noch fehlenden Praktikabilität eines anderen Ansatzes müssen die Forderungen von [SCG91] unterstützt werden. Die Autoren von [SCG91] haben in [CSG93] ein GDM vorgestellt, das über die aufgestellten Charakteristiken noch hinausgeht.

Verschiedene Ansätze für Methodiken zur Schemaübersetzung werden in der Literatur behandelt. Zwei unterschiedliche Methoden, explizite Transformation durch den DBA und Transformation nach Mappingregeln, werden von [SL90] geliefert. Dabei dürfte eine Kombination beider Methoden die besten Ergebnisse liefern, da jede für sich allein nicht zu realisieren ist. So ist der Aufwand für die reine Transformation von Hand zu groß, und vollautomatisches Transformieren führt zu semantischen Fehlern. [UW91] propagieren einen Ansatz auf der Basis eines semantischen Meta-Modells zur Beschreibung struktureller Differenzen zwischen Datenmodellen. Bei letzterem Ansatz werden jedoch Schematransformation und Schemaintegration wiederum vermischt, was wegen der

Komplexität vermieden werden soll. Eine eindeutig bevorzugbare Methodik gibt es nicht, deshalb sollte von Fall zu Fall entschieden werden.

Oftmals liegen die lokalen Schemata in Datenmodellen vor, deren Ausdrucksfähigkeiten begrenzt gegenüber denen des GDMs sind. Deshalb wird die Schemaübersetzung mit einer semantischen Anreicherung [HK95] gekoppelt, d. h. zusätzliche durch Wissensaneignung gewonnene Semantik sowie in den Schemata vorhandene inhärente Semantik soll explizit in den Komponentenschemata ausgedrückt werden. Dabei ist nach [SL90] zu beachten, daß das Komponentenschema dieselbe Datenbasis repräsentiert wie das lokale Schema und daß Befehle an das Komponentenschema in Befehle an das korrespondierende lokale Schema übersetzt werden können, so daß man von einer *kontexterhaltenden Transformation* spricht.

5.2 Schemaintegration

Die Integration der Exportschemata zu einem oder mehreren föderierten Schemata ist der wichtigste Prozeß bei der Realisierung eines FDBSs. Diese Arbeit beschränkt sich auf die Schaffung eines einheitlichen globalen Schemas, weil dadurch eine gesamtheitliche Betrachtungsweise für alle exportierten Daten geschaffen und die semantische Heterogenität überwunden wird. Einen grundlegenden Überblick über Herangehensweisen zur Schemaintegration bieten [BLN86]. In diesem Abschnitt werden die bei einer Schemaintegration notwendigen Schritte, Qualitätskriterien für das föderierte Schema, Konzepte und Techniken zur Auflösung von Schemakonflikten und die dafür notwendige Unterstützung durch das GDM sowie existierende Integrationsansätze diskutiert.

5.2.1 Schritte der Schemaintegration

Der erste Schritt bei der Schemaintegration ist die Wahl einer Integrationsstrategie. Im allgemeinen müssen im Rahmen einer Schemaintegration zur Realisierung eines FDBSs mehr als zwei Exportschemata integriert werden. Dann ist es notwendig, eine Strategie zu wählen, in welcher Reihenfolge die Exportschemata integriert werden sollen. Nach [BLN86] kann man vier Strategien unterscheiden, die in zwei Klassen, binäre und n-stellige Strategien, eingeteilt werden können. Abbildung 5.1 zeigt die vier unterschiedlichen Strategien.

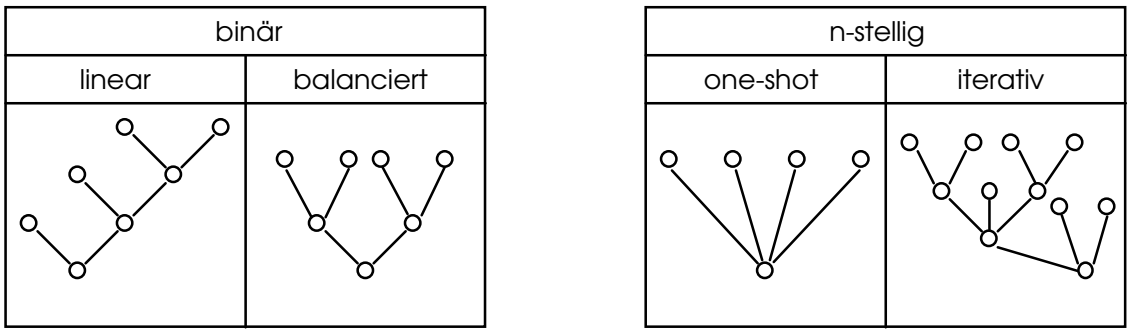

Abbildung 5.1: Schemaintegrationsstrategien nach [BLN86]

Bei den binären Strategien werden immer nur zwei Schemata integriert. Dabei entstehende Zwischenschemata werden mit weiteren Schemata integriert. Ähnlich ist es bei der iterativen Strategie, nur daß auch mehr als zwei Schemata gleichzeitig integriert werden können. Bei der one-shot Strategie werden alle Exportschemata gleichzeitig integriert. Die binären Strategien sind zu bevorzugen, da die Komplexität auf die Integration von immer nur zwei Schemata begrenzt wird. Bei sehr vielen zu integrierenden Schemata kann jedoch der Aufwand dann sehr groß werden, da sehr viele Integrationsschritte notwendig werden. Die folgenden Schritte der

Schemaintegration müssen bei den Strategien linear, balanciert und iterativ für jeden Integrationsschritt wiederholt werden.

Der zweite Schritt bei der Schemaintegration ist die Analyse und der Vergleich der zu integrierenden Exportschemata. Dabei müssen alle Schemaelemente erfaßt werden, die semantisch vergleichbare Realweltklassen abbilden. Alle auftretenden Schemakonflikte müssen fixiert werden. Zur Unterstützung dieses Schrittes wurden die in Abschnitt 4.1.2.1 erwähnten *correspondence assertions* vorgeschlagen, die datenmodellunabhängig eine Fixierung von Schemakonflikten auf logischer Basis erlauben. Dieser Schritt muß vom DBA des FDBSs manuell auf der Basis von gesicherten Erkenntnissen über die Semantik der zu integrierenden Exportschemata durchgeführt werden.

Der dritte Schritt ist die Erstellung des integrierten Schemas, das Kernstück der Schemaintegration und die entscheidende Phase bei der Realisierung eines FDBSs. Für diesen Schritt wurden in der Literatur viele und sehr verschiedene Ansätze und Algorithmen vorgeschlagen. [BLN86] haben Qualitätskriterien für das föderierte Schema erarbeitet, an denen jeder Ansatz gemessen werden kann. Diese Qualitätskriterien sind im einzelnen:

- **Vollständigkeit und Korrektheit**

  Alle Realweltkonzepte, die von den Exportschemata abgebildet wurden, müssen auch vom föderierten Schema abgebildet werden.

- **Minimalität**

  Wenn ein Realweltkonzept von mehr als einem Exportschema abgebildet wird, darf es nur einmal im föderierten Schema abgebildet werden.

- **Verständlichkeit**

  Das föderierte Schema soll leicht verständlich für den FDBS-Nutzer sein, das bedeutet, wenn verschiedene Repräsentationen für eine Realweltklasse zur Auswahl stehen, sollte die verständlichste gewählt werden.

Die Qualitätskriterien stellen einen hohen Anspruch an das föderierte Schema. Die Forschung nach Integrationsansätzen, die diesem Anspruch genügen, ist derzeit in vollem Gange.

5.2.2 Auflösung von Schemakonflikten

Die Hauptaufgabe bei der Erstellung des föderierten Schemas ist die Auflösung von Schemakonflikten. Die zwei folgenden unterschiedlichen Aspekte haben sich dabei ergeben:

- Nutzung allgemeiner datenmodellunabhängiger Konfliktlösungstechniken,
- Nutzung datenmodellspezifischer Konzepte.

Nachfolgend werden beide Aspekte genauer beschrieben. Anforderungen an die für beide Aspekte notwendige technische Unterstützung durch das GDM folgen im nächsten Abschnitt.

Bei der Forschung nach Ansätzen zur Schemaintegration wurden Techniken zur Auflösung von Schemakonflikten entwickelt, die allgemein für jede Föderierung angewendet werden können, da sie nicht auf Konzepten bestimmter Datenmodelle beruhen. Sie müssen nur jeweils an die technischen Möglichkeiten des verwendeten GDMs angepaßt werden. Beispiele für solche Techniken sind das Umbenennen bei Namenskonflikten, Transformationsfunktionen [Rah94] zur Auflösung von Attributkonflikten sowie das Objektifizieren von Attributen und Beziehungen zur Auflösung von Strukturkonflikten.

Die Charakteristiken von [SCG91] hinsichtlich der strukturellen und verhaltensmäßigen Ausdrucksfähigkeiten eines GDMs gelten auch hinsichtlich der Schemaintegration, um ein föderiertes Schema zu erschaffen, das die gesamte Semantik der zugrundeliegenden Exportschemata sowie zusätzlich gewonnene Semantik abbildet. Die Ausdrucksfähigkeiten des GDMs werden dazu genutzt, um semantisch vergleichbare Schemaelemente so zu integrieren, daß bestehende Schemakonflikte aufgelöst und die Qualitätskriterien erfüllt werden. Zwei Konzepte, die in der Literatur zur Auflösung von semantischen und Eigenschaftsmengenkonflikten zwischen Objekttypen gefordert werden, sollen an dieser Stelle besondere Erwähnung finden.

Das am häufigsten geforderte Konzept zur Auflösung dieser Konflikte ist die Generalisierung/Spezialisierung [Bra93, Dup94, McL93]. Durch den Einsatz von verschiedenen Vererbungstechniken, wie z. B. nichtdisjunkte totale Vererbung, werden semantisch vergleichbare Schemaelemente der Exportschemata so integriert, daß ein Abbild der realen Welt entsteht, das der vereinigten Semantik der zu integrierenden Schemaelemente entspricht. Der Einsatz dieser Techniken bei der Schemaintegration ist weitgehend erforscht und hat sich bewährt, so daß für ein GDM eine breite Unterstützung in diesem Bereich gefordert werden kann.

Ein weiteres Konzept, daß zur Lösung dieser Konflikte vorgeschlagen wurde [Dup94], ist das Rollenkonzept [Bee93], auch als mehrfache dynamische Klassenzugehörigkeit von Objekten [Sch94] oder Multi-Instantiation [Dup94] bezeichnet. Dieses Konzept bedeutet, daß Objekte mit unveränderter Identität gleichzeitig oder aufeinanderfolgend Instanzen verschiedener Objekttypen sein können. Für die Schemaintegration bedeutet das, daß semantisch vergleichbare Objekttypen unverändert in das föderierte Schema übernommen werden und bei der Datenintegration als identisch erkannte Objekte mit derselben globalen Identität versehen werden. Dies widerspricht den Forderungen nach Minimalität und Verständlichkeit, so daß eine Forderung nach diesem Konzept für das GDM nicht sinnvoll ist.

5.2.3 Technische Realisierung

Ein wichtiger Gesichtspunkt ist die technische Unterstützung des GDMs für die Schemaintegration. Diese drückt sich einerseits durch die Unterstützung von multiplen Schemata und andererseits durch bereitgestellte Operationen zum Restrukturieren und Mischen von Schemaelementen der Exportschemata in das föderierte Schema aus. Nach [SCG91] ist dies Teil des *semantischen Relativismus* eines Datenmodells. Darunter wird die Kraft der Operationen eines Datenmodells zur Erstellung externer Schemata verstanden. Dies gilt sowohl für die Schemaintegration als auch die Ableitung von Sichten.

Zwei unterschiedliche Ansätze zur technischen Realisierung der Schemaintegration haben sich ergeben. Zum einen werden spezielle Operationen gefordert [CSG93], die die Techniken zur Konfliktauflösung unterstützen. Zum anderen werden Ansätze propagiert, die auf den in vielen Datenmodellen vorhandenen Ableitungsmechanismen auf der Basis extensionaler Anfragen beruhen.

Für Operationen zur Schemaintegration leiteten [SCG91] folgende Charakteristik ab, die ein GDM erfüllen sollte:

- Charakteristik 6: Operationen zur Schemaintegration
  - Bedingung: Unterstützung von Integrationsoperationen mit Funktionalität zur Auflösung von Schemakonflikten
  - Empfehlung: Unterstützung von *upward inheritance*

Unter upward inheritance wird die Schaffung von Supertypen verstanden, die von den zu integrierenden Typen Struktur und Verhalten "erben". Die allgemeine Forderung nach Integrationsoperationen muß weiter hinsichtlich der benötigten Funktionalität spezifiziert werden, weshalb im folgenden einige Operationen genannt werden sollen:

- Operationen zum Mischen von Objekttypen der Exportschemata in einen Objekttyp des föderierten Schemas,
- Operationen zur Unterstützung der geforderten Generalisierung/Spezialisierung wie das Hinzufügen von Subtypen und die von [SCG91] als Empfehlung gegebene upward inheritance,
- Operation zur Umwandlung von Metadaten in Attribute und zurück zur Lösung des Konfliktes Attribut versus Metainfo,
- Operationen zur Unterstützung allgemeiner Konfliktlösungstechniken, insbesondere:
  - Operationen zum Umbenennen von Schemaelementen,
  - Transformationsfunktionen [Rah94],
  - Operationen zum Objektifizieren von Attributen und Beziehungen.

Mit der Unterstützung dieser und eventuell weiterer Integrationsoperationen durch das GDM sollte die technische Realisierung der Schemaintegration möglich sein, da über die bereitgestellte Funktionalität Schemakonflikte aufgelöst werden können.

Für die Schemaintegration auf der Basis von Ableitungsmechanismen wurde von [SCG91] folgende Charakteristik für ein GDM gefordert:

- Charakteristik 7: Ableitungsoperationen
  - Bedingung: Unterstützung von Operationen zum Ableiten externer Schemata mit mindestens derselben Funktionalität wie Ableitungsmechanismen auf Basis der relationalen Algebra

Insbesondere wird gefordert, daß jedes Konzept, das vom GDM unterstützt wird, auch durch Ableitungsoperationen erzeugt werden kann. Mit den Ableitungsoperationen allein ist jedoch keine vollständige technische Realisierung der Schemaintegration möglich, da bestimmte Konfliktauflösungen wie z. B. die von Attributkonflikten damit nicht möglich sind. Deshalb erscheint eine Kombination von Ableitungsmechanismen und Integrationsoperationen als sinnvoller Ansatz zur technischen Realisierung der Schemaintegration.

5.2.4 Automatisierbarkeit und Integrationsansätze

Eine interessante Fragestellung, die sich bei der Forschung nach Integrationsansätzen ergeben hat, ist die nach der Automatisierbarkeit der Schemaintegration. Eine vollständige automatische Durchführung des gesamten Prozesses der Schemaintegration ist nicht möglich, da kein System die komplette Semantik eines Datenbankschemas erfassen kann [SL90]. Die Unterstützung durch Menschen mit Kenntnis der Semantik der beteiligten Schemata ist also notwendig. Eine vollständige manuelle Abarbeitung der Schemaintegration erscheint aber aufgrund des zu erwartenden Aufwandes nicht sinnvoll. Ein empfehlenswerter Kompromiß ist deshalb die Unterstützung der Schemaintegration durch geeignete Softwaretools.

Eine Betrachtung und Diskussion bisheriger Ansätze zur Schemaintegration soll diesen Abschnitt abrunden. Erste Ansätze wurden Ende der 70er und Anfang der 80er Jahre veröffentlicht. Sie basieren zumeist auf dem relationalen Modell oder dem Entity-Relationship-Modell (ER-Modell) als GDM. Die Vielzahl und Komplexität der Schemakonflikte wurde noch nicht voll erkannt, so daß jeweils nur Konfliktlösungen für einen Teilbereich der

Schemakonflikte propagiert wurden. Ein Überblick und Vergleich dieser Ansätze findet sich in [BLN86]. Spätere Ansätze [SP92, SPD92] auf der Basis eines erweiterten ER-Modells sowie eines generischen Datenmodells berücksichtigten zwar die gesamten Schemakonflikte, boten jedoch keine ausreichenden Konfliktlösungen zur Erfüllung der Forderungen von [BLN86]. Neuere Ansätze propagieren eine Schemaintegration auf Basis von objektorientierten Datenmodellen. Basierend auf einer objektorientierten Anfragesprache (OSQL) präsentieren [CL93] einen Ansatz, der die deklarative Definition von Mappings zwischen verschiedenen Schemata, die auf einem objektorientierten GDM beruhen, erlaubt. Damit wird zwar Interoperabilität zwischen unterschiedlichen Datenbanksystemen ermöglicht, ein föderiertes Schema im geforderten Sinn wird damit jedoch nicht erstellt. Ein Ansatz von [SST93] ermöglicht die Schemaintegration durch Ableitungsmechanismen auf der Grundlage einer objektorientierten Algebra. Zwei Ansätze zur Schemaintegration mit dem ODMG-93 Objektmodell als GDM [BFN94, Rad94] werden im nächsten Kapitel diskutiert.

