Schemaevolution in Föderierten Datenbanksystemen

Silvia Kolmschlag

C-LAB*
Fürstenallee 11, D-33094 Paderborn
E-Mail: siko@c-lab.de, http://www.c-lab.de/~siko

1 Motivation

Im Gegensatz zu herkömmlichen Datenbanken kann Schemaevolution in einem Föderierten Datenbanksystem (FDBS) auf zwei Schemaebenen ausgelöst werden: auf den Externen bzw. Föderierten Schemata und auf den lokalen Schemata der Komponenten-Datenbanksysteme (KDBS) (siehe Abb. 1). Die Schemaänderungen auf diesen Schemaebenen haben wechselseitige Auswirkungen aufeinander. Diese Auswirkungen müssen bei ihrer Abbildung über mehrere Schemaebenen hinweg betrachtet werden.

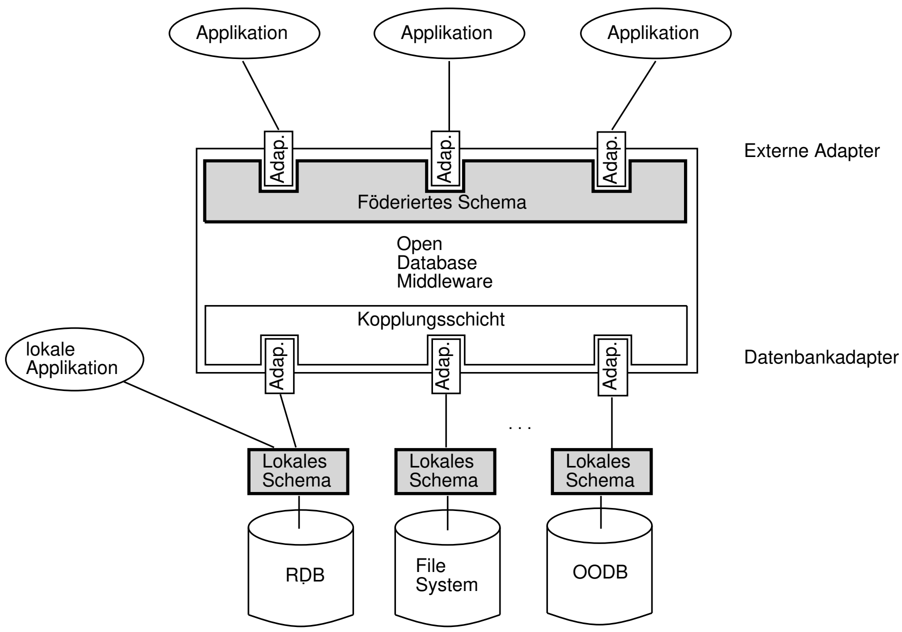

Abbildung 1: Schemaevolution in FDBS

Als Lösungsansatz wird ein Werkzeug für den Administrator eines FDBS vorgestellt, das der Entscheidungsfindung dient, indem verschiedene Alternativen für Abbildungen der veränderten Schemakomponenten auf eine andere Schemaebene vorgestellt werden. Dabei bildet ein Katalog von Standardabbildungen zwischen den Datenmodellen die Informationsquelle bei einer Schemaevolution für den FDBS-Administrator. Er erhält dadurch die Möglichkeit, eine Alternative auszuwählen oder eine eigene Abbildung zu definieren.
Als Voraussetzung zur Realisierung eines solchen Werkzeugs wird gezeigt, daß verschiedene Abbildungen auch zu unterschiedlichen Ergebnissen führen können. Dabei wird Schemaevolution in einem FDBS durch Mechanismen unterstützt, die in beide Richtungen wirken: Abbildungen und ihre Umkehrungen bilden Schemaevolution vom Föderierten auf Lokale Schemata ab und umgekehrt.

*Joint R&D Institute University-GH Paderborn / Siemens Nixdorf Informationssysteme AG.

