Ein Schema-Transformationsansatz für Datenbank-Agenten in FDBS

Stefan Sander
SAPHIR - Unternehmensberatung, Mendelstraße 11, D-48149 Münster, s.sander@saphir.net

Wilhelm Hasselbring[1]
Universität Dortmund, Software-Technologie, D-44221 Dortmund, hasselbring@acm.org

1. Einleitung

Föderierte Datenbanksysteme (FDBS) stellen einen Ansatz zur Integration von heterogenen Datenbeständen dar. FDBS versuchen mithilfe einer zusätzlichen Softwareschicht die Heterogenität der zu integrierenden Datenbanksysteme transparent zu machen und eine neue integrierte Sicht auf die verteilten Daten einer Organisation zu unterstützen oder auch die Integrität der Daten in verschiedenen Systemen zu sichern. Im Rahmen der Föderation bewahren die bestehenden Datenbanksysteme weitgehend ihre Autonomie, so daß bestehende Anwendungen weiterhin genutzt werden können. Insofern ist ein föderiertes Datenbanksystem durchaus mit einer politischen Föderation, wie etwa der *Europäischen Union* oder der *UNO*, zu vergleichen. Ähnlich wie bei einer politischen Föderation besteht die besondere Problematik bei der Entwicklung eines FDBS in der Heterogenität der zu integrierenden Teilsysteme. Die Transformation lokaler Schemata in kanonische Schemata ist dabei ein wichtiger Aspekt, bevor die verschiedenen Schemata zu einem föderierten Schema integriert werden können [SL90, Con97]. In Abschnitt 2 wird zunächst die Software-Architektur eines realisierten FDBS präsentiert. Abschnitt 3 stellt das in diesem Zuammenhang entwickelte Datenbank-Agenten-Konzept dar, mit dessen Hilfe die Datenmodell-Heterogenität der zu integrierenden Datenbanksysteme gelöst wurde.

2. Architektur eines föderierten Datenbanksystems

Dieser Abschnitt stellt die grobe Architektur eines entwickelten FDBS vor. Hierbei wird bereits auf die Datenbank-Agenten eingegangen. Der detaillierte Entwurf der Datenbank-Agenten ist jedoch Gegenstand von Abschnitt 3. Die Systemkomponenten sind in Abb. 1 in der Übersicht dargestellt. Das FDBS wurde als System verteilter Objekte, die mithilfe eines CORBA-kompatiblen Object Request Brokers kommunizieren, konzipiert. Im Mittelpunkt der Architektur steht der FDBS-Kern, welcher die Kooperation der Komponenten-Datenbanksysteme realisiert und den globalen Anwendungen eine homogenisierte Sicht auf die verteilten, heterogenen Datenbestände bietet. Die Meta-Daten des FDBS (die föderierten Schemata, die Lokalisation einzelner DB-Objekte und die Komponenten-Schemata) werden in einer ODMG-93-konformen Hilfs-Datenbank verwaltet.

Die Datenbank-Spezifika der Komponenten-Datenbanksysteme werden durch die Datenbank-Agenten gekapselt. Der FDBS-Kern hat also nur indirekt Zugriff auf die KDBS. Die wesentliche Aufgabe der Datenbank-Agenten besteht in der Schematransformation in das kanonische Datenmodell (Überwindung der Datenmodell-Heterogenität). Die Kommunikation zwischen dem FDBS-Kern und den Datenbank-Agenten erfolgt über den Object Request Broker.

Einige Systemkomponenten des FDBS-Kerns und deren Aufgaben werden im folgenden kurz skizziert. Im ***Data-Dictionary*** werden die Meta-Daten des FDBS verwaltet. Es beinhaltet die Komponenten-Schemata, die föderierten Schemata, etc. Die im *Data-Dictionary* verwalteten Meta-Daten des FDBS werden selbst wieder durch ein Datenbank-Schema beschrieben. Dieses, im folgenden als "*Meta-Schema*" be-

1. Neue Adresse ab August 1998: Infolab, Department of Information Management and Computer Science, Tilburg University, 5000 LE Tilburg, Niederlande.