5.3 Ableitung externer Schemata

Externe Schemata bieten die Möglichkeit, den Nutzern eines FDBSs unterschiedliche Sichten auf die von den CDBSen bereitgestellten Daten zu ermöglichen. Jedes externe Schema muß aus dem föderierten Schema abgeleitet werden. Dabei kann man prinzipiell die folgenden zwei verschiedene Arten von externen Schemata unterscheiden:

- externe Schemata ausgedrückt im GDM,
- externe Schemata ausgedrückt in einem anderen Datenmodell.

Letzteres geht über die im ANSI/X3/SPARC 3-Schichten-Modell vorgesehenen externen Schemata hinaus. Im Rahmen einer Föderation kann es jedoch notwendig sein, den Nutzern den Zugriff auf die Daten über unterschiedliche Anfrageschnittstellen zu ermöglichen. Als Erweiterung kann die Forderung bestehen, daß ein externes Schema die Informationen eines Exportschemas ausgedrückt im Datenmodell des zugrundeliegenden lokalen Schemas bereitstellt. Nachfolgend werden beide Arten genauer betrachtet.

Für die Ableitung von externen Schemata ausgedrückt im GDM gilt die in Charakteristik 7 von [SCG91] aufgestellte Forderung. Aus dem föderierten Schema sollte bei Unterstützung dieser Forderung durch das GDM jede beliebige Untermenge als externes Schema abgeleitet werden können, womit jedem Nutzer oder jeder Nutzergruppe ein beliebige Sicht gewährt wird. Die zwei folgenden zusätzlichen Anforderungen an das GDM lassen sich nach [SL90] für die Ableitung externer Schemata stellen:

- Unterstützung der Addition zusätzlicher Integritätsbedingungen und
- Unterstützung von Maßnahmen zur Zugriffskontrolle.

Damit soll die Möglichkeit eröffnet werden, externe Schemata für bestimmte Nutzer oder Nutzergruppen zuzuschneiden.

Für die Ableitung von externen Schemata in andere Datenmodelle als das GDM gibt es zwei Möglichkeiten. Bei der ersten Möglichkeit wird zuerst ein externes Schema ausgedrückt im GDM abgeleitet und dann durch Schematransformation in ein anderes Datenmodell überführt. Bei der zweiten Möglichkeit wird das gesamte föderierte Schema in ein anderes Datenmodell transformiert. Externe Schemata werden dann auf der Basis der in diesem Datenmodell vorhandenen Ableitungsmechanismen erstellt.

5.4 Datenintegration

Die Datenintegration ist ein Prozeß, bei dem die von den CDBSen verwalteten Daten dem Nutzer des FDBSs an der globalen Schnittstelle auf der Grundlage des föderierten Schemas oder der externen Schemata zur Verfügung gestellt werden. Die Daten werden dabei entlang der Schemaebenenhierarchie in das Format des GDMs oder des Datenmodells eines externen Schemas transformiert, d. h. der Aufbau der Schemaebenenhierarchie ist Voraussetzung für den Beginn der Datenintegration. Grundsätzlich lassen sich die folgenden zwei Etappen der Datenintegration erkennen:

1. Vorintegration, d. h. die Integration aller Datenbestände der CDBSe bei Realisierung des FDBSs,
2. laufende Integration, d. h. die Integration der Daten bei laufendem Betrieb des FDBSs.

Die Vorintegration ist notwendig, wenn die Daten der CDBSe mit Beginn des Betriebs des FDBSs global durch Identitätsinformationen repräsentiert sein sollen. Da die Daten nur in den Datenbanken und im Format des Datenmodells der jeweiligen CDBSe gespeichert sind und bleiben, ist der Prozeß der laufenden Datenintegration mit jeder globalen Transaktion verbunden, die auf die Daten zugreift. Zwei Aspekte der Datenintegration, die der Unterstützung durch Konzepte und Mechanismen des GDMs bedürfen, nämlich die Identifizierung realweltäquivalenter Objekte und die Behandlung der Objektidentität, werden in diesem Abschnitt diskutiert.

5.4.1 Identifizierung realweltäquivalenter Objekte

Eine besondere Aufgabe der Datenintegration ist die Identifizierung von Objekten verschiedener Datenbanken, die dasselbe Realweltobjekt abbilden. Diese Objekte werden als realweltäquivalente Objekte oder als Proxy-Objekte desselben Realweltobjektes bezeichnet [SS95, SST93], d. h. sie sind Instanzen semantisch vergleichbarer Objekttypen der zugehörigen Exportschemata. Ziel ist es, diese Proxy-Objekte durch eine direkte Instanz eines Objekttyps des föderierten Schemas zu repräsentieren. Eine Basis für die Identifizierung von realweltäquivalenten Objekten sind die Eigenschaftswerte der Objekte, da durch sie Aussagen über das zugrundeliegende Realweltobjekt getroffen werden können. Im folgenden werden Methoden zur Proxy-Objektidentifizierung und die notwendige Unterstützung durch das GDM behandelt.

Die folgenden zwei Methoden zur Identifizierung realweltäquivalenter Objekte wurden in der Literatur bisher besonders favorisiert:

1. Schlüsseläquivalenz [GB91], d. h., existiert eine gemeinsame Schlüsselbedingung, so liegen bei übereinstimmenden Werten der Schlüsseleigenschaften Proxy-Objekte vor,
2. Attributäquivalenz [CS91], d. h., stimmen die Werte aller gemeinsamen Attribute überein, so liegen Proxy-Objekte vor.

Nach [LSPR93] sind beide Methoden nur eingeschränkt nutzbar. Da in vielen Fällen keine gemeinsamen Schlüsselbedingungen existieren, ist die Schlüsseläquivalenz nicht für alle semantisch vergleichbaren Objekttypen möglich. Oftmals sind auch nur wenige gemeinsame Eigenschaften der Objekttypen vorhanden, so daß bei Attributäquivalenz sich die Anzahl von Objektidentifizierungskonflikten häufen würde. Daraus ergibt sich, daß in einer heterogenen Umgebung die Eigenschaftswerte zur sicheren Identifizierung nicht ausreichen [Bee93].

Ein praktikabler Ansatz zur Proxy-Objektidentifizierung, der dies berücksichtigt, wird von [LSPR93] beschrieben. Dieser Ansatz beruht auf den folgenden zwei Punkten:

1. erweiterte Schlüsseläquivalenz auf der Basis einer erweiterten Schlüsselbedingung und
2. funktionale Abhängigkeiten auf Instanzenebene (ILFDs - *instance level functional dependencies*).

Die erweiterte Schlüsselbedingung ergibt sich aus der Vereinigung der Schlüsselbedingungen der Objekttypen. ILFDs sind Funktionen, die Abhängigkeiten zwischen Eigenschaftswerten von nichtgemeinsamen Eigenschaften der Objekttypen herstellen. Diese durch Aneignung von zusätzlichem Wissen über die von den Exportschemata abgebildeten Diskursbereiche abgeleiteten Funktionen müssen als Metainformationen im Data Dictionary des FDBMSs gespeichert werden können.

5.4.2 Behandlung der Objektidentität

Besondere Bedeutung bei der Föderation heterogener, autonomer Datenbanksysteme kommt der Behandlung der Objektidentität zu. Die Verwaltung, Speicherung und der Zugriff auf die in einem FDBS bereitgestellten Daten obliegt den CDBSen. Diese Daten werden als lokale Objekte bezeichnet. Dem Nutzer eines FDBSs werden globale, virtuelle Objekte, die auf lokale Objekte zurückgeführt werden können, zur Verfügung gestellt. Objektidentität, die Eigenschaft eines Objektes, die es von allen anderen Objekten unabhängig von seinem Zustand unterscheidet [KC86], wird von den meisten heutigen DBMSen in unterschiedlicher Art und Weise unterstützt. Es ist in diesem Abschnitt zu untersuchen, welche Anforderungen diesbezüglich an das FDBMS und das GDM gestellt werden müssen.

Nach [EK91] werden die folgenden drei Arten der Unterstützung von Objektidentität abhängig von ihrer räumlichen und zeitlichen Begrenzung unterschieden:

- wertbasierte Identität, d. h., die Identität eines Objektes ist durch die Werte der zu einer Primärschlüsselbedingung gehörenden Eigenschaften (Attribute und Beziehungen) bestimmt, sie wird räumlich durch den Gültigkeitsbereich der Primärschlüsselbedingung und zeitlich durch den Bestand des Schlüsselwertes begrenzt; ein Beispiel für wertbasierte Identität sind die Primärschlüsselbedingungen des relationalen Modells;
- *session object identifiers*, d. h., die Identität eines Objektes ist räumlich durch die Datenbank und zeitlich durch die Dauer einer *Session*, einer Gruppe von Operationen, begrenzt; ein Beispiel hierfür sind *handle*-Mechanismen in objektorientierten Datenbanksystemen wie O_2;
- unveränderliche Objektidentität, d. h., die Identität eines Objektes ist unabhängig von seinem Zustand räumlich durch die Datenbank und zeitlich durch die Lebenszeit des Objektes begrenzt.

Die unveränderliche Objektidentität wird zumeist durch sogenannte *object identifiers* (OIDs) mit Hilfe des Surrogatmechanismus realisiert. Damit wird eine eindeutige Identifizierung eines Objektes innerhalb einer Datenbank ermöglicht. Die unveränderliche Objektidentität wird deshalb nach [EK91] auch als sichere Objektidentität bezeichnet. Mit einer sicheren Objektidentität kann der Prozeß der Datenintegration erleichtert werden, da der Aufwand für die Proxy-Objektidentifizierung gesenkt wird, denn sind realweltäquivalente Objekte einmal identifiziert, so kann dies durch die unveränderliche Objektidentität dauerhaft gesichert werden. Wegen dieser Vorteile kann die Forderung nach einer unveränderlichen Objektidentität für alle globalen Objekte eines FDBSs erhoben werden. Nachfolgend wird untersucht, ob und wie diese Forderung zu realisieren ist, und welche Unterstützung durch das GDM notwendig ist.

Zur Realisierung unveränderlicher Objektidentität in einem FDBS sind die folgenden zwei Punkte notwendig:

- Alle globalen Objekte erhalten eine eigene globale Objektidentität, repräsentiert durch eine globale OID (GOID).
- Eine die gesamte Lebenszeit des globalen Objektes dauernde Abbildung zwischen der GOID und der lokalen Objektidentität der zugrundeliegenden Objekte, repräsentiert durch Funktionen zur lokalen Objektidentifizierung [EK91], wird realisiert.

Nach [EK91] erfordert dies eine globale Repräsentation aller Objekte, d. h. das FDBMS muß in einer eigenen Datenbank die Identitätsinformationen für alle Objekte speichern. Ein von [SS95] propagierter Ansatz erlaubt es allerdings, einige dieser Informationen in den CDBSen zu speichern. Da die lokale Objektidentität wertbasiert oder durch *session* OIDs realisiert sein kann, ist es notwendig, Maßnahmen zur Sicherung der dauerhaften Abbildung zu ergreifen. [EK91] machen dafür folgende Vorschläge, die in die Ausführungsautonomie der CDBSe eingreifen:

- Verbot der Änderung von Schlüsselattributen,
- Änderungs-Log für Schlüsselattribute mit Lesezugriff des FDBMSs zum Ändern der lokalen Objektidentifizierungsfunktion,
- Überwachung des Transaktions-Log durch das FDBMS.

Weitere Vorschläge finden sich in [SS95]. Eine generelle Lösung dieses Konfliktes zwischen Autonomieforderung und unveränderlicher Objektidentität gibt es zur Zeit nicht [SS95]. Eine Alternative ohne Autonomieverletzung, bei der jedoch die Forderung nach Sicherung unveränderlicher Objektidentität für alle globalen Objekte deutlich abgeschwächt wird, bieten [HD93]. Dabei werden sogenannte Integrationsalternativen angeboten, d. h. für jeden Objekttyp des föderierten Schemas wird die Art der Objektidentität festgelegt.

Bei der Handhabung der Objektidentität in einem FDBS sind besondere Mechanismen zur Behandlung von realweltäquivalenten Objekten notwendig. Wurden solche Objekte durch eine der im letzten Abschnitt beschriebenen Methoden identifiziert, so werden sie gemäß der unveränderlichen Objektidentität dauerhaft mit einem globalen Objekt verbunden. Durch lokale Änderungen können sich die Bedingungen, die zu einer Proxy-Objektidentifizierung geführt haben, ändern. Es wird dazu vorgeschlagen, eine Abbildung, auch als *same*-Realation bezeichnet [SS95], zwischen den Identifizierungsfunktionen von Proxy-Objekten desselben Realweltobjekts global zu implementieren und mit den beschriebenen Mechanismen von [EK91, SS95] zu überwachen. Durch das lokale oder globale Einfügen von Objekten können neue realweltäquivalente Objekte identifiziert werden. Wenn ein Proxy-Objekt lokal gelöscht wird, so muß auch der Eintrag in der *same*-Relation gelöscht werden. Diese Änderungen in der *same*-Relation ziehen Konsequenzen für zugehörige globale Objekte nach sich. Diese Konsequenzen können Vereinigungen von globalen Objekten oder die Veränderung des Objekttyps eines globalen Objekts sein. Eine entsprechende Unterstützung durch das GDM ist dafür notwendig.

5.5 Anforderungen an ein gemeinsames Datenmodell

Zum Abschluß dieses Kapitels sollen alle Anforderungen, die bei der Erörterung der Integrationsprozesse an ein gemeinsames Datenmodell erarbeitet wurden, zusammengefaßt werden. Die Anforderungen werden in der nachfolgenden Tabelle präsentiert.

Integrationsprozeß	Anforderungen an das GDM
Schematransformation	1. Unterstützung der Charakteristiken von [SCG91]
Schemaintegration	2. Erstellung eines föderierten Schemas gemäß der Kriterien von [BLN86] soll möglich sein, dazu Anpassung von Konfliktlösungstechniken und Unterstützung bestimmter Konzepte 3. Unterstützung von Integrationsoperationen und Ableitungsmechanismen
Ableitung externer Schemata	4. Unterstützung eines Sichtenkonzeptes mit Zugriffskontrolle und zusätzlichen Integritätsbedingungen 5. Unterstützung der Ableitung in andere Datenmodelle
Datenintegration	6. Unterstützung eines Mechanismus zur Identifizierung realweltäquivalenter Objekte 7. Unterstützung eines Mechanismus zur Gewährleistung unveränderlicher Objektidentität

Tabelle 5.1: Anforderungen an ein gemeinsames Datenmodell

6 Eignung des ODMG-93 Objektmodells als GDM zur Integration

Die Integration heterogener, autonomer Datenbanksysteme in einem FDBS mit dem ODMG-93 Objektmodell als GDM soll in diesem Kapitel untersucht werden, um die Eignung des ODMG-93 Objektmodells zu bewerten. Ausgangspunkt für die Untersuchung sind die in Kapitel 5 beschriebenen Integrationsprozesse, die daraus abgeleiteten Anforderungen an ein GDM sowie die Kapitel 4 beschriebenen Konflikte bei der Integration. Für jeden Prozeß wird die Erfüllung der Anforderungen ermittelt, Techniken zur Konfliktlösung auf ihre Durchführbarkeit überprüft, sowie Realisierungsvorschläge unterbreitet oder auf ihre Umsetzbarkeit hin untersucht. Abschließend wird die Eignung des ODMG-93 Objektmodells bewertet.

6.1 Schematransformation

Bei Verwendung des ODMG-93 Objektmodells als GDM eines FDBSs werden bei der Schematransformation die lokalen Schemata der CDBSe ausgedrückt in den unterschiedlichsten Datenmodellen in Komponentenschemata ausgedrückt im ODMG-93 Objektmodell übertragen.. Dabei wird gefordert, daß die Transformation semantisch verlustfrei erfolgt und wenn nötig eine semantische Anreicherung vorgenommen werden kann. Deshalb ist nachfolgend zu untersuchen, ob das ODMG-93 Objektmodell die geforderten Ausdrucksfähigkeiten unterstützt, welche Ansätze es zur Transformation von Schemata aus gängigen Datenmodellen gibt und in welcher Form eine semantische Anreicherung möglich ist. Technische Aspekte wie das Erstellen von Transformationen zwischen den lokalen Schemata und den entsprechenden Komponentenschemata bleiben hier unberücksichtigt.