2 Realisierungsumfeld OpenDM

Das Realisierungsumfeld für die Umsetzung der Konzepte ist der im C-LAB entwickelte Baukasten für Database Middleware: OpenDM (Open Database Middleware) (siehe [OpenDM]). OpenDM ist eine generische und konfigurierbare Softwareschicht oberhalb heterogener Datenbanksysteme (DBS) und koppelt diese mittels generischer Adaptertechnik. Die verschiedenen Systemkonfigurationen von OpenDM lösen die Problemfelder Föderierung von heterogenen DBS, Migration von Daten bzw. Anwendungen und Entwicklung von datenbankunabhängigen Anwendungen.
OpenDM basiert auf einem objekt-orientierten kanonischen Datenmodell und bietet als globale Schnittstelle eine objekt-orientierte Definitions- bzw. Manipulationssprache, die einen einheitlichen, transparenten Zugriff auf die gekoppelten DBS über Föderierte Schemata erlaubt. Weiterhin bietet die globale Schnittstelle Schemaintegrations- und Migrationsbasisfunktionalität.
Der Aufbau der Föderierten Schemata erfolgt gemäß der 5-Ebenen-Schema-Architektur [Sheth et al. 90]. Diese 5 Ebenen beschreiben das Lokale, das Komponenten-, das Export-, das Föderierte und das Externe Schema (siehe Abb. 2). Schemaevolution in diesem Umfeld wird an diesen Schemaebenen reflektiert.

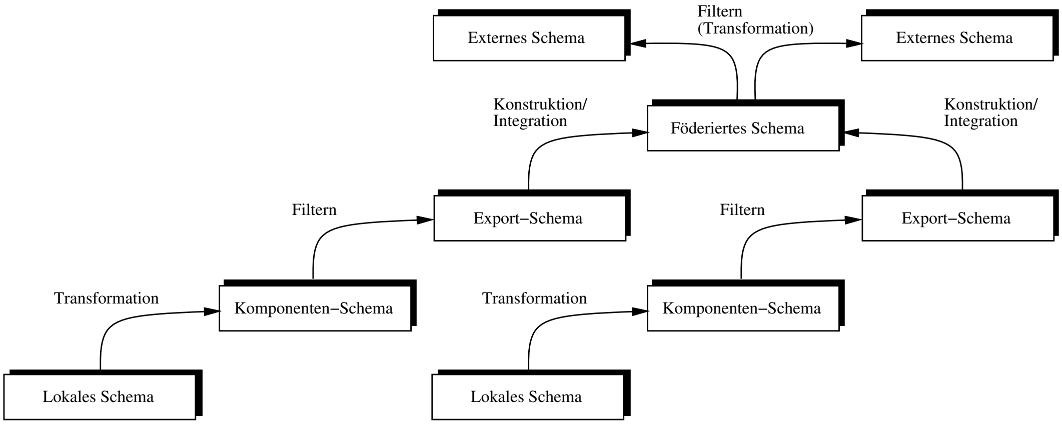

Abbildung 2: 5 Schemaebenen

Schemaintegration wird mittels einer ODL (Object Definition Language) beschrieben, die Mechanismen wie Ausblenden, Umbenennen und Hinzufügen (*drop*, *rename*, *add*) beinhaltet.
Wenn über lokale Schemata zugreifende Anwendungen nach dem Ankoppeln eines DBS an OpenDM problemlos weiterarbeiten sollen, müssen die föderierten DBS autonom bleiben. Hier erlaubt OpenDM die Wahl zwischen verschiedenen Autonomiegraden.

3 Wirkungsrichtungen der Schemaevolution in FDBS

Im Hinblick auf die Realisierungsbasis OpenDM wird der Aufbau der Föderierten Schemata gemäß der 5-Ebenen-Schema-Architektur vorausgesetzt. Demnach kann die Schemaevolution in einem FDBS auf verschiedenen Schemaebenen stattfinden: Zum einen können neue Anforderungen an das Externe bzw. Föderierte Schema gestellt werden. Das kann durch neue oder modifizierte Anwendungen nötig werden. Zum anderen kann eine Schemaevolution für das Lokale Schema einer angekoppelten KDBS erfolgen, die eventuell auf das Externe bzw. Föderierte Schema abgebildet werden muß. Das Externe und das Föderierte Schema wird im folgenden auch als Globales Schema bezeichnet, da auf beiden Schemata globale Applikationen laufen können.
Eine lokale Schemaevolution bewirkt, daß die Änderung *bottom up* vor sich geht. Das bedeutet, daß die Änderungsinformation vom Lokalen Schema des entsprechenden KDBS zum Globalen Schema abgebildet wird, so z.B. bei einer Schemareduktion eines Lokalen Schemas oder bei einer Schemaänderung, die auch global relevant ist. Dafür sind benutzerunterstützende Automatismen zu finden, die diese Abbildung über die verschiedenen Schemaebenen vornehmen, also auch vom Lokalen Schema zum Komponenten-Schema in das kanonische Datenmodell.

Eine neue Anforderung an das Globale Schema löst eine globale Schemaevolution aus. Die Änderungsinformation wird vom Globalen Schema *top down* an die darunterliegenden Schemaebenen weitergereicht. Dabei muß unterschieden werden, ob die Autonomie der KDBS verletzt werden darf oder nicht. Wenn die Autonomie der KDBS bei einer globalen Schemaevolution, die auf lokale Schemata abgebildet werden soll, gewahrt bleiben muß, schlägt die Schemaänderung nur bis auf das Komponenten-Schema, aber nicht bis auf das Lokale Schema durch. Eine ausgezeichnete DB, deren Autonomie verletzt werden darf, muß die Schemaänderungsinformation aufnehmen. Wenn ein Eingriff in die Autonomie des lokalen KDBS erlaubt oder sogar erwünscht ist, bewirkt eine Schemaänderung auf globaler Ebene auch eine Schemaevolution auf lokaler Ebene. Das führt zu Problemen, wenn das lokale Datenmodell nicht mächtig genug ist, um diese Änderung darzustellen. So gibt es im relationalen Datenmodell keine Attribute mit mehreren Werten (multivalued attributes) und keine zusammengesetzten Attribute (composite attributes).
Die Datenbestände des KDBS entsprechen nach einer lokalen Schemaevolution nicht mehr den neuen Strukturen und dem Verhalten, sondern werden teilweise durch die Schemaänderung modifiziert. Um Datenverluste zu vermeiden, müssen die Daten oder die Zugriffsmethoden dem neuen Schema angepasst werden. Dabei können verschiedene Vorgehensweisen realisiert werden, die auf der Grundidee der Migration beruhen.

4 Abbildung der Schemaevolution auf andere Schemaebenen

Gemäß der beiden Wirkungsrichtungen der Schemaevolution in FDBS müssen die Abbildungen der Schemaänderungen auch in beide Wirkungsrichtungen unterstützt werden. Wenn also auf dem Globalen Schema eine Schemaänderung vorgenommen wird, muß diese Änderung auf das lokale Schema der betreffenden KDBS oder der globalen DB, die Daten speichert, die in keiner der angekoppelten KDBS gespeichert werden können, abgebildet werden. Umgekehrt wird eine lokale Schemaänderung auf das Globale Schema abgebildet. Schemaevolution muß also über mehrere Schemaebenen hinweg von einer Ebene zur nächsten abgebildet werden. Abbildung 3 veranschaulicht lokale und globale Schemaänderungen anhand von neuen Beziehungen. Im folgenden werden die Abbildungen zwischen den einzelnen Schemaebenen näher betrachtet.