6.1.1 Die Ausdrucksfähigkeiten des ODMG-93 Objektmodells

Zuerst wird untersucht, wie das ODMG-93 Objektmodell die von [SCG91] geforderten Charakteristiken bezüglich der Ausdrucksfähigkeiten unterstützt. Das Ergebnis zeigt die folgende Tabelle:

Charakteristik	Bed./Empf.	Unterstützung durch das ODMG-93 Objektmodell
C1 - Klassifikation Instantiierung	Bedingung	Objekttypen, Klassen und Objekte
	Empfehlung	Typ *type* als Metatyp für Objekttypen
C2 - Generalisierung/ Spezialisierung	Bedingung	unbegrenzte Anzahl von Ebenen von Sub-/Supertypen durch Vererbung
	Empfehlung	multiple Vererbung, totale und nichttotale Vererbung, disjunkte Vererbung
C3 - Aggregation Assoziation	Bedingung	Assoziation durch Beziehungen, Aggregation durch *struct* und mengenwertige Attribute
	Empfehlung	mengenwertige Attribute der Formen List, Bag und Array
C4 - Verhaltens-modellierung	Bedingung	Definition von Operationen, Schlüsselbedingungen, prozedurale Integritätsbedingungen
	Empfehlung	Trennung von Interface und Implementierung
C5 - Basisstruktur-elemente	Bedingung	Objekttypen sind einziges Basisstrukturelement

Tabelle 6.1: Unterstützung der Charakteristiken von [SCG91] durch das ODMG-93 Objektmodell

Das ODMG-93 Objektmodell unterstützt offensichtlich die Charakteristiken weitgehend. Wichtig für die Schematransformation sind dabei besonders die Konzepte der Generalisierung/ Spezialisierung und der Aggregation und Assoziation, da diese Konzepte in vielen Datenmodellen Verwendung finden. Zusammen mit der Unterstützung der Bedingung von Charakteristik 4, d. h. die Möglichkeit der Definition und Erweiterung von funktionalem Verhalten, ist damit ein Ansatz für mögliche semantische Anreicherungen gegeben. Die Frage, ob die Ausdrucksfähigkeiten des ODMG-93 Objektmodells damit ausreichen, um Schemata aus den verbreitetsten Datenmodellen, z. B. dem relationalen Modell, dem ER-Modell oder einem anderen objektorientierten Modell, semantisch verlustfrei zu übertragen, bedarf genauerer Analysen. Da eine vollständige Untersuchung dazu aus Zeitgründen nicht möglich ist, sollen im weiteren nur einige Ansätze erwähnt sowie Probleme aufgezeigt werden, die bei der Transformation in das ODMG-93 Objektmodell auftreten können.

6.1.2 Die Transformation vom Relationenmodell in das ODMG-93 Objektmodell

Der Übertragung relationaler Schemata sollte aufgrund der enormen Verbreitung von auf dem relationalen Modell basierenden Systemen besondere Aufmerksamkeit gewidmet werden. Die folgenden vier Punkte sollen konkret angesprochen werden:

1. Übertragung der Attribute,
2. Übertragung der Relationen,
3. Übertragung der Schlüsselbedingungen und -beziehungen,
4. Übertragung weiterer Integritätsbedingungen.

Die Übertragung der Attribute ist einfach zu lösen, da das Attributkonzept im ODMG-93 Objektmodell mit dem des relationalen Modells nahezu identisch ist. Die meisten Datentypen, vor allem wenn sich die zugrundeliegenden Systeme am SQL-Standard orientieren, sind sehr leicht in entsprechende Literaltypen des ODMG-93 Objektmodells zu übertragen, denn in diesem Bereich ist eine starke Ausrichtung des ODMG-93 Standards am SQL-Standard wahrzunehmen.

Die naheliegende vollständige 1:1-Übertragung aller Relationen des relationalen Schemas in Objekttypen des ODMG-93 Schemas wäre ein einfacher und mit wenig Aufwand durchzuführender Transformationsansatz. Zwei Gründe sprechen jedoch gegen die Durchführung eines solchen Ansatzes. Zum einen bestehen zwischen den Konzepten Relation und Objekttyp Unterschiede. So läßt sich z. B. im ODMG-93 Objektmodell ein Realweltkonzept durch genau einen Objekttyp ausdrücken, während dasselbe Realweltkonzept im Relationenmodell über mehrere Relationen verteilt abgebildet wird. Deshalb sollten verfeinerte Verfahren angewendet werden. Zum Beispiel können Relationen, die m:n-Beziehungen zwischen zwei Relationen ohne zusätzliche Attribute abbilden, in m:n-Beziehungen zwischen den entsprechenden Objekttypen überführt werden. Durch Aggregation über mengenwertige und strukturierte Attribute lassen sich durch Normalisierung getrennte Relationen zu einem Objekttyp zusammenfassen. Zum anderen unterstützt dieser Ansatz die Fähigkeiten des ODMG-93 Objektmodells zur Darstellung von komplexen Zusammenhängen nicht, so können z. B. keine Sub-/Supertyp-Beziehungen eingesetzt werden. Einen Ansatz, wie in relationalen Schemata implizit bereits vorhandene Semantik bei der Übertragung der Relationen in das ODMG-93 Schema explizit gemacht werden kann, bietet [HK95]. Es existiert jedoch die Gefahr, daß es bei der Ausnutzung von semantisch reichen Konzepten wie der Generalisierung/Spezialisierung zu umfangreichen Remodellierungen der lokalen Schemata kommt, die aus zwei Gründen nicht wünschenswert sind. Zum einen wird der Aufwand für die

Transformation von Befehlen und Daten entlang der Schemaebenen erhöht, zum anderen kann sich die Zahl möglicher Schemakonflikte erhöhen. Ein Kompromiß zwischen einfacher Übertragung und aufwendiger Remodellierung ist empfehlenswert.

Einfache und zusammengesetzte Primärschlüsselbedingungen von Relationen können sehr einfach durch Schlüsselbedingungen des ODMG-93 Objektmodells ausgedrückt werden. Primär-/Fremdschlüsselbeziehungen zwischen Relationen lassen sich in bidirektionale Beziehungen zwischen den den Relationen entsprechenden Objekttypen umwandeln. Dabei wird in dem einen Objekttyp das Fremdschlüsselattribut durch einen *traversal path* ersetzt und in dem anderen Objekttyp der inverse *traversal path* hinzugefügt.

In den heutigen relationalen Systemen gibt es vielfältige Möglichkeiten zur deklarativen und prozeduralen Definition von einfachen und komplexen Integritätsbedingungen. Als Beispiele seien deklarative Konstrukte des SQL-Standards, wie `CHECK` oder `DEFAULT`, sowie weiterführende Konzepte, wie das Trigger-Konzept und das Konzept der Stored-Procedures, genannt. Dagegen ist im ODMG-93 Objektmodell nahezu ausschließlich die prozedurale Definition von Integritätsbedingungen möglich, so daß es zu ermitteln gilt, wie sich die Integritätsbedingungen übertragen lassen. Bedingungen über einzelnen Attributen, wie Defaultwerte oder `CHECK`-Klauseln, lassen sich über die Manipulierungsfunktion *set_value* der entsprechenden Attribute im ODMG-93 Schema implementieren. Bedingungen, die einzelne Relationen betreffen und durch Constraint-Klauseln repräsentiert sind, können in der Konstruktorfunktion *create* des der Relation entsprechenden Objekttyps realisiert werden. Für Bedingungen, die z. B. durch Trigger über mehrere Relationen definiert sind, müssen spezielle neue Operationen der zugehörigen Objekttypen implementiert werden, die sich eventuell gegenseitig aufrufen. Alle Integritätsbedingungen können übertragen werden. Mehr Möglichkeiten zur deklarativen Definition von Integritätsbedingungen wären wünschenswert.

Wegen der gegenüber dem ODMG-93 Objektmodell begrenzten Ausdrucksfähigkeiten des relationalen Modells soll die Transformation relationaler Schemata mit einer semantischen Anreicherung verbunden werden. Dies erfordert die Aneignung zusätzlichen Wissens über das zugrundeliegende Schema und den abgebildeten Weltausschnitt. Die zwei folgenden Möglichkeiten zur Umsetzung der gewonnenen Semantik im ODMG-93 Schema bieten sich an:

1. Remodellierung durch Verwendung der strukturellen Ausdrucksfähigkeiten,
2. Implementierung zusätzlicher Operationen zur Definition von Verhalten.

Eine Reihe von Beispielen läßt sich für die erste Möglichkeit anführen. Ein aus einer Relation gewonnener Objekttyp kann durch Vererbung verfeinert werden. Mehrere Objekttypen können unter einem abstrakten Supertyp zusammengefaßt werden. Die Probleme bei einer zu großen Remodellierung wurden oben bereits diskutiert. Wenn erkannt wird, daß unterschiedliche Relationen überlappende Realweltkonzepte und -klassen darstellen, dann muß remodelliert werden. Durch multiple Vererbung wird ein gemeinsamer Subtyp erzeugt, der als Extension alle gemeinsamen Tupel der Relationen aufnimmt. Mit der Definition von Operationen ist es möglich, den aus den Relationen gewonnenen Objekttypen Verhalten zuzuordnen. Damit wird eine realitätsnähere Abbildung des Weltausschnitts geschaffen.

Ein abschließendes Beispiel soll die bei einer Schematransformation vom relationalen Modell ins ODMG-93 Objektmodell notwendigen Schritte verdeutlichen.

Beispiel 6.1: Ein Schema LS_1 über eine Krankenhausverwaltung liegt in SQL-Notation vor:

```
CREATE TABLE patient (          CREATE TABLE arzt (
  Pat_Id    : NUMBER(6),          Arzt_Id   : NUMBER(2),
  Name      : CHAR(30),           Name      : CHAR(30),
  Geb_Dat   : DATE,               Geb_Dat   : DATE,
```

```
  PRIMARY KEY Pat_Id,
  );
CREATE TABLE fall (
  Fall_Id   : NUMBER(6),
  Pat_Id    : NUMBER(6),
  Aufn_Dat  : DATE,
  Entl_Dat  : DATE,
  PRIMARY KEY Fall_Id,
  FOREIGN KEY(Pat_Id)
    REFERENCES patient(Pat_Id)
  );
```

```
  PRIMARY KEY Arzt_Id,
  );
CREATE TABLE fall_arzt (
  Fall_Id   : NUMBER(6),
  Arzt_Id   : NUMBER(2),
  PRIMARY KEY(Fall_Id,Arzt_Id),
  FOREIGN KEY(Fall_Id)
    REFERENCES fall(Fall_Id),
  FOREIGN KEY(Arzt_Id)
    REFERENCES arzt(Arzt_Id)
  );
```

Durch Transformation in das ODMG-93 Objektmodell entsteht das vorläufige Schema KS_1, das in Abbildung 6.1 in graphischer Notation dargestellt ist:

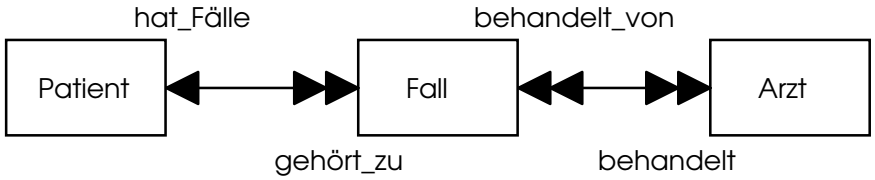

Abbildung 6.1: Das transformierte Schema KS_1 im ODMG-93 Objektmodell

Durch Wissensaneignung wird festgestellt, daß die Relationen Patient und Arzt Teilkonzepte eines übergeordneten Realweltkonzepts Person abbilden und daß die zugehörigen Realweltklassen sich überlappen können. Mit diesem Wissen wird das Schema KS_1 remodelliert. Das Ergebnis wird in Abbildung 6.2 in graphischer Notation und anschließend in ODL-Notation gezeigt.

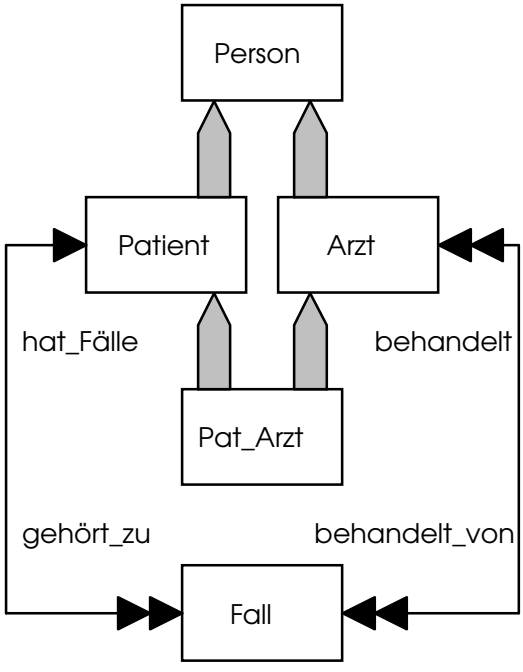

Abbildung 6.2: Das transformierte und angereicherte Schema KS_1 im ODMG-93 Objektmodell

```
interface person {
  extent personen;
  attribute string<30> name;
  attribute date geb_dat; };
```

```
interface pat_arzt
          :patient,arzt {
  extent kranke_ärzte; };
```

```
interface patient:person {
  extent patienten;
  key p_id;
  attribute integer[6] p_id;
  relationship Set<fall>
    hat_fälle inverse
    fall::gehört_zu; };
```

```
interface arzt:person {
  extent ärzte;
  key a_id;
  attribute integer[2] a_id;
  relationship Set<fall>
    behandelt inverse
    fall::behandelt_von; };
```

```
interface fall {
  extent fälle;
  key f_id;
  attribute integer[6] f_id;
  attribute date aufn_dat;
  attribute date entl_dat;
  relationship patient gehört_zu inverse patient::hat_fälle;
  relationship Set<arzt> behandelt_von
    inverse arzt::behandelt;};
```

6.1.3 Die Transformation weiterer Datenmodelle in das ODMG-93 Objektmodell

Das ER-Modell mit seinen zahlreichen Varianten und Erweiterungen besitzt seit Jahrzehnten eine besondere Bedeutung bei der Modellierung konzeptioneller Datenbankschemata. Deshalb soll hier eine kurze Betrachtung über Möglichkeiten zur Transformation von Schemata in ER-Modellen in ODMG-93 Schemata erfolgen. ER-Modelle verfügen über zwei Basisstrukturelemente, Entitytyp und Relationshiptyp. Entitytypen können leicht in Objekttypen übertragen werden, da es sich um verwandte Konzepte handelt. Auch bestehende Generalisierungs/ Spezialisierungshierarchien zwischen Entitytypen lassen sich in entsprechende Objekttyphierarchien überführen, wobei überlappenden Subtypen mit der Hilfe der multiplen Vererbung remodelliert werden müssen. Ein Relationshiptyp kann entweder als Objekttyp mit Beziehungen zu den zugehörigen Objekttypen oder nur als Beziehung zwischen den zugehörigen Objekttypen übertragen werden. Die letzte Möglichkeit kommt jedoch nur für binäre Relationships ohne Attribute in Frage. Wenn die Ausdrucksfähigkeiten der zugrundeliegenden Variante des ER-Modells zu begrenzt sind, sollte die Schematransformation mit einer semantischen Anreicherung verbunden werden. Dabei gelten grundsätzlich dieselben Möglichkeiten, die in Abschnitt 6.1.2 für das Relationenmodell diskutiert wurden.

Noch stärker als beim ER-Modell ist die Vielfalt unterschiedlicher objektorientierter Modelle, die zur Modellierung konzeptioneller Datenbankschemata eingesetzt werden. Sie versuchen zumeist, datenbankspezifische Konzepte mit Konzepten der objektorientierten Programmierung zu vereinen. Das ODMG-93 Objektmodell stellt wie bereits erwähnt einen Versuch der Hersteller von verbreiteten ODBMSen dar, ein einheitliches Modell zu propagieren, um unter anderem die Interoperabilität zwischen den Systemen zu fördern. Daher werden zu diesen ODBMSen inzwischen Schnittstellen angeboten, die eine Transformation eines konzeptionellen Schemas in ein ODMG-93 konformes Schema ermöglichen. Da dieser Standard einen Querschnitt und nicht die Summe aller in den Objektmodellen der ODBMSe vorhandenen Konzepte darstellt, ist bei der Schematransformation mit semantischem Verlust zu rechnen. Konkrete Untersuchungen hierzu wären notwendig. Für die Transformation objektorientierter Schemata ist im allgemeinen keine semantische Anreicherung notwendig.