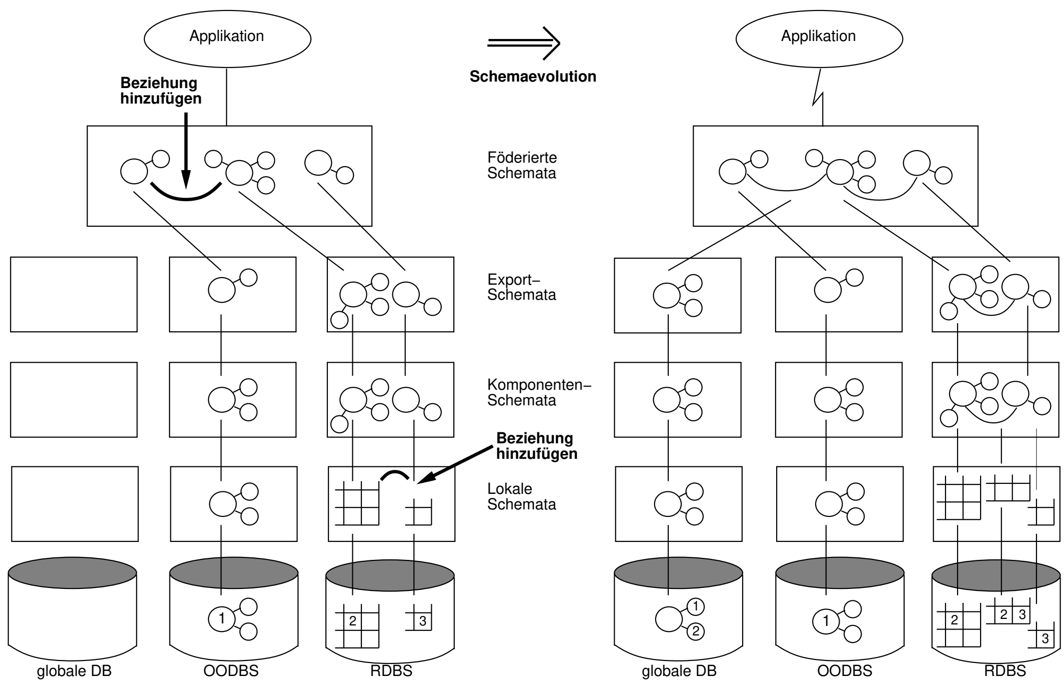

Abbildung 3: Abbildung von Schemaänderungen

Lokales Schema ↔ Komponenten-Schema

Die Abbildung der Schemaänderungen vom Globalen Schema bis auf das Komponenten-Schema und umgekehrt ist im kanonischen Datenmodell beschreibbar. Dagegen geht der Schritt zwischen Komponenten-Schema und Lokalem Schema über Datenmodell-Grenzen hinweg und zwar zu den verschiedenen Datenmodellen der KDBS. Bedingt durch die Heterogenität eines FDBS stehen dem kanonischen Datenmodell eines FDBS die Datenmodelle der angekoppelten KDBS gegenüber. Bei der Integration der Schemata der KDBS werden bereits Abbildungen zwischen den Schemakomponenten der verschiedenen Schemaebenen definiert. Falls sich eine Schemakomponente evolutionär verändert, muß diese Änderung auf die nächste Schemaebene abgebildet werden. Die Schemaänderungen müssen also in andere Datenmodelle transformiert werden. Zusätzliche Schwierigkeiten treten auf, wenn ein KDBS keine Schemaevolution unterstützt, so daß das FDBS die Schemaevolution abfangen oder nachbilden muß.

Komponenten-Schema ↔ Export-Schema

Neben der Überwindung der Grenze zwischen den unterschiedlichen Datenmodellen gibt es weitere Schwierigkeiten bei der Propagierung von Schemaevolution an unterschiedliche Schemaebenen. So findet zwischen dem Komponenten-Schema und dem Export-Schema ein Filtern durch *drop*, *rename* und *add* statt. Dadurch haben einige Schemakomponenten keine Entsprechung in der jeweils anderen Ebene mehr. Schemabestandteile, die bei der Abbildung vom Komponenten-Schema auf das Export-Schema ausgeblendet wurden, müssen bei der Abbildung einer lokalen Schemaevolution auf höhere Schemaebenen nicht berücksichtigt werden. Ebenso bedeutungslos ist eine lokale Schemaevolution eines KDBS für eine Schemakomponente, die bei diesem Übergang hinzugefügt wurde und in einer ausgezeichneten globalen DB abgelegt wird. Eine globale Schemaevolution jedoch kann Auswirkungen auf diese hinzugefügte Entität haben, so daß eine Änderung bis auf die globale DB abgebildet wird.

Export-Schema ↔ Föderiertes Schema

Beim Übergang vom Export- zum Föderierten Schema werden Entitäten konstruiert und integriert. Dabei gibt es viele unterschiedliche Objektarten von global neuen Objekten bis zu reduziert föderierten vereinigten Objekten (siehe [Radeke 95]). Objekte oder deren Komponenten aus unterschiedlichen KDBS können zu einem Objekt zusammengesetzt oder als semantisch äquivalent definiert werden, so daß sie eine gemeinsame Repräsentation auf föderierter Ebene haben. Komplett neue Objekte werden in der globalen DB abgelegt. Bei der Abbildung einer lokalen Schemaevolution können sich Bestandteile eines aus verschiedenen lokalen Objekten zusammengesetzten Objektes verändern. Wenn ein solches zusammengesetztes Objekt von einer globalen Schemaänderung betroffen ist, werden die Änderungen auf ihre jeweiligen Entsprechungen der lokalen Objekte abgebildet.

Schwieriger gestaltet sich die Situation für Objekte auf föderierter Ebene, die auf mehrere semantisch äquivalente lokale Objekte abgebildet werden. Eine lokale Schemaänderung eines der äquivalenten Entitäten bewirkt das Aufbrechen der Äquivalenz, so daß eine neue globale Repräsentation gefunden werden muß. Nur ein Filtern der Schemaänderung kann die Aufhebung der Einheit verhindern. Jedoch bleibt dadurch die Schemaänderung global verborgen. Bei einer Schemaevolution auf föderierter Ebene muß die Änderung auf alle assoziierten lokalen Objekte abgebildet werden.

Föderiertes Schema ↔ Externes Schema

Das Externe Schema in einem FDBS entspricht einer herkömmlichen DB-Sicht auf das Föderierte Schema. Dabei können Transformationen in wiederum andere Datenmodelle vorgenommen oder wie schon beim Übergang vom Komponenten- in das Export-Schema Filtermechanismen eingesetzt werden. Eine Schemaänderung auf dem Föderierten Schema sollte die Externen Schemata möglichst nicht ändern. Dazu müssen die Abbildungen auf das Externe Schema dahingehend modifiziert werden, daß sich für die Anwendungen des FDBS ein unverändertes Bild auf die Daten bietet. Falls das Föderierte Schema allerdings reduziert wird, hat diese Reduktion auch Auswirkungen auf die Externen Schemata. Wenn eine neue Anforderung an ein Externes Schema gestellt wird, die nicht durch Änderung der Abbildung auf das Föderierte Schema abgefangen werden kann, muß diese Schemaänderung an das Föderierte Schema weitergereicht werden. Diese Problematik entspricht der Problematik von Sichten in Datenbanken. Deshalb wird hier der Fokus auf die tatsächlich manifestierten Änderungen des Föderierten Schemas gelegt.

5 Werkzeug für Schemaevolution

Bei der Abbildung der Schemaevolution zwischen Lokalem und Komponenten-Schema müssen Datenmodellgrenzen überschritten werden. Diese Abbildung wird nun näher untersucht. Während des Aufbaus eines FDBS werden die Schemata der anzukoppelnden KDBS integriert. Dabei werden auch die Abbildungen zwischen den Schemakomponenten der verschiedenen Schemaebenen definiert, so auch die Abbildung zwischen der Darstellung jedes Schemas in seinem KDBS Datenmodell und der des kanonischen Datenmodells.
Zusätzlich zu dieser Abbildung zwischen den Datenmodellen ist die Abbildung bekannt, die die Schemaevolution realisiert. Zwei Abbildungen sind bekannt:

- Abbildung einer Schemakomponente von einer Schemaebene in eine andere: Transformation.
- Abbildung einer Schemakomponente innerhalb einer Schemaebene in eine modifizierte Schemakomponente: Modifikation.

Aus diesen beiden Abbildungen können nun Vorschläge für eine modifizierte Schemakomponente in der anderen Schemaebene entwickelt werden. Die modifizierte und transformierte Schemakomponente kann durch zwei alternative Abbildungen erreicht werden (siehe auch Abb. 4):

- Die transformierte Schemakomponente wird modifiziert (Abb. 4 (i)).
- Die modifizierte Schemakomponente wird transformiert (Abb. 4 (ii)).