Das ODMG-93 Objektmodell bietet Möglichkeiten, die Mehrzahl von Konzepten anderer Datenmodelle in modelleigene Konzepte zu überführen. Durch Remodellierung und Definition von Verhalten läßt sich für Schemata in bestimmten Datenmodellen, wie dem Relationen-

modell, ein semantisch reicheres Abbild des zugrundeliegenden Weltausschnitts darstellen. Als nachteilig erweisen sich die eingschränkten Möglichkeiten zur Definition von deklarativen Integritätsbedingungen im ODMG-93 Objektmodell. Da das ODMG-93 Objektmodell als Standard für objektorientierte Datenmodelle etabliert werden soll, besteht die Möglichkeit, daß Hersteller verbreiteter relationaler DBMSe in Zukunft eine Exportschnittstelle für das ODMG-93 Objektmodell anbieten. Damit würde die Basis für eine Integration heterogener, autonomer Datenbanksysteme mit dem ODMG-93 Objektmodell als GDM geschaffen.

6.2 Schemaintegration

Mit dem Prozeß der Schemaintegration soll durch die Schaffung eines föderierten Schemas eine gesamtheitliche Sicht auf alle exportierten Daten der beteiligten CDBSe erzeugt werden. Dazu müssen alle Schemaelemente der Exportschemata in das föderierte Schema integriert werden, wobei alle existierenden Schemakonflikte aufgelöst werden müssen. Voraussetzung für eine erfolgreiche Schemaintegration ist die Unterstützung durch das GDM. Die folgende Untersuchung wird zeigen, inwieweit eine erfolgreiche Schemaintegration mit dem ODMG-93 Objektmodell als GDM möglich ist. Zuerst werden dazu die in Kapitel 5 gestellten Anforderungen mit den Möglichkeiten des ODMG-93 Objektmodells verglichen. Anschließend wird geprüft, wie sich allgemeine Konfliktlösungstechniken an das ODMG-93 Objektmodell anpassen lassen. Dann wird die Integration von semantisch vergleichbaren Objekttypen unter Nutzung von Konzepten des ODMG-93 Objektmodells untersucht. Abschließend folgt die bereits angekündigte Diskussion existierender Lösungen zur Schemaintegration mit dem ODMG-93 Objektmodell.

6.2.1 Schemaintegrationsmöglichkeiten mit dem ODMG-93 Objektmodell

Bevor das ODMG-93 Objektmodell bezüglich spezieller Unterstützung für die Schemaintegration untersucht wird, gilt es, einige allgemeine Aussagen über grundsätzliche Möglichkeiten zur Schemaintegration mit dem ODMG-93 Objektmodell zu treffen. Dazu wird den in Abschnitt 5.2.1 aufgestellten Integrationsschritten sowie den in den Abschnitten 5.2.2 und 5.2.3 diskutierten Anforderungen an das GDM gefolgt. Außerdem werden in diesem Abschnitt einige Annahmen für die Untersuchung getroffen.

Der erste Schritt der Schemaintegration ist die Auswahl einer geeigneten Lösungsstrategie. Die Auswahl einer binären Strategie (linear oder balanciert) erscheint wünschenswert, da die Komplexität begrenzt und eine softwaremäßige Unterstützung erleichtert wird [Rah94]. Für die Untersuchung bedeutet dies, daß die Integration von zwei Exportschemata hinreichend für die Eignungsbewertung ist.

Der zweite Schritt ist die Analyse und der Vergleich der Exportschemata hinsichtlich der überlappenden Diskursbereiche und der dabei entstehenden Schemakonflikte. Um die Schemaintegration zu erleichtern, sollten alle Schemakonflikte fixiert werden. Die bereits mehrfach erwähnten *correspondence assertions* bieten dazu eine formelbasierte Möglichkeit.

Der dritte Schritt ist die Phase der Erstellung des föderierten Schemas, das die Forderungen von [BLN86] nach Vollständigkeit und Korrektheit, Minimalität und Verständlichkeit erfüllt. Die dabei zu erfüllende Hauptaufgabe ist die Auflösung von Schemakonflikten. Wie sich dabei allgemeine datenmodellunabhängige Konfliktlösungstechniken an das ODMG-93 Objektmodell anpassen lassen, wird im nächsten Abschnitt untersucht. Die Unterstützung des ODMG-93 Objektmodells für die von [SCG91] aufgestellten Charakteristiken hinsichtlich der Ausdrucksfähigkeiten wurde in Abschnitt 6.1.1 behandelt. Wie die vorhandenen Konzepte zur Schemaintegration genutzt werden können, wird im übernächsten Abschnitt untersucht.

In der vorliegenden Fassung bietet der ODMG-93 Standard wenig technische Unterstützung für eine Schemaintegration. Multiple Schemata, d. h. die Verwaltung mehrerer Schemata, die aufeinander aufbauen, werden nicht unterstützt. Da das ODMG-93 Modell als Standard für objektorientierte Datenbankmanagementsysteme entwickelt wurde, bietet es auch fast keine Unterstützung für die in Charakteristik 6 geforderten Integrationsoperationen. Ein explizites Ableitungskonzept, das die in Charakteristik 7 geforderten Bedingungen erfüllt, ist nicht vorhanden, aber ein Ansatz dafür ist durch die Anfragesprache OQL gegeben. Daher ist eine Erweiterung des Standards hinsichtlich der fehlenden Funktionalität notwendig. Im Anschluß an die Untersuchung werden zwei Ansätze [BFN94, Rad94] diskutiert, in denen eine Erweiterung des Standards vorgenommen wurde.

Für die Untersuchung ist es notwendig, nachfolgend einige Annahmen zu treffen. Zwei Exportschemata ES_1 und ES_2 sollen zu einem föderierten Schema *FS* integriert werden. *SN*.X{X_1..X_n} mit $SN \in \{ES_1, ES_2, FS\}$ bezeichnet einen Objekttyp X des Schemas *SN* mit den zugehörigen Attributen X_1..X_n. *SN*.X:Y1..YN bezeichnet eine Subtypbeziehung des Objekttyps X zu den Objekttypen Y1 bis YN. *SN*.X{RY}/Y{RX} bezeichnet eine bidirektionale Beziehung zwischen den Objekttypen X und Y desselben Schemas *SN*, wobei RY und RX die zugehörigen *traversal paths* bezeichnen.

6.2.2 Basistechniken zur Schemaintegration

In den nachfolgenden Abschnitten wird die Anpassung von allgemeinen Konfliktlösungstechniken für Namens-, Attribut- und Strukturkonflikte an das ODMG-93 Objektmodell untersucht und beschrieben. Mögliche Probleme sollen aufgezeigt werden. Die notwendige technische Unterstützung durch Integrationsoperationen oder Ableitungsmechanismen wird dabei diskutiert.

6.2.2.1 Namenskonfliktauflösung

Die Auflösung von Namenskonflikten erfolgt während der gesamten Phase der Erstellung des föderierten Schemas. Die hierfür am häufigsten verwendete Konfliktlösungstechnik ist das Umbenennen [Dup94], d. h. ein Schemaelement wird bei der Übertragung vom Exportschema ins föderierte Schema mit einem neuen Namen versehen. Werden die Schemaelemente des föderierten Schemas durch Ableitungsoperationen aus den Schemaelementen der Exportschemata abgeleitet, ist die Verwendung neuer Namen leicht möglich. Ansonsten muß eine explizite Operation zum Umbenennen von Schemaelementen unterstützt werden. Die Entscheidung, welche Namen verwendet werden, sollte manuell getroffen werden und der Forderung nach Verständlichkeit Rechnung tragen.

6.2.2.2 Attributintegration

Attribute sind im ODMG-93 Objektmodell wie in vielen anderen Datenmodellen die Hauptträger der verwalteten Daten. Sie sind im Objektmodell an Objekttypen gebunden; das bedeutet, sie werden mit den zugehörigen Objekttypen integriert. Von besonderem Belang sind für die Schemaintegration gemeinsame Attribute semantisch vergleichbarer Objekttypen. Für diese Attribute gilt es eine Integrationstechnik zu finden, so daß sie im föderierten Schema nur durch ein Attribut vertreten sind. Alle anderen Attribute werden unverändert in das föderierte Schema übernommen.

Bei der Integration gemeinsamer Attribute müssen existierende Ausprägungen der in Abschnitt 4.2.4 beschriebenen Attributkonflikte aufgelöst werden. Die hierfür empfohlene Konfliktlösungstechnik ist die Aufstellung von Transformationsfunktionen [Rah94], die eine eineindeutige Abbildung zwischen Attributwerten ermöglichen sollen. Mit Hilfe der Transformationsfunktionen läßt sich eine Technik zur Integration gemeinsamer Attribute

angeben. Im föderierten Schema wird dabei ein neues Attribut erzeugt, das durch Transformationsfunktionen mit den Attributen der Exportschemata verbunden wird. Abbildung 6.3 zeigt die Integration zweier semantisch vergleichbarer Attribute ES_1.A{A_i} und ES_2.B{B_j} zu einem Attribut FS.AB{AB_{ij}} über die Transformationsfunktionen TFA und TFB mit den zugehörigen Umkehrfunktionen TFA' und TFB'.

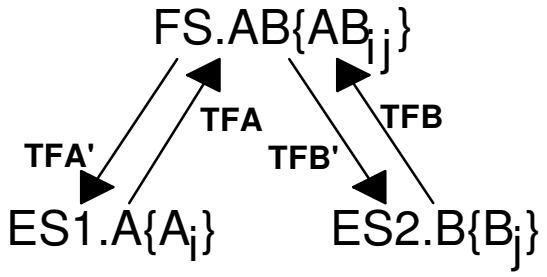

Abbildung 6.3: Integration von gemeinsamen Attributen mit Transformationsfunktionen

Die Umkehrfunktionen sind notwendig, um Änderungen durch globale Nutzer zu ermöglichen. Eine Transformationsfunktion kann auch zwischen einem Attribut einerseits und mehreren Attributen andererseits erzeugt werden, um den Konflikt 1 Attribut versus n Attribute zu lösen. Während die Mehrzahl der Transformationsfunktionen manuell aufgestellt werden kann, könnten Funktionen zwischen unterschiedlichen Datentypen, z. B. eine Funktion *string_nach_integer*, durch das FDBMS unterstützt werden.

Eine entscheidende Frage, die es zu untersuchen gilt, ist, ob es mit Transformationsfunktionen möglich ist, alle Beschreibungskonflikte zwischen Attributen zu lösen. Ebenfalls wichtig ist die Frage nach der Art des integrierten Attributes. Dazu wird versucht, für die in Beispiel 4.10 aufgestellten Konflikte die Attribute zu integrieren und die notwendigen Transformationsfunktionen auf Bijektivität zu untersuchen.

Beispiel 6.2: Die in Beispiel 4.10 beschriebenen Schemata ES_1 und ES_2 sollen integriert werden. Das folgende föderierte Schema *FS* in ODMG-93 ODL wird gebildet:

```
interface adresse {
  attribute string<30> str;
  attribute integer nr;
  attribute string plz_ort; };
interface restaurant {
  attribute float nettokosten;
  attribute float bruttokosten; };
```

Zwischen den Attributen von *FS*.adresse und ES_1.adresse{string<30> str; string plz_ort} sowie ES_2.adresse{integer nr} sind die Transformations- sowie Umkehrfunktionen trivial und bijektiv, da sich die Attribute 1:1 aufeinander abbilden. Gleiches gilt für *FS*.restaurant {float nettokosten} und ES_2.restaurant{float kosten}, sowie *FS*.restaurant{float bruttokosten} und ES_1.restaurant{float kosten}.

Zwischen *FS*.adresse{string<30> str} und ES_2.adresse{string<50> str} läßt sich keine bijektive Transformationsfunktion aufstellen, so daß ein semantischer Verlust auftritt, d. h. nicht alle in der Datenbank gespeicherten Werte von ES_2.adresse{string<50> str} können dem globalen Nutzer vollständig sichtbar gemacht werden. Aber auch bei der Wahl von *FS*.adresse{string<50> str} als integriertem Attribut würde ein semantischer Verlust auftreten, da nicht alle globalen Änderungen in ES_1.adresse{string<30> str} gespeichert werden können.

Zwischen *FS*.adresse{integer nr} und ES_1.adresse{string nr} wird eine Funktion zur Umwandlung von alphanumerischen Werten in numerische Werte benötigt. Auch hierbei ergibt sich ein semantischer Verlust, wenn in ES_1.adresse{string nr} Werte mit nicht-numerischen Zeichen gespeichert sind.

Zwischen *FS*.adresse{string plz_ort} und ES_2.adresse{string plz; string ort} läßt sich eine Transformationsfunktion verwenden, die Stringwerte zusammenhängt und trennt. Ein semantischer Verlust sollte dabei nur in Ausnahmefällen auftreten.

Eine bijektive Transformationsfunktion für das *meal-cost* Problem läßt sich nur aufstellen, wenn Steuer und Trinkgeld sich eindeutig bestimmen lassen. Beide Attribute werden dann als integrierte Attribute genommen, da nur so die Forderung nach Vollständigkeit erfüllt wird.

Das Beispiel macht deutlich, daß eine semantisch verlustfreie Integration von gemeinsamen Attributen, die die Forderungen von [BLN86] erfüllt, mit dem ODMG-93 Objektmodell derzeit nicht möglich ist. Durch sinnvolle Kompromisse zwischen entstehendem semantischen Verlust und zu erwartendem Gewinn durch Präsentation eines gemeinsamen Attributes im föderierten Schema sollte es jedoch möglich sein, die Mehrzahl der gemeinsamen Attribute mit Hilfe von Transformationsfunktionen zu integrieren.

Eine Möglichkeit, Attribute ohne semantischen Verlust zu integrieren, ergibt sich, wenn die Technik der extensionalen Zerlegung verwendet wird. Bei dieser Technik wird ein Realweltkonzept maximal, bei Überlappung der Realweltklassen, durch drei Objekttypen, die Ausgangsobjekttypen der Exportschemata sowie ein Objekttyp mit integrierten Attributen wie in Beispiel 6.2, im föderierten Schema repräsentiert. Dieser Objekttyp hat als Instanzen nur Objekte, die auf realweltäquivalente Objekte in unterschiedlichen Datenbanken zurückgeführt werden können. Für diesen Objekttyp ist die Bijektivität der Transformationsfunktionen gegeben, da die Ausgangswerte identisch sein müssen. Jedoch läßt sich diese Technik nicht an das Konzept der Generalisierung/ Spezialisierung, mit dessen Hilfe semantisch vergleichbare Objekttypen im ODMG-93 Objektmodell integriert werden sollen, anpassen, da nur identische Attribute von Supertypen einmal in gemeinsamen Subtypen enthalten sind. Notwendig wären Mechanismen, die eine Konfliktauflösung bei der Attributvererbung ermöglichen würden.

Das ODMG-93 Objektmodell bietet durch die Möglichkeit, das Verhalten von Objekttypen durch Operationen darzustellen, eine gute Unterstützung für die Implementation der Transformationsfunktionen. Die für jedes Attribut eines Objekttyps vorhandenen Operationen *get_value* und *set_value* sind prädestiniert, als Implementationsgrundlage zu dienen.

6.2.2.3 Auflösung von Strukturkonflikten

Die prinzipielle Technik zum Auflösen von Strukturkonflikten ist die Angleichung der Strukturelemente der beteiligten Schemaelemente. Danach kann die Integration dieser Schemaelemente auf andere Integrationstechniken zurückgeführt werden.

Das erste zu lösende Problem ist, welches Strukturelement an welches angeglichen wird. Die Angleichung an Basisstrukturelemente, wenn sie an einem Konflikt beteiligt sind, ist ein praktikabeler Weg, da sie andere Strukturelemente einschließen. Für das ODMG-93 Objektmodell läßt sich also fordern, den Konflikt Objekttyp versus Attribut(e), den Objekttyp/ Beziehungs-Konflikt und die strukturelle Fragmentierung durch Überführung der Attribut(e) und Beziehung(en) in Objekttypen zu lösen. Für den Konflikt Beziehung versus Attribut(e) ergibt sich dann, daß die Attribut(e) durch eine Beziehung ersetzt werden.

Das zweite zu lösende Problem ist die technische Realisierung der Angleichung. Mit Ableitungsmechanismen auf der Basis von OQL-Anfragen läßt sich die Auflösung der Strukturkonflikte mit den Integrationstechniken zur Integration semantisch vergleichbarer

Objekttypen verbinden, da mit der OQL Objekte aus Attributen und Beziehungen erzeugt werden können. Operationen zum Objektifizieren von Attributen und Beziehungen wären als Integrationsoperationen notwendig, werden aber vom ODMG-93 Objektmodell nicht unterstützt.

6.2.3 Integration von semantisch vergleichbaren Objekttypen

Die Integration von semantisch vergleichbaren Objekttypen ist das Kernstück der Schemaintegration mit dem ODMG-93 Objektmodell, denn alle Konflikte außer Namenskonflikten existieren zwischen semantisch vergleichbaren Objekttypen (semantische und Beschreibungskonflikte) oder können daauf zurückgeführt werden (Strukturkonflikte, siehe Abschnitt 6.2.2.3). Die Integration zerfällt dabei in zwei Abschnitte, die Auswahl einer Integrationstechnik zur Lösung von semantischen und Eigenschaftsmengenkonflikten und die Bildung der Objekttypen des föderierten Schemas. Im zweiten Abschnitt werden die Attribute mit der in Abschnitt 6.2.2.2 beschriebenen Technik sowie die Beziehungen und Schlüsselbedingungen integriert und existierende Attribut/Metainfo-Konflikte gelöst. Dementsprechend sind die nachfolgenden Abschnitte gegliedert. In den ersten beiden Abschnitten werden Integrationstechniken für die Integration von zwei und mehreren semantisch vergleichbaren Objekttypen vorgestellt. Daran schließen sich Betrachtungen zur Integration von Beziehungen und Schlüsselbedingungen sowie zur Auflösung von Attribut/Metainfo-Konflikten an.

6.2.3.1 Integrationstechniken für zwei semantisch vergleichbare Objekttypen

Nach [Dup94] lassen sich fünf Integrationstechniken für zwei Entitytypen, die sich in semantischem und/oder Eigenschaftsmengenkonflikt befinden, unterscheiden. Diese Techniken basieren auf einem "idealen" Datenmodell, das die Charakteristiken von [SCG91] erfüllt. Es ist zu untersuchen, ob und wie diese Techniken sowie eine weitere Technik durch Konzepte des ODMG-93 Objektmodells unterstützt werden.

Eine von [Dup94] diskutierte Integrationstechnik, Multi-Instantiation, das bedeutet die Übernahme beider Objekttypen in das föderierte Schema, wobei äquivalente Objekte als Instanzen beider Objekttypen ausgewiesen werden, wurde bereits in Abschnitt 5.2 als untauglich für die Schemaintegration ausgewiesen. Außerdem kann diese Technik nicht mit dem ODMG-93 Objektmodell als GDM genutzt werden, da das Konzept der multiplen Instanzen in der aktuellen Revision nicht vorgesehen ist.

Zur Untersuchung der Integrationstechniken werden folgende Annahmen getroffen. In den Exportschemata existieren zwei semantisch vergleichbare Objekttypen ES_1.A($A_1..A_h$, $A_{h+1}..A_m$) und ES_2.B($B_1..B_k$, $B_{k+1}..B_n$) mit h,k,m,n ≥ 0. Die Attribute $A_1..A_h$ und $B_1..B_k$ sind gemeinsame Attribute und können zu Attributen $AB_1..AB_l$ des föderierten Schemas integriert werden. Die Attribute $A_{h+1}..A_m$ und $B_{k+1}..B_n$ sind unabhängige Attribute, d. h. sie sind semantisch nicht miteinander vergleichbar. Sie werden unverändert als Attribute $A'_{h+1}..A'_m$ und $B'_{k+1}..B'_n$ in das föderierte Schema übernommen. Zwischen beiden Objekttypen existiert basierend auf den in den Abschnitten 4.2.3 und 4.2.4 eingeführten Formeln jeweils eine der folgende Beziehungen:

- ES_1.A *mop* ES_2.B mit *mop* ∈ ($\equiv_S$, $\subseteq_S$, $\supseteq_S$, $\cap_S$, $\varnothing_S$) und
- ES_1.A *bop* ES_2.B mit *bop* ∈ ($\equiv_B$, $\subseteq_B$, $\supseteq_B$, $\cap_B$, $\varnothing_B$).

Ziel ist die Integration beider Objekttypen im föderierten Schema *FS* in ODMG-93, so daß eventuelle semantische und/oder Eigenschaftsmengenkonflikte gelöst werden. Alle Techniken beruhen darauf, daß ein Realweltkonzept nur durch Anwendung von Generalisierung/ Spezialisierung zerlegt werden darf. Zuerst werden die Techniken vorgestellt und beschrieben. Dabei wird die Verteilung der gemeinsamen und unabhängigen Attribute untersucht. Die

optimale Einsetzbarkeit der Techniken für die Konfliktlösung wird unter den folgenden beiden Prämissen betrachtet:

- Ein semantischer Konflikt ist optimal gelöst, wenn die durch extensionalen Vergleich der Objekttypen entstehenden Mengen ES_1.A \ ES_2.B, ES_1.A ∩ ES_2.B und ES_2.B \ ES_1.A, wenn sie eigene Elemente besitzen, durch jeweils einen eigenen Objekttyp in einer Hierarchie abgebildet werden. Damit können globale Einfügungen, Änderungen und Löschungen korrekt auf den Datenbanken der CDBSe durchgeführt werden.
- Ein Eigenschaftsmengenkonflikt ist optimal gelöst, wenn gemeinsame Attribute genau einmal definiert werden und durch die unabhängigen Attribute bei der Integration der Objekte keine zusätzlichen Nullwerte entstehen.

Aus diesen Prämissen entstehende Probleme mit anderen Forderungen werden anschließend betrachtet. Es folgen die Beschreibungen der Konfliktlösungstechniken.

- **Merging**

  Beim Merging wird aus den zwei Objekttypen ES_1.A und ES_2.B ein Objekttyp FS.A_B(AB_1..AB_l, A'_{h+1}..A'_m, B'_{k+1}..B'_n) gebildet, d. h. alle gemeinsamen und alle unabhängigen Attribute werden in einem Objekttyp vereinigt. Abbildung 6.4 zeigt den gemischten Objekttyp FS.A_B.

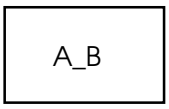

Abbildung 6.4: Merging von ES_1.A und ES_2.B zu FS.A_B

  Merging ist optimal einsetzbar, wenn kein semantischer Konflikt vorliegt, d. h. wenn alle Objekte in den zugehörigen Datenbanken jeweils Paare von Proxy-Objekten desselben Realweltobjekts sind. Nullwerte können bei keinem der möglichen Eigenschaftsmengenkonflikte entstehen, da sich die unabhängigen Attribute für jedes globale Objekt ergänzen.

- **Sub-Class**

  Bei der Technik Sub-Class werden aus den zwei Objekttypen ES_1.A und ES_2.B zwei Objekttypen FS.B'(AB_1..AB_l, B'_{k+1}..B'_n) und FS.A':B'(A'_{h+1}..A'_m) oder FS.A'(AB_1..AB_l, A'_{h+1}..A'_m) und FS.B':A'(B'_{k+1}..B'_n) gebildet, d. h. im jeweiligen Supertyp werden die gemeinsamen Attribute definiert. Die unabhängigen Attribute werden entsprechend den zugrundeliegenden Objekttypen verteilt. Abbildung 6.5 zeigt die zwei Möglichkeiten der Sub-Class Technik.

Abbildung 6.5: Aufbau von Vererbungsbeziehungen zwischen FS.A' und FS.B'

  Sub-Class ist optimal einsetzbar, wenn extensionale Inklusion vorliegt, d. h. wenn alle Instanzen eines Objekttyps, der dann Subtyp wird, realweltäquivalent mit Instanzen des anderen Objekttyps sind. Dies gilt für alle Eigenschaftsmengenkonflikte.

- **Union**

  Bei der Technik Union werden die Objekttypen ES_1.A und ES_2.B als FS.A'(A'_{h+1}..A'_m) und FS.B'(B'_{k+1}..B'_n) mit den jeweils unabhängigen Attributen in das föderierte Schema

übernommen, und es wird ein gemeinsamer Supertyp *FS*.A_B(AB_1..AB_l) gebildet, der die gemeinsamen Attribute enthält. Abbildung 6.6 zeigt das Ergebnis dieser Technik im föderierten Schema *FS*.

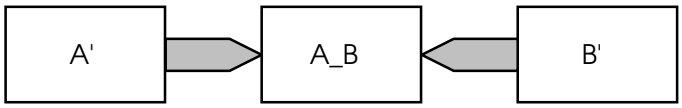

Abbildung 6.6: Bildung eines Supertyps FS.A_B von FS.A' und FS.B'

Union ist optimal einsetzbar bei extensionaler Disjunktheit, d. h. wenn keine realweltäquivalenten Instanzen der beiden Objekttypen existieren.

- **Intersection**

Bei der Technik Intersection werden die Objekttypen ES_1.A und ES_2.B als *FS*.A'(AB_1..AB_l, A'_{h+1}..A'_m) und *FS*.B'(AB_1..AB_l, B'_{k+1}..B'_n) so in das föderierte Schema übernommen, daß die gemeinsamen Attribute in beide Objekttypen integriert werden. Dann wird ein gemeinsamer Subtyp *FS*.A_B gebildet, der die gemeinsamen und unabhängigen Attribute erbt. Abbildung 6.7 zeigt das Ergebnis dieser Technik im föderierten Schema *FS*.

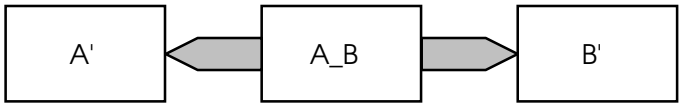

Abbildung 6.7: Bildung eines Subtyps FS.A_B von FS.A' und FS.B'

Offenbar ist Intersection nur optimal einsetzbar, wenn keine gemeinsamen Attribute existieren und sich die Objekttypen extensional überlappen.

- **Union/Intersection-Kombination**

Die Integrationstechnik Union/Intersection-Kombination ist eine speziell auf das im ODMG-93 Objektmodell angewandte Konzept der exklusiven Vererbung zugeschnittene Möglichkeit zur Integration zweier semantisch vergleichbarer Objekttypen. Dabei werden die Objekttypen ES_1.A und ES_2.B als *FS*.A'(A'_{h+1}..A'_m) und *FS*.B'(B'_{k+1}..B'_n) in das föderierte Schema übernommen. Die gemeinsamen Attribute enthält ein gemeinsamer Supertyp *FS*.A_B_1(AB_1..AB_l). Ein gemeinsamer Subtyp FS.A_B_2 erbt alle Attribute. Abbildung 6.8 zeigt das Ergebnis dieser Technik im föderierten Schema *FS*.

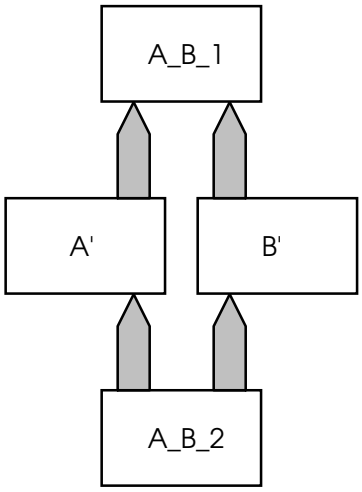

Abbildung 6.8: Bildung eines Super- und eines Subtyps

Diese Technik ist optimal einsetzbar bei extensionaler Überlappung, wenn gemeinsame Attribute existieren.

Mit den Ausdrucksfähigkeiten des ODMG-93 Objektmodells ist es möglich, die fünf beschriebenen Integrationstechniken zu realisieren. Die technische Realisierung ist jedoch nicht ohne weiteres möglich. Auf der Basis des geforderten aber nur im Ansatz vorhandenen Ableitungskonzeptes ließen sich alle Techniken realisieren. Ansonsten wären Integrationsoperationen zum Mischen von Objekttypen, zum Aufbau von Vererbungsbeziehungen, zur Bildung von Supertypen (upward inheritance) und Subtypen notwendig.

Unter den obengenannten Prämissen konnte festgestellt werden, daß die Integrationstechniken ausreichen, um alle Konfliktkombinationen optimal zu lösen. Dabei kann es jedoch dazu kommen, daß mehrere Objekttypen in einer Hierarchie dieselben Eigenschaften besitzen, was nicht den Forderungen nach Minimalität und Verständlichkeit entspricht. Unter Abschwächung der Prämissen, z. B. durch Verzicht auf extensionale Zerlegung, können deshalb die Integrationstechniken für weitere Konfliktkombinationen eingesetzt werden. Die folgende Tabelle zeigt zusammenfassend die Konfliktkombinationen und Integrationstechniken unter den folgenden Gesichtspunkten zur Einsetzbarkeit:

- optimal: Konflikte werden optimal gelöst,
- optimal(*): Konflikte werden optimal gelöst, Minimalität wird nicht erreicht,
- möglich: Einsatz der Technik bei nichtoptimaler Konfliktlösung möglich,
- nein: Einsatz der Technik ist nicht möglich.

In der Spalte Objektmengen wird die extensionale Beziehung und in der Spalte Eigenschaftsmengen wird die intensionale Beziehung zwischen den Objekttypen betrachtet. Auf $A \supseteq_S B$ wird aus Redundanzgründen verzichtet.

Objekt-mengen	Eigenschafts-mengen	Merging	Sub-Class	Union	Inter-section	Union/ Intersection
$A \equiv_S B$	$A \equiv_B, \subseteq_B, \supseteq_B, \cap_B B$	optimal	möglich	nein	möglich	möglich
$A \equiv_S B$	$A \varnothing_B B$	optimal	möglich	nein	möglich	nein
$A \subseteq_S B$	$A \equiv_B, \subseteq_B B$	möglich	optimal(*)	nein	möglich	möglich
$A \subseteq_S B$	$A \supseteq_B, \cap_B B$	möglich	optimal	nein	möglich	möglich
$A \subseteq_S B$	$A \varnothing_B B$	möglich	optimal	nein	möglich	nein
$A \cap_S B$	$A \equiv_B, \subseteq_B, \supseteq_B B$	möglich	nein	nein	möglich	optimal(*)
$A \cap_S B$	$A \cap_B B$	möglich	nein	nein	möglich	optimal
$A \cap_S B$	$A \varnothing_B B$	möglich	nein	nein	optimal	nein
$A \varnothing_S B$	$A \equiv_B, \subseteq_B, \supseteq_B B$	möglich	nein	optimal(*)	nein	nein
$A \varnothing_S B$	$A \cap_B B$	möglich	nein	optimal	nein	nein

Tabelle 6.2: Einsetzbarkeit der Integrationstechniken zur Konfliktlösung

6.2.3.2 Integrationstechniken für mehrere semantisch vergleichbare Objekttypen

Wegen der in Abschnitt 4.2.3 behandelten semantischen Fragmentierung müssen in manchen Fällen mehr als zwei semantisch vergleichbare Objekttypen gleichzeitig integriert werden. Deshalb sind für diese Fälle weitergehende Integrationstechniken anzuwenden, die jedoch ebenfalls zum großen Teil auf dem Konzept der Generalisierung/Spezialisierung beruhen. Aus Zeit- und Platzgründen sollen hier nur einige ausgewählte Beispiele dargestellt werden. Auf die Beschreibung der Attributverteilung wird verzichtet.

Für Klassifikationskonflikte werden folgende vereinfachende Annahmen getroffen. Es existieren semantisch vergleichbare Objekttypen ES_1.A, ES_2.B und ES_2.C. Sie sollen unter denselben Prämissen wie im letzten Abschnitt integriert werden. Es soll untersucht werden, ob sich die bereits beschriebenen Techniken für Klassifikationskonflikte erweitern lassen. Die folgenden drei Fälle ergeben sich:

1. ES_1.A $\equiv_S$ ES_2.B $\cup$ ES_2.C. In diesem Fall können die Objekttypen zu einem Objekttyp *FS*.A_B_C zusammengefaßt werden. Dies entspricht einer Erweiterung der Technik Merging aus dem letzten Abschnitt.