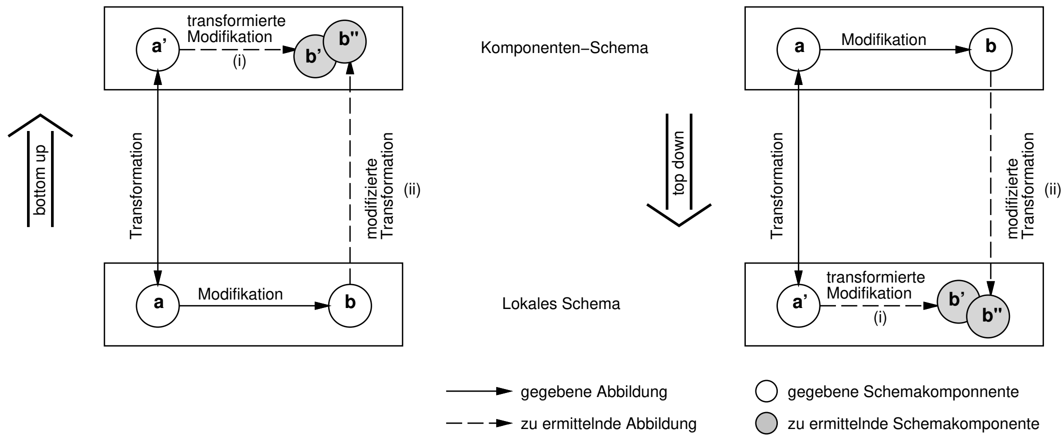

Abbildung 4: Abbildungen von Schemaänderungen in ein anderes Datenmodell

Dabei werden beide Wirkungsrichtungen der Schemaevolution unterstützt. Bei einer Schemaevolution auf dem Komponenten-Schema oder dem Lokalen Schema ist die Änderungsabbildung von der alten zur neuen Schemakomponente bekannt. Ebenso ist die Abbildung der alten Schemakomponente zu ihrer Entsprechung in der jeweils anderen Schemaebene bekannt. Mit dieser Information läßt sich die veränderte Schemakomponente der angrenzenden Schemaebene ableiten und zwar für beide Schemaevolutionsrichtungen.
Diese verschiedenen Alternativen für Abbildungen der veränderten Schemakomponenten auf eine andere Schemaebene können dem Administrator vorgeschlagen werden. Dabei erhält er die Möglichkeit, eine Alternative auszuwählen oder eine eigene Abbildung zu definieren.
Im folgenden vermittelt ein einfaches Beispiel einen Eindruck von den verschiedenen Abbildungsalternativen.

Beispiel:
Ein relationales KDBS enthält Bücher und Autoren. Zusätzlich sollen nun Abhängigkeiten zwischen Büchern und Autoren eingeführt werden. In der Ausgangssituation werden Bücher und Autoren im relationalen Datenmodell auf dem Lokalen Schema der KDB (linke Seite) und im objekt-orientierten Datenmodell nach ODMG-93 Standard auf dem Komponenten-Schema (rechte Seite) dargestellt:

```
CREATE TABLE Book                                 class Book
( author    CHAR(30)  KEY,                        { d_String  author;
  title     CHAR(30),          ==>                  d_String  title;
  isbn      CHAR(20)      )                         d_String  isbn;  };

CREATE TABLE Author                               class Author
( name      CHAR(30)  KEY,                        { d_String  name;
  address   CHAR(50)      )    ==>                  d_String  address; };
```

Um Bücher und Autoren nun in Beziehung zu setzen, wird auf relationaler Ebene eine neue 1:n Beziehung in Form einer neuen Tabelle eingeführt (linke Seite). Diese Beziehung wird auch auf das objekt-orientierte Komponenten-Schema übertragen. Dies wird erreicht, indem die bereits modifizierte Schemakomponente in das objekt-orientierte Datenmodell transformiert wird. Das Ergebnis ist eine neue Klasse auf der Komponenten-Schemaebene, die als Attribute die Schlüssel der zwei Klassen, die es verbinden soll, hat (rechte Seite).

```
CREATE TABLE RelBookAuthor                        class RelBookAuthor
( isbn      CHAR(20)  KEY,      ==>               { d_String  isbn;
  name      CHAR(30)  KEY )                         d_String  name; };
```

Eine andere Abbildungsmöglichkeit ergibt sich, wenn die bereits in die nächste Schemaebene transformierte Schemakomponente modifiziert wird. Die Schemaänderung wird also auf die bereits transformierte Komponente angewendet. Das bedeutet in diesem Fall, daß die neue Relation zwischen Autoren und Büchern im objekt-orientierten Datenmodell als Referenz in den beiden betreffenden Klassen realisiert wird.

```
class Book                                        class Author
{ d_String  title;                                { d_String  name;
  d_String  isbn;                                   d_String  address;
  Set<Ref<Author>> author inverse book; };          Set<Ref<Book>> book inverse author; };
```

Diese beiden Abbildungen ergeben unterschiedliche Darstellungen im objekt-orientierten Datenmodell. Es ist dem Administrator überlassen, welches Verfahren er anwendet.

6 Zusammenfassung

Es wurde gezeigt, wie Schemaevolution über die verschiedenen Schemaebenen eines FDBS abgebildet wird. Das Hauptaugenmerk wurde dabei auf den Übergang zwischen den unterschiedlichen Datenmodellen der KDBS einerseits und dem kanonischen Datenmodell des FDBS andererseits gelegt. Diese Untersuchungen bilden die Basis für die Entwicklung eines Werkzeugs für den FDBS-Administrator. Dieses Werkzeug ermittelt die Abbildungen, die nötig sind, um die Änderungen vom Föderierten bis zum Lokalen und vom Lokalen zum Föderierten Schema zu propagieren. Dieses Vorschlagswerkzeug offeriert dem FDBS-Administrator eine Auswahl von Abbildungen und zusätzlich die Möglichkeit, einen eigenen Vorschlag mit einzubringen.

Literatur

[OpenDM] "OpenDM 1.0 System Overview". *C-LAB, Paderborn* Germany, March 1996.

[Radeke 95] Radeke, Elke "Federation and Migration Among Database Systems". *Dissertation.* 1995.

[Sheth et al. 90] Sheth, A., Larson, J. "Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases". *ACM Computing Surveys,* Vol. 22, No. 3, September 1990.