2. ES_1.A $\supseteq_S$ ES_2.B $\cup$ ES_2.C. In diesem Fall wird ES_1.A als *FS*.A' in das föderierte Schema übernommen. Die Objekttypen ES_2.B und ES_2.C werden als Subtypen *FS*.B':*FS*.A und *FS*.C':*FS*.A integriert. Dies ist eine Erweiterung der Technik Sub-Class.

3. ES_1.A $\subseteq_S$ ES_2.B $\cup$ ES_2.C und ES_1.A $\cap_S$ ES_2.B $\cup$ ES_2.C. In diesem Fall können die Intersection und die Union/Intersection Technik erweitert und kombiniert werden. Durch die Bildung von gemeinsamen Subtypen werden den jeweiligen realweltäquivalenten Objekten eigene Objekttypen zugeordnet. Durch die Bildung von gemeinsamen Supertypen werden gemeinsame Eigenschaften erfaßt.

Es zeigt sich, daß Klassifikationskonflikte mit Erweiterungen und Kombinationen der im letzten Abschnitt beschriebenen Integrationstechniken lösbar sind.

Bei Vererbungskonflikten liegen in den Exportschemata bereits Generalisieruns/ Spezialisierungshierarchien vor. Bei der Integration kommt es darauf an, im föderierten Schema eine vereinigte Hierarchie zu erzeugen, so daß die Prämissen erfüllt werden. Grundsätzlich lassen sich auch hier dieselben Techniken basierend auf denselben Einsetzbarkeitsregeln wie oben verwenden. Objekttypen, die durch eine Differenzoperation bei der Konflikterkennung subtrahiert wurden, können unverändert übernommen werden.

6.2.3.3 Integration von Beziehungen

Die Art und Weise der Integration von Beziehungen von zwei semantisch vergleichbaren Objekttypen ist aufgrund der Struktur dieser Schemaelemente ein sehr komplexes Problem. Beziehungen im ODMG-93 Objektmodell sind keine eigenständigen Strukturelemente, sondern sie sind Eigenschaften zwischen genau zwei Objekttypen. Deshalb können die in der Literatur vorgeschlagenen Integrationstechniken für Beziehungen nicht ohne weiteres auf das ODMG-93 Objektmodell übertragen werden, da sie zumeist für das relationale Modell oder das ER-Modell entworfen wurden. Bei der Integration von Beziehungen sind weiterhin eine Reihe unterschiedlicher Fälle zu betrachten, wobei hier eine Beschränkung auf die Integration von gemeinsamen Beziehungen von semantisch vergleichbaren Objekttypen erfolgt. Existierende Kardinalitätskonflikte müssen bei der Integration gelöst werden. Abbildung 6.9 zeigt eine nachfolgend erläuterte Erweiterung der in Abschnitt 6.2.3.1 getroffenen Annahmen.

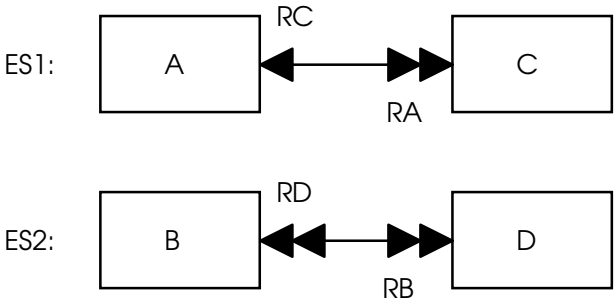

Abbildung 6.9: Zwei Exportschemata ES_1 und ES_2

Sowohl die Objekttypen *ES1*.A und *ES2*.B als auch die Objekttypen *ES1*.C und *ES2*.D sind jeweils semantisch vergleichbar. *ES1*.A{RC}/*ES1*.C{RA} und *ES2*.B{RD}/*ES2*.D{RB} bilden dieselbe Realweltbeziehung ab. Die Möglichkeiten zur Integration der beide Beziehungen richten sich nach den Konfliktlösungstechniken, die für die beteiligten Objekttypen verwendet werden. Die wünschenswerteste Möglichkeit ist das Mischen der beiden Ausgangsbeziehungen zu einer einzigen Beziehung im föderierten Schema, was man als Relationship_Merging (R_Merging) bezeichnen kann. Diese Möglichkeit läßt sich für alle Integrationstechniken, bei denen ein Objekttyp die gemeinsamen Eigenschaften enthält, d. h. alle außer Intersection, verwirklichen. Für den Fall, daß Intersection auf einer Seite verwendet wird, läßt sich die Möglichkeit R_Teilmerging angeben, bei der die zwei Beziehungen in das föderierte Schema übernommen werden, aber auf der Intersectionseite durch Namensangleichung der traversal paths nur einmal an den gemeinsamen Subtyp vererbt werden. Die Abbildungen 6.10 und 6.11 zeigen jeweils ein Beispiel für R_Merging und R_Teilmerging. Dabei wurden jeweils unterschiedliche Konfliktlösungstechniken für die Objekttypen verwendet.

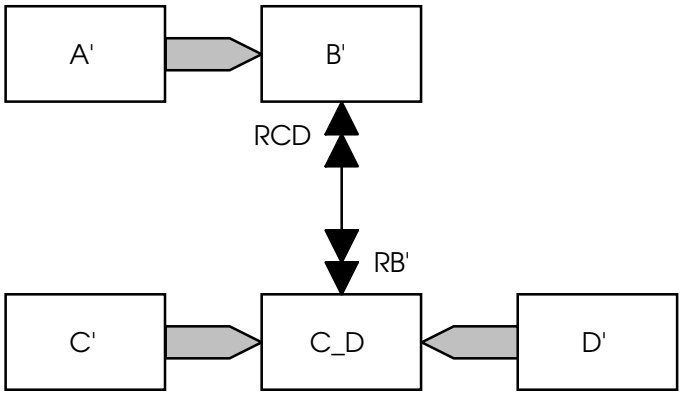

Abbildung 6.10: Beispiel für R_Merging

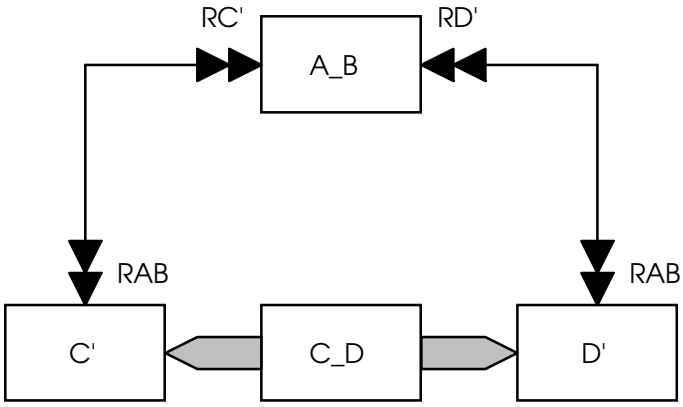

Abbildung 6.11: Beispiel für R_Teilmerging

Bei einigen Kombinationen kann es bei Verwirklichung von R_Merging oder R_Teilmerging zu Problemen bei Einfügungen oder Änderungen von Beziehungsinstanzen durch FDBS-Nutzer kommen, d. h. der FDBS-Nutzer kann Beziehungen zwischen Objekten aus unterschiedlichen Datenbanken aufbauen. So kann im Beispiel aus Abbildung 6.10 eine Beziehung zwischen einer direkten Instanz von *FS*.B' und einer Instanz von *FS*.C' eingefügt werden, obwohl beide Objekttypen keine direkten Instanzen enthalten, die jeweils in beiden lokalen Systemen gespeichert sind. Soll dies ermöglicht werden, erfordert dies die globale Speicherung von Referenzinformationen.

Bei der Integration gemeinsamer Beziehungen durch R_Merging und R_Teilmerging gilt es existierende Kardinalitätskonflikte aufzulösen. Im ODMG-93 ist für jede Seite einer Beziehung eine der drei Kardinalitäten *List*, eine geordnete Menge von Referenzen auf den anderen Objekttyp, *Set*, eine ungeordnete Menge von Referenzen auf den anderen Objekttyp, sowie *1*, maximal eine Referenz auf den anderen Objekttyp, möglich. Ein Konflikt zwischen *List* und *Set* läßt sich leicht mit einer Sortierfunktion lösen, wobei *List* als höherwertiges Konzept in das föderierte Schema übernommen werden sollte. Zwei unterschiedliche Wege zur Auflösung von Konflikten zwischen *List/Set* und *1* gibt es, jedoch tritt bei beiden ein semantischer Verlust ein. Wenn eine mehrwertige Kardinalität in das föderierte Schema übernommen wird, können nicht alle globalen Updates gespeichert werden. Wird *1* übernommen, können nicht die gesamten Beziehungen dem FDBS-Nutzer sichtbar gemacht werden.

Als Ausweg für die obengenannten Probleme bleibt die direkte Übernahme der Beziehungen aus den Exportschemata, wobei dies den Integrationsforderungen widerspricht. Technisch kann die Integration von Beziehungen sowohl über Ableitungsmechanismen als auch über eine explizite Integrationsoperation erfolgen.

6.2.3.4 Integration von Schlüsselbedingungen

Der Hauptgrundsatz bei der Integration von Schlüsselbedingungen ist, daß keine Aufweichung der Schlüsselbedingungen der zugrundeliegenden lokalen Schemata erfolgen darf, damit Änderungen durch FDBS-Nutzer korrekt in den Datenbanken der CDBSe wirksam werden können. Aus dieser Forderung kann direkt abgeleitet werden, daß für alle Objekttypen des föderierten Schemas, die nur Instanzen einer Datenbank enthalten, die Schlüsselbedingungen 1:1 aus den Exportschemata übernommen werden können. Das gleiche gilt für Objekttypen, die globale Instanzen von realweltäquivalenten Objekten besitzen, wenn kein Schlüsselkonflikt zwischen den zugrundeliegenden Objekttypen der Exportschemata existiert. Die Integration von Schlüsselbedingungen für diese Objekttypen bei Existenz eines Schlüsselkonfliktes gilt es nun zu untersuchen, wobei zwei Fälle unterschieden werden.

Im ersten Fall wurden bei der Integration von semantisch vergleichbaren Objekttypen die in Abschnitt 6.2.3.1 gestellten Prämissen eingehalten, so daß der Objekttyp nur globale Instanzen von realweltäquivalenten Objekten enthält. Dann wird die Einhaltung aller Schlüsselbedingungen der zugrundeliegenden Objekttypen gefordert. Dies ist im ODMG-93 Objektmodell leicht zu realisieren, da für einen Objekttyp mehrere Schlüsselbedingungen definiert werden können.

Im zweiten Fall, wenn der Objekttyp nicht nur Instanzen von realweltäquivalenten Objekten besitzt, kann die Einhaltung aller Schlüsselbedingungen zu einer Verschärfung der lokalen Schlüsselbedingungen führen. Dies tritt immer dann ein, wenn eine lokale Schlüsselbedingung schärfer als andere lokale Schlüsselbedingungen semantisch vergleichbarer Objekttypen ist. Wie verschärfte globale Schlüsselbedingungen realisiert werden können, kann [Gro95] entnommen werden. Ebenfalls können hier Probleme durch Nullwerte auftreten, wenn eine Schlüsseleigenschaft als unabhängige Eigenschaft eines zugrundeliegenden Objekttyps integriert wurde. In diesem Fall sollten bei der Integration die Prämissen eingehalten werden.

6.2.3.5 Auflösung von Attribut/Metainfo-Konflikten

Bei der Integration von semantisch vergleichbaren Objekttypen gilt es, Konflikte des Typs Attribut versus Metainfo aufzulösen. Die entscheidende Frage hierbei ist, ob ein Realweltkonzept durch ein Attribut oder durch einen Objekttyp dargestellt werden soll. Durch die geforderte Angleichung bei Strukturkonflikten und die bei der Integration gestellten Prämissen ergibt es sich, daß alle Realweltkonzepte, die mindestens in einem Exportschema als Objekttyp abgebildet werden, dies auch im föderierten Schema tun. Damit ist ein Diskriminantenattribut

im föderierten Schema nicht notwendig, da die enthaltene Information über Namen von Objekttypen ausgedrückt wird. Beim Einfügen neuer Objekte durch FDBS-Nutzer muß das Diskriminantenattribut lokal auf den dem Objekttyp entsprechenden Wert gesetzt werden. Dies kann durch Implementierung in der *create* Operation des Objekttyps geregelt werden.

6.2.4 Bestehende Ansätze zur Schemaintegration mit dem ODMG-93 Objektmodell

Zum Abschluß der Betrachtungen zur Schemaintegration mit dem ODMG-93 Objektmodell sollen zwei Ansätze beschrieben und diskutiert werden, bei denen der Aufbau eines FDBSs mit dem ODMG-93 Objektmodell als GDM versucht wurde. Hierbei handelt es sich um das IRO-DB Projekt [BFN94] und das Projekt Efendi [Rad94]. In beiden Ansätzen wurde eine Erweiterung des ODMG-93 Standards vorgenommen. Es soll untersucht werden, wie die Schemaintegration in beiden Ansätzen durchgeführt wird.

Beim Projekt Efendi wird die ODL erweitert, um eine Schemaintegration zu ermöglichen. Es wird das Schlüsselwort *Schema* eingeführt, das die Zusammenfassung von Interfacebeschreibungen und Implementationen zu einem Schema ermöglicht. Zur Unterstützung der Ableitung von Schemata können zwischen Schemata Vererbungsbeziehungen aufgebaut werden, d. h. bei der Schemaintegration "erbt" das föderierte Schema alle Definitionen der Exportschemata. Mit einer Operation *rename* können ererbte Schemaelemente umbenannt werden, um Namenskonflikte aufzulösen. Dieser Ansatz erlaubt damit keine Schemaintegration im geforderten Sinne sondern eher eine Vereinigung der Exportschemata.

Beim IRO-DB Projekt erfolgt die Schemaintegration über sogenannte virtuelle Klassen. Dabei handelt es sich um Interfacebeschreibungen gemäß der ODL, die über Mappingspezifikationen aus den Exportschemata abgeleitet werden. Diese Mappingspezifikationen sind Erweiterungen der ODL, die mit OQL-Anfragen verbunden werden. Damit wird ein Ableitungskonzept definiert, das weitgehend die geforderte Funktionalität erfüllt. So ist die Auflösung der meisten Konflikte möglich. Die Attributintegration wird parallel auf zwei Arten gelöst. Zum einen werden für jedes Attribut einer virtuellen Klasse die Operationen *get_value* und *set_value* mit den notwendigen Zugriffsinformationen und Transformationsfunktionen implementiert. Zum anderen wird in den Mappingspezifikationen für jedes Attribut der Zugriff durch eine OQL Anfrage definiert. Transformationsfunktionen werden dabei durch Funktionsaufrufe realisiert. So wird bei der Schemaintegration ein globales Data Dictionary mit allen notwendigen Informationen erstellt. Die technische Realisierung einer Schemaintegration mit dem ODMG-93 Objektmodell ist durch diesen Ansatz gegeben.

Zusammenfassend läßt sich erkennen, daß der Ansatz des IRO-DB Projekts zur Realisierung einer Schemaintegration mit dem ODMG-93 Objektmodell besser geeignet ist als der des Projektes Efendi. Beide können jedoch die grundsätzlichen Probleme, die z. B. bei der Attribut- und Beziehungsintegration entstehen, nicht lösen.

6.3 Ableitung externer Schemata

Die Ableitung externer Schemata hat den Zweck, die durch die Schemaintegration gewonnene einheitliche Betrachtungsweise auf die Daten der CDBSe an die Wünsche, Bedürfnisse und Rechte der FDBS-Nutzer anzupassen. Basierend auf den gestellten Anforderungen ist es notwendig zu untersuchen, inwieweit das ODMG-93 Objektmodell hierfür Unterstützung bereitstellt.

Zuerst werden externe Schemata ausgedrückt im ODMG-93 Objektmodell betrachtet. Ihre Ableitung erfolgt durch Mechanismen gemäß Charakteristik 7 von [SCG91]. Im Abschnitt zur

Schemaintegration wurde die fehlende Unterstützung des ODMG-93 Objektmodells für ein Sichtenkonzept diskutiert. Entscheidende Verbesserungen sind hier erst durch weitere Versionen des Standards zu erwarten. Die im IRO-DB Projekt vorgenommene Erweiterung des Standards bietet aber bereits einen Ansatz für ein Sichtenkonzept unter Ausnutzung der Anfragesprache OQL. Es bleibt zu untersuchen, ob dieser Ansatz die gestellten Anforderungen erfüllt, d. h. ob sich jede Untermenge des föderierten Schemas ezeugen läßt.

Desweiteren soll das ODMG-93 Objektmodell auf Erfüllung der zusätzlichen Forderungen untersucht werden. In Abschnitt 6.1.2 wurden die Möglichkeiten für die Definition von Integritätsbedingungen im ODMG-93 Objektmodell betrachtet. Ein geeignetes Mittel zur Definition zusätzlicher Integritätsbedingungen ist die Erweiterung der Funktionalität von abgeleiteten Objekttypen. Dies kann durch Hinzufügen neuer oder durch Überladen bestehender Operationen geschehen. Ein großer Nachteil entsteht durch die fehlende Möglichkeit, Integritätsbedingungen deklarativ zu formulieren. Die Gewährleistung globaler Integritätsbedingungen in einem FDBS ist aktueller Forschungsgegenstand (siehe [Gro95]). Das ODMG-93 Objektmodell verfügt in der aktuellen Version über keine vordefinierten Mechanismen zur Zugriffskontrolle. Hier muß auf die Sicherungsmechanismen des jeweiligen ODBMSs, das das ODMG-93 Objektmodell unterstützt, zurückgegriffen werden.

Ebenfalls soll die Möglichkeit untersucht werden, externe Schemata in andere Datenmodelle abzuleiten. Dazu muß eine Schematransformation vom ODMG-93 Objektmodell in das gewünschte Datenmodell vorgenommen werden. Diese verläuft unter umgekehrten Vorzeichen als die in Abschnitt 6.1 beschriebene Transformation der Exportschemata in die Komponentenschemata, da das ODMG-93 Objektmodell hier als Quelldatenmodell dient. Verfügt das Zieldatenmodell über geringere Ausdrucksfähigkeiten als das ODMG-93 Objektmodell, so ist die Übertragung bestimmter Konzepte nur eingeschränkt möglich. So können z. B. Generalisierungs-/Spezialisierungshierarchien im Relationenmodell nur unzureichend ausgedrückt werden. Da es sich bei der Übertragung objektorientierter Konzepte in einfachere Datenmodelle um einen aktuellen Forschungsgegenstand handelt (vgl. [HK95], [Sch95]), soll an dieser Stelle keine weitere Untersuchung erfolgen.

Eine weitere Problematik kann sich aus der Forderung ergeben, ein externes Schema so abzuleiten, daß es die Schemainformationen eines Exportschemas ausgedrückt im Datenmodell des zugrundeliegenden Komponentenschemas enthält. Dazu ist neben einer Schematransformation eine Filterung der ursprünglichen Schemainformationen des Exportschemas notwendig. Dabei müssen bei der Schemaintegration zerlegte Objekttypen zusammengefaßt werden. Bei der Attribut- und Beziehungsintegration eingegangene Kompromisse lassen sich dabei nur schwer rückgängig machen.

6.4 Datenintegration

Mit der Transformation und Integration der Daten zur Repräsentation für den FDBS-Nutzer an der globalen Schnittstelle findet die Integration heterogener, autonomer Datenbanksysteme ihre Vollendung. Verschiedene Voraussetzungen sind notwendig, um diesen Prozeß zu realisieren. In Abschnitt 5.4 wurden dabei zwei Aspekte, die von besonderer Bedeutung sind, hervorgehoben und entsprechende Anforderungen dafür an das GDM definiert. Bevor diese in den folgenden Unterabschnitten auf Erfüllung durch das ODMG-93 Objektmodell untersucht werden, sollen einige weitere Aspekte kurz angesprochen werden.

Im Gegensatz zu den anderen Integrationsprozessen und der Vordatenintegration erfolgt der Prozeß der laufenden Datenintegration während des Betriebes des FDBSs. Für diesen Prozeß werden die folgenden Fähigkeiten des GDMs benötigt:

- eine deklarative, vollständige und abgeschlossene Anfragesprache zur Formulierung von Anfragen an das föderierte Schema oder externe Schemata,
- eine Datenmanipulationssprache zur Ermöglichung des globalen Einfügens, Änderns und Löschens von Daten.

Mit der OQL und der OML des ODMG-93 Standards sind die Voraussetzungen für globale Anfragen und Datenmanipulationen gegeben. Eine Analyse der beiden Sprachen, ob und wie sie genau diese Anforderungen erfüllen, ist nicht Gegenstand dieser Arbeit.

Auch für die Realisierung der laufenden Datenintegration ist eine besondere Unterstützung durch das GDM notwendig Es wird ein Transaktionsmodell benötigt, das die Umsetzung globaler Anfragen und Änderungen in lokale Anfragen und Änderungen an die CDBSe bei weitgehender Einhaltung der ACID-Eigenschaften erlaubt. In einem FDBS müssen globale Transaktionen in Teiltransaktionen zerlegt werden, die jeweils lokal in einem CDBS ausgeführt werden. Bei vollständiger Autonomie der CDBSe können die ACID-Eigenschaften für diese Teiltransaktionen nur schwer eingehalten werden. So ist z. B. zur Sicherung der Eigenschaften ein globales Commit-Protokoll notwendig, das jedoch in die Kommunikations- und Ausführungsautonomie der CDBSe eingreift [Rah94]. Deshalb wird unter anderem für ein FDBS ein offen geschachteltes Transaktionsmodell empfohlen [BGS95]. Wie in Abschnitt 2.2 berichtet, unterstützt das ODMG-93 Objektmodell ein offen geschachteltes Transaktionsmodell, so daß in diesem Bereich ein Ansatz für die Realisierung eines Transaktionsmanagements eines FDBSs gegeben ist. Ausführlichere Untersuchungen zu den Problemen der Transaktionsverwaltung in föderierten Systemen waren aus Zeitgründen nicht möglich. Eine Betrachtung verschiedener Ansätze dazu bieten [VG93].

6.4.1 Identifizierung von realweltäquivalenten Objekten

Die Implementierung eines Mechanismus zur Identifizierung von Proxy-Objekten desselben Realweltobjekts in einem FDBS ist eine wichtige Voraussetzung für eine erfolgreiche Datenintegration. In Abschnitt 5.4.1 wurden einige Vorschläge für diesbezügliche Techniken diskutiert. Ein Ansatz, der vorgestellt und als einsatzfähig charakterisiert wurde, ist der Mechanismus von [LSPR93], der für das relationale Modell entworfen wurde. Möglichkeiten für seine Umsetzung mit dem ODMG-93 Objektmodell sollen hier analysiert werden.

Eine Proxy-Objektidentifizierung muß jeweils für alle Instanzen von semantisch vergleichbaren Schemaelementen unterschiedlicher Exportschemata, die im föderierten Schema durch einen gemeinsamen Objekttyp vertreten sind, durchgeführt werden. Grundlage des Ansatzes von [LSPR93] ist der paarweise Vergleich der Instanzen dieser Schemaelemente anhand einer erweiterten Schlüsselbedingung, die sich aus der Vereinigung der zugrundeliegenden Schlüsselbedingungen ergibt. Weisen nun zwei Objekte unterschiedlicher Datenbanken gleiche Werte für die erweiterte Schlüsselbedingung auf, so werden sie als Proxy-Objekte identifiziert. Dieser Prozeß läßt sich so nicht ohne weiteres für alle in Frage kommenden Schemaelemente realisieren.

Als erstes ergibt sich für das ODMG-93 Objektmodell die Frage, wie die Schlüsselbedingungen von bei der Auflösung von Strukturkonflikten objektifizierten Schemaelementen aussehen. Hier läßt sich die Kombination aller Eigenschaften dieser Schemaelemente als Schlüsselbedingung ansehen, was jedoch zu Problemen mit existierenden Nullwerten führen kann. Abhilfe kann hier der Erwerb zusätzlicher Kenntnisse über die Semantik der Exportschemata schaffen, um die Schlüsselbedingung auf weniger Eigenschaften einzuschränken.

Das zweite Problem besteht dann, wenn nicht in jedem Schemaelement alle Eigenschaften der erweiterten Schlüsselbedingung vorhanden sind. Dann müssen die in Abschnitt 5.4.1

beschriebenen ILFDs auf der Grundlage von erworbenem Wissen implementiert werden. Da für bestimmte Paare von Schemaelementen ILFDs für sehr viele Wertepaare zu erwarten sind, wird von [LSPR93] vorgeschlagen, diese als Datenstruktur im globalen Data Dictionary abzuspeichern. Im ODMG-93 Standard gibt es zur Zeit keine exakte Definitionsgrundlage für den Aufbau eines Data Dictionary. Damit ist man auch hier wie bei der Realisierung der Schemaintegration gezwungen, den Standard eigenmächtig zu interpretieren oder zu erweitern.

Das dritte zu lösende Problem sind Objektidentifizierungskonflikte, die durch Konflikte zwischen Eigenschaftswerten hervorgerufen werden. Sofern diese Konflikte auf inkorrekten Eingaben beruhen, können sie nur manuell durch den DBA des FDBS gelöst werden. Bei unterschiedlichen Repräsentationen könnte eine Vereinheitlichung durch Verhandlungen zwischen den einzelnen CDBSen erfolgen und so das Auftreten von Konflikten verhindert werden. Wie Konflikte, die durch unterschiedliche Änderungszeitpunkte an Proxy-Objekten desselben Realweltobjekts entstehen, lösbar sind, wird unter anderem im nächsten Abschnitt angesprochen.

6.4.2 Behandlung der Objektidentität

Die Datenintegration ist ein aufwendiger Prozeß, so daß ein effizienter und schneller globaler Zugriff auf die Daten nur schwer zu realisieren ist. Deshalb wurde die Forderung nach unveränderlicher Objektidentität für alle globalen Objekte erhoben, um eine schnelle und sichere Objektidentifizierung innerhalb der gesamten föderierten Datenbank zu ermöglichen. In diesem Abschnitt soll daher untersucht werden, ob und wie diese Forderung mit dem ODMG-93 Objektmodell zu erfüllen ist.

Das ODMG-93 Objektmodell stattet, wie in Abschnitt 2.2 beschrieben, alle Instanzen von Objekttypen mit einer auf Lebenszeit festgelegten Identität aus, die durch eine OID repräsentiert wird. Damit ist eine wichtige Voraussetzung für die Erfüllung der Forderung vorhanden, da jedes globale Objekt mit einer eigenen Identität versehen werden kann. Eine weitere Voraussetzung, die erfüllt werden muß, ist, eine Abbildung zwischen dem globalen Objekt und den zugrundeliegenden lokalen Objekten zu realisieren.

Da die globalen Objekte nur aus lokalen Objekten abgeleitet werden und damit nur virtuell sind, entsteht die Frage, wo und wie die Identitätsinformationen gespeichert werden sollen. Sollen diese Informationen global gespeichert werden, so muß eine globale Datenbank angelegt werden. Dies hat den Vorteil, daß durch eine globale Anfrageoptimierung der Zugriff auf die Daten schneller und effizienter gestaltet werden kann. Demgegenüber steht der zu erwartende Aufwand für das Anlegen und Warten der Datenbank. Vor allem die in Abschnitt 5.4.2 beschriebenen Verfahren zur Überwachung lokaler Objektidentifizierungsfunktionen sind mit ständigem Recheneinsatz verbunden und bedeuten einen Eingriff in die Autonomie der CDBSe. Bei lokaler Speicherung der Identitätsinformationen wie von [SS95] vorgeschlagen entfällt dieser Aufwand, jedoch ergeben sich andere Probleme. So muß die OID für jedes globale Objekt im Format des jeweiligen DBMSs gespeichert werden, wobei sich Ansatzpunkte für nicht erlaubte Manipulationen ergeben. Weitere Probleme ergeben sich hinsichtlich der globalen Repräsentation von realweltäquivalenten Objekten, d. h. hier sind systemübergreifende Mechanismen notwendig.

Für die Sicherung unveränderlicher Objektidentität ist es zwingend notwendig, Informationen über realweltäquivalente Objekte global zu speichern. Die angesprochenen Probleme mit dem Aufbau eines Data Dictionary und den Überwachungsverfahren gelten auch hier. Die Überwachungsverfahren können Attributwertkonflikte, die durch unterschiedliche Änderungszeitpunkte in den CDBSen entstehen, aufdecken. Durch Kooperation zwischen globalem DBA und lokalen DBAen kann eine Lösung herbeigeführt werden. Die

unveränderliche Objektidentität kann bei Änderungen in der *same*-Relation nicht immer aufrechterhalten werden. Ergibt sich, daß zu einem vorhandenen Objekt einer Datenbank ein realweltäquivalentes Objekt in einer anderen Datenbank eingefügt wird, so muß die bereits vorhandene globale Repräsentation durch eine neue ersetzt werden, die dem Objekttyp für die realweltäquivalenten Objekte zugeordnet wird. Genau umgekehrt verhält es sich, wenn eines der realweltäquivalenten Objekte lokal gelöscht wird. Die unveränderliche Objektidentität könnte in diesen Fällen gesichert werden, wenn Objekte während ihrer Lebenszeit den Objekttyp in einer Hierarchie wechseln könnten. Dieses Konzept wird zur Zeit aber vom ODMG-93 Objektmodell nicht unterstützt.

Abschließend soll die Realisierung der Datenintegration und der damit verbundenen Behandlung der Objektidentität im schon erwähnten IRO-DB Projekt angesprochen werden. Die Forderung nach unveränderlicher Objektidentität wird nicht erhoben, vielmehr wird ein *session oid* Konzept verwirklicht. Globale Objekte werden dabei nur bei Zugriff durch eine globale Transaktion erzeugt und durch diese auch wieder vernichtet, d. h. nur während der Dauer dieser Transaktion bleibt die Objektidentität konstant.

6.5 Eignungsbewertung

Die Untersuchung des ODMG-93 Objektmodells bezüglich der Eignung als GDM zur Integration heterogener, autonomer Datenbanksysteme brachte interessante Ergebnisse. Die folgende Tabelle listet die Ergebnisse auf der Basis der in Abschnitt 5.5 gestellten Anforderungen auf.

Integrationsprozeß	Ergebnisse der Eignungsuntersuchung
Schematransformation	1. Das ODMG-93 Objektmodell besitzt vielfältige Ausdrucksfähigkeiten, die die Schematransformation aus anderen Datenmodellen ermöglichen und semantische Anreicherung gestatten.
Schemaintegration	2. Erkenntnisse zur Schemaintegration im Anschluß 3. Das ODMG-93 Objektmodell bietet wenig Unterstützung für Integrationsoperationen. Ableitungsmechanismen werden nicht unterstützt.
Ableitung externer Schemata	4. Das ODMG-93 Objektmodell enthält kein Sichtenkonzept. Die prozedurale Definition von Integritätsbedingungen ist möglich. Maßnahmen zur Zugriffskontrolle werden nicht unterstützt. 5. Die Ableitung in andere Datenmodelle ist möglich, Probleme existieren bei der Übertragung bestimmter Konzepte in einfachere Datenmodelle.
Datenintegration	6. Ein formales Metamodell zum Aufbau eines Data Dictionary fehlt. 7. Ein Mechanismus zur Sicherung unveränderlicher Objektidentität läßt sich mit sehr viel Aufwand und Eingriffen in die Autonomie der CDBSe realisieren. Probleme ergeben sich bei der Behandlung realweltäquivalenter Objekte.

Tabelle 6.3: Ergebnisse der Eignungsuntersuchung

Besonders komplex und widersprüchlich sind die Ergebnisse, die bei der Untersuchung der Schemaintegration erzielt wurden. Das Konzept der Generalisierung/Spezialisierung im ODMG-93 Objektmodell bietet die Möglichkeit, semantisch vergleichbare Objekttypen mit optimaler Konfliktauflösung, d. h. bei Erfüllung der gestellten Prämissen, zu integrieren. Gleichzeitig treten dabei Probleme bei der Integration von Attributen und Beziehungen auf. Weiterhin läßt sich feststellen, daß die Erfüllung der Prämissen den Forderungen nach Minimalität und Verständlichkeit zuwiderläuft, da durch die extensionale Zerlegung die Fragmentierung gefördert wird.

Das ODMG-93 Objektmodell eignet sich bedingt als GDM eines FDBSs zur Integration heterogener, autonomer Datenbanksysteme. Die objektorientierten Konzepte bieten gute Voraussetzungen zur Realisierung von Schematransformation und -integration als den Kernprozessen der Integration. Fehlende, aber in späteren Versionen zu erwartende Funktionalität, erlaubt zur Zeit eine praktische Umsetzung nur auf der Basis eigenständiger Standarderweiterungen. Praktische Ansätze wie das IRO-DB Projekt zeigen bei abgeschwächten Anforderungen einen begehbaren Weg zur Realisierung eines FDBSs auf der Basis des ODMG-93 Objektmodells.

7 Schlußbetrachtung und Ausblick

In dieser Arbeit sollte die Eignung des ODMG-93 Objektmodells als Objektmodell einer Föderierungsschicht zur Integration heterogener, autonomer Datenbanksysteme untersucht werden. Die folgenden drei Etappen kennzeichneten dabei den Verlauf der Arbeit:

- Einarbeitung in den ODMG-93 Standard und das Gebiet der föderierten Datenbanksysteme,
- Ausarbeitung von Anforderungen an ein gemeinsames Datenmodell eines FDBS bezüglich der Integration der CDBSe,
- Untersuchung des ODMG-93 Objektmodells hinsichtlich der gestellten Anforderungen, sowie der Umsetzbarkeit von Ansätzen zur Realisierung der Integration.

An dieser Stelle sollen die Ergebnisse anhand der Etappen zusammengefaßt und aufgearbeitet werden.

Die Einarbeitung in den ODMG-93 Standard für objektorientierte Datenbankmanagementsysteme erfolgte auf der Grundlage des Buches von [Cat94]. Dabei zeigte sich, daß die Entwicklung offenbar unter Zeitdruck und mit ungenauer Abstimmung erfolgte, da sich inhaltliche Widersprüche zwischen den einzelnen Teilen des Standards ergeben, so wird z. B. die Aggregation von Objekten im Objektmodell und in der ODL unterschiedlich gehandhabt. Hier macht sich das Fehlen einer formalen Definition bemerkbar. Die Funktionalität des Standards stellt einen Kompromiß zwischen den Herstellern objektorientierter Datenbanksysteme dar. Eine große Nähe besteht zum Objektmodell der Programmiersprache C++. Da es sich um die erste Version des Standards handelt, sind eine Reihe notwendiger Konzepte wie ein Sichtenkonzept und ein Metamodell noch nicht definiert wurden. Hier muß die weitere Entwicklung abgewartet werden.

Das Gebiet der föderierten Datenbanksysteme ist ein relativ junger Forschungsgegenstand, der sich in ständiger Weiterentwicklung befindet. Basis für die Einarbeitung in diesen Bereich war die Arbeit von [SL90], in der wichtige Grundlagen geklärt und ein Referenzmodell abgeleitet wird. Es konnte jedoch festgestellt werden, daß weiterhin unterschiedliche Begriffe und Betrachtungsweisen verwendet werden. Deshalb mußte für diese Arbeit eine einheitliche Begriffsgrundlage geschaffen und verwendet werden.

Durch Literaturstudium wurde erkannt, wie komplex das Problem der Integration heterogener, autonomer Datenbanksysteme in ein föderiertes Datenbanksystem ist. Bevor deshalb Anforderungen an ein gemeinsames Datenmodell erarbeitet werden konnten, erwies es sich als notwendig, dieses Problem in Form einer Klassifikation der auftretenden Konflikte aufzuarbeiten. Dabei wurde ein enger Zusammenhang zwischen den Konflikten und dem jeweils verwendeten gemeinsamen Datenmodell festgestellt. Dabei konnten bezüglich des ODMG-93 Objektmodells sowohl Vor- als auch Nachteile nachgewiesen werden.

Wegen der nötigen und aufwendigen Vorbetrachtungen mußte bei der Ausarbeitung der Anforderungen eine Beschränkung vorgenommen werden. Orientiert am Referenzmodell von [SL90] wurden vier Prozesse ausgewählt, die für die Integration wichtig sind und besonderer Unterstützung durch die Fähigkeiten des gemeinsamen Datenmodells bedürfen. Aus der Menge der vorliegenden Literatur wurden Realisierungsansätze für diese Prozesse gewählt und diskutiert und die Anforderungen abgeleitet. Dabei hat sich herausgestellt, daß oftmals gegensätzliche Positionen zu einem Thema bezogen werden. So werden zum Beispiel unterschiedliche Ansätze hinsichtlich der Wahl des gemeinsamen Datenmodells oder bezüglich der Behandlung der Objektidentität in einem FDBS verfolgt. Die Ableitung der Anforderungen

erfolgte darum immer unter dem Gesichtspunkt, daß das ODMG-93 Objektmodell als gemeinsames Datenmodell untersucht werden soll.

Die Ergebnisse der Untersuchung und die Eignungsbewertung des ODMG-93 Objektmodells wurden in Abschnitt 6.5 vorgestellt. Die Eignungsbewertung konnte ausschließlich auf der Basis der im Verlauf der Untersuchung gesammelten Informationen erfolgen. Aus Zeitgründen konnte kein Vergleich mit Ansätzen, die andere Datenmodelle als GDM nutzen, stattfinden, so daß mögliche Alternativen in die Bewertung nicht einfließen können. Die Untersuchung fand nur auf theoretischer Basis statt, d. h. eine Implementierung des ODMG-93 Standards in einem ODBMS wurde nicht zur Untersuchung herangezogen.

Abschließend soll ein Ausblick auf die weitere Arbeit in diesem Bereich folgen. Zum ersten sind grundsätzliche Untersuchungen der Integrationsprozesse notwendig, um wichtige Probleme der Integration bereits im Ansatz zu klären. Zum zweiten sollten die in dieser Arbeit erwähnten praktischen Realisierungen von FDBSen auf der Basis des ODMG-93 Objektmodells durch Untersuchungen weiterverfolgt werden. Eine eigenständige praktische Realisierung am Institut für Technische Informationssysteme ist zur Zeit nicht zu empfehlen. Hier sollte die Akzeptanz des Standards sowie weitere Versionen abgewartet werden. Zum dritten müssen die weiteren Bestandteile des Standards, vor allem OQL und OML, auf ihre Eignung zur effizienten Nutzung in einem FDBS untersucht werden.

Quellenverzeichnis

[Bee93] Beeri C.: *Some Thoughts on the Future Evolution for Object-Oriented Databases Concepts.* In: Stucky W., Oberweis A., editors: Proceedings GI-Fachtagung Datenbanksysteme in Büro, Technik und Wissenschaft (BTW93). Informatik aktuell, 18-32, Springer-Verlag, Braunschweig, März 1993.

[BFN94] Busse R., Fankhauser P., Neuhold E.J.: *Federated Schema in ODMG.* In: Eder J., Kalinichenko L. A., editors: Extending Information Systems Technology - Proc. of the 2nd Int. East-West Database Workshop, Klagenfurt, Austria. 356-379, Springer-Verlag, Workshops in Computing, Sept. 1994.

[BGS95] Breitbart Y., Garcia-Molina H., Silberschatz A.: *Transaction Management in Multidatabase Systems.* In: Kim, W., editor: Modern Database Systems. 573-591, ACM Press, 1995.

[BLN86] Batini E., Lenzerini M., Navathe S.: *A Comparative Analysis of Methodologies for Database Schema Integration.* ACM Computing Surveys, Vol. 18, No. 4, 323-364, 1986.

[Bra93] Bratsberg S.E.: *Integrating Independently Developed Classes.* In: Özsu M., Dayal U., Valduriez P., editors: Distributed Object Management. 329-333, Morgan Kauffmann Publishers, San Matteo, California, 1993.

[Cat94] Cattell R.G.G., editor: *The Object Database Standard: ODMG-93.* Morgan Kaufmann Publishers, San Matteo, California, 1994.

[CL93] Chomicki J., Litwin W.: *Declarative Definition of Object-Oriented Multidatabase Mappings.* In: Özsu M., Dayal U., Valduriez P., editors: Distributed Object Management. 375-392, Morgan Kauffmann Publishers, San Matteo, California, 1993.

[CS91] Chatterjee A., Segev A.: *Data Manipulation in Heterogenous Databases.* ACM SIGMOD Record, Vol. 20, No 4, 64-68, Dec. 1991.

[CSG93] Castellanos M., Saltor F., Garcia-Solaco M.: *A Canonical Model for the Interoperability Among Object-Oriented and Relational Databases.* In: Özsu M., Dayal U., Valduriez P., editors: Distributed Object Management. 309-314, Morgan Kauffmann Publishers, San Matteo, California, 1993.

[Dup94] Dupont Y.: *Resolving Fragmentation Conflicts in Schema Integration.* In: Loucopoulos, editor: Proc. 13th Intern. Conf. on the Entity-Relationship Approach (ER´94), Manchaster UK. 513-532, Springer-Verlag, LNCS 881, Dec. 1994.

[EK91] Eliassen F., Karlsen R.: *Interoperability and Object Identity.* ACM SIGMOD Record, Vol. 20, No 4, 25-29, Dec. 1991.

[GB91] Gangopadhyay D., Barsalou T.: *On the Semantic Equivalence of Heterogenous Representations in Multimodel Multidatabase Systems.* ACM SIGMOD Record, Vol. 20, No 4, 35-39, Dec. 1991.

[Gro95] Grohmann G.: *Grundlagen der Konsistenzsicherung in heterogenen, föderierten Datenbankumgebungen.* Diplomarbeit, Otto-von-Guericke-Universität Magdeburg, Institut für Technische Informationssysteme, Sept. 1995.

[HD93] Härtig M., Dittrich K.: *Objektidentifikation in Heterogenen Datenbanksystemen.* In: Stucky W., Oberweis A., editors: Proceedings GI-Fachtagung Datenbanksysteme in Büro, Technik und Wissenschaft (BTW93). Informatik aktuell, Springer-Verlag, Braunschweig, März 1993.

[HK95] Hohenstein U., Körner C.: *Semantische Anreicherung relationaler Datenbanken.* In: Lausen G., editor: Proc. GI-Fachtagung Datenbanksysteme in Büro, Technik und Wissenschaft (BTW95), Dresden. Informatik aktuell, Springer-Verlag, Mai 1995.

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

[KPP+94] Kappel G., Preishuber S., Pröll E., Rausch-Scott S., Retschitzegger W., Wagner R., Gierlinger C.: *COMan - Coexistence of Object-Oriented and Relational Technology.* In: Loucopoulos, editor: Proc. 13th Intern. Conf. on the Entity-Relationship Approach (ER´94), Manchaster UK. Springer-Verlag, LNCS 881, Dec. 1994.

[KS91] Kim W., Seo J.: *Classiffying Schematic and Data Heterogeneity in Multidatabase Systems.* IEEE Computer 24 (12), 12-18, Dec. 1991.

[LSPR93] Lim E., Srivastava J., Prabhakar S., Richardson J.: *Entity Identification in Database Integration.* In: Proc. 9th Intern. Conf. on Data Engineering. 294-301, Vienna, Austria, April 1994.

[McL93] McLeod D.: *Beyond Object Databases.* In: Stucky W., Oberweis A., editors: Proceedings GI-Fachtagung Datenbanksysteme in Büro, Technik und Wissenschaft (BTW93). Informatik aktuell, 1-17, Springer, Braunschweig, März 1993.

[Rad94] Radeke E.: *Efendi: Federated Database System of Cadlab.* Technical Report, University of Paderborn & Siemens Nixdorf, 1994.

[Rah94] Rahm E.: *Mehrrechner-Datenbanksysteme.* Addison-Wesley, Bonn, 1994.

[RS94a] Radeke E., Scholl M.H.: *Federation and Stepwise Reduction of Database Systems.* In: Proc. of the 1st Int. Conf. on Applications of Databases. Sweden, 1994.

[RS94b] Radeke E., Scholl M.H.: *Object Migration in Federated Database Systems.* In: Cadlab-Report 3/94. Paderborn, 1994.

[SCG91] Saltor F., Castellanos M., Garcia-Solaco M.: *Suitability of Data Models as Canonical Models for Federated Databases.* ACM SIGMOD Record, Vol. 20, No 4, 44-48, Dec. 1991.

[Sch94] Schmitt I.: *Die mehrfache dynamische Klassenzugehörigkeit von Objekten in Objektorientierten Datenbankmodellen.* In: Conrad S., Löhr P., Saake G., editors: Kurzfassungen des 6. GI-Workshops "Grundlagen von Datenbanken", Bad Helmstedt, 19.-22. September 1994. 122-126, Otto-von-Guericke-Universität Magdeburg, Institut für Technische Informationssysteme, Bericht 94-01, 1994.

[Sch95] Scholz U.: *Untersuchung der Möglichkeiten einer effizienten Datenabbildung einer TROLL-Spezifikation auf das DBMS ORACLE 7 unter Ausnutzung des Triggerkonzepts und PL/SQL.* Diplomarbeit, Otto-von-Guericke-Universität Magdeburg, Institut für Technische Informationssysteme, Sept. 1995.

[SL90] Sheth A.P., Larson J.A.: *Federated Database Systems for Managing Distributed, Heterogenous, and Autonomous Databases.* ACM Computing Surveys, Vol. 22, No 3, 183 - 236, Sept. 1990.

[SP91] Spaccapietra S., Parent C.: *Conflicts and Correspondence Assertions in Interoperable Databases.* ACM SIGMOD Record, Vol. 20, No 4, 49-54, Dec. 1991.

[SP92] Spaccapietra S., Parent C.: *View integration: a step forward in solving structural conflicts.* In: IEEE Transactions on Knowledge and Data Engineering. Oct. 1992.

[SPD92] Spaccapietra S., Parent C., Dupont Y.: *Model Independent Assertions for Integration of Heterogenous Schemas.* Very Large Database Journal, Vol 1, No. 1, 1992.

[SS95] Schmitt I., Saake G.: *Managing Object Identity in Federated Database Systems.* to appear: 14th Int. Conf. on Object-Oriented & Entity-Relationship Modelling (O-O&ER´95), Queensland (Australia). 1995.

[SST93] Scholl M.H., Schek H-J., Tresch M.: *Object Algebra and Views for Multiple-Objectbases.* In: Özsu M., Dayal U., Valduriez P., editors: Distributed Object Management. 353-374, Morgan Kauffmann Publishers, San Matteo, California, 1993.

[TK78] Tsichritzis D., Klug A.: *The ANSI/X3/SPARC DBMS framework.* Inf. Syst. 3, 4, 1978.

[UW91] Urban S.D., Wu J.: *Resolving Semantic Heterogeneity through the explicit Representation of Data Model Semantics.* ACM SIGMOD Record, Vol. 20, No 4, 55-58, Dec. 1991.

[VG93] Vossen G., Gross-Hardt, M.: *Grundlagen der Transaktionsverarbeitung.* Addison-Wesley, Bonn, 1993.

Erklärung

Hiermit erkläre ich, daß ich die vorliegende Arbeit selbständig und nur mit erlaubten Hilfsmitteln angefertigt habe.

Magdeburg, den 4.10.1995 (Eyk Hildebrandt)

Thesen

1. Das ODMG-93 Objektmodell verfügt über die klassischen Abstraktionsprinzipien. Es vereinigt in sich sowohl objektorientierte Konzepte, wie z. B. unveränderliche Objektidentität und die Definition von Verhalten, als auch Datenbankkonzepte, wie z. B. ein Transaktionsmodell.

2. Föderierte Datenbanksysteme bieten eine Möglichkeit zur Integration von heterogenen, autonomen Datenbanksysteme. Eng gekoppelte Systeme erlauben die Überwindung der syntaktischen und semantischen Heterogenität der CDBSe durch Propagierung eines einheitlichen föderierten Schemas auf der Basis eines gemeinsamen Datenmodells.

3. Die bei einer Integration heterogener, autonomer Datenbanksysteme in einem FDBS auftretenden Konflikte sind komplex und vielfältig. Eine genaue Untersuchung und Klassifikation der Konflikte ist notwendig, um Lösungsansätze für die Prozesse zur Realisierung der Integration finden zu können.

4. Kernpunkte der Realisierung einer Integration heterogener, autonomer Datenbanksysteme in einem FDBS sind der Aufbau der Schemaebenenhierarchie und die Datenintegration. Dazu sind umfangreiche Anforderungen an ein gemeinsames Datenmodell zu stellen.

5. Das ODMG-93 Objektmodell kann die Anforderungen an ein gemeinsames Datenmodell nur bedingt erfüllen. Die vorhandenen Konzepte bieten die Möglichkeit, die Mehrzahl der Konflikte aufzulösen und damit ein föderiertes Schema zu erstellen, daß den gestellten Forderungen nahekommt. Eine praktische Realisierung des Aufbaus der Schemaebenenhierarchie ist jedoch nicht möglich. Ansätze zur Unterstützung der Datenintegration sind vorhanden, ein Teil der notwendigen Funktionalität fehlt jedoch.

6. Zur Zeit kann der praktische Einsatz des ODMG-93 Objektmodells in einem FDBS nicht emphohlen werden.