zeichnete Datenbank-Schema definiert, wie die Meta-Daten des FDBS in der ODMG-93-konformen Datenbank abgelegt werden. Um neue Komponenten-Datenbanksysteme möglichst einfach in die Datenbank-Föderation zu integrieren, ermöglicht das *Data-Dictionary* einen dynamischen Import von Komponenten-Schemata. Der ***Event-Handler*** nimmt Ereignisse der Datenbank-Agenten entgegen und organisiert deren parallele Bearbeitung durch Delegation an die anderen Systemkomponenten des FDBS-Kerns. Der ***Object-Manager*** verwaltet zu allen lokalen DB-Objekten entsprechende Stellvertreter-Objekte (Proxy-Objekte), die einen eindeutigen und dauerhaften *Object-Identifier* (*OID*) haben [EK91]. Der *Object-Manager* realisiert eine wertunabhängige Objekt-Identität. Die Stellvertreter-Objekte kapseln Informationen über die Lokalisation und den Status von lokalen DB-Objekten. Darüber hinaus realisieren sie eine Abbildung auf die lokale Identität des korrespondierenden lokalen DB-Objektes. Der *Object-Manager* greift bei seiner Arbeit auf das *Data-Dictionary* zurück, um DB-Objekte zu materialisieren. Er ist das Bindeglied zwischen dem *Event-Handler* und dem *Replication-Manager*, der den Replikations-Mechanismus des FDBS kontrolliert. Der ***Replication-Manager*** realisiert ein *Publisher-Subscriber-Replikations-Verfahren* [Has97, San97] und stützt sich bei dieser Aufgabe auf das *Data-Dictionary* und den *Object-Manager* ab.

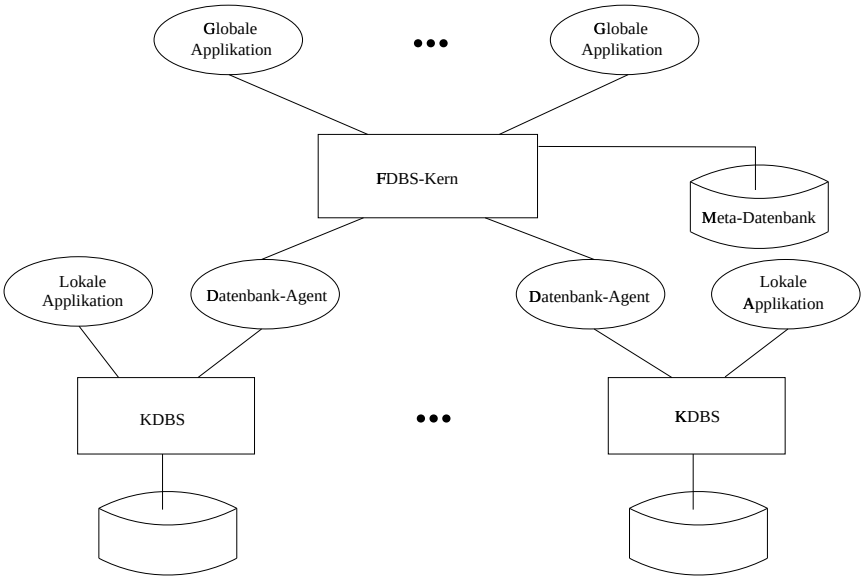

Abbildung 1: Die grobe Software-Architektur.

Jeder Datenbank-Agent implementiert ebenfalls eine Reihe von Systemkomponenten. Das ***lokale Data-Dictionary*** verwaltet das Komponenten-Schema des Komponenten-Datenbanksystems (KDBS) und dessen bijektive Abbildung auf das lokale Schema (Schematransformation). Das Komponenten-Schema besteht aus einer Menge von Objekten mit IDL-Interface, welche die Schema-Elemente gemäß dem kanonischen Datenmodell beschreiben. Auf diese Objekte greift das *Data-Dictionary* des FDBS-Kerns zu, um das Komponenten-Schema des KDBS zu importieren (beim ersten Ankoppeln des KDBS) oder mit einer bereits vorhandenen Kopie abzugleichen. Der ***lokale Object-Manager*** transformiert Operationen auf den DB-Objekten des Komponenten-Schemas entsprechend der bijektiven Schematransformations-Abbildung in Operationen bezüglich des lokalen Schemas. Darüber hinaus hat er die Aufgabe, die DB-Objekte des Komponenten-Schemas mit eindeutigen, wertunabhängigen und dauerhaften *lokalen Object-Identifiern* (*LOID*) zu versehen. Dadurch wird gewährleistet, daß alle KDBS gegenüber dem FDBS-Kern das gleiche Objekt-Identitäts-Konzept unterstützen [EK91].

3. Die Datenbank-Agenten

In dem nun folgenden Abschnitt soll der Entwurf der Datenbank-Agenten des FDBS dargestellt werden. Die Entwicklung eines Datenbank-Adapters ist dabei durch das kanonische Datenmodell und das Interface zum FDBS-Kern determiniert. Die Schnittstellen zwischen den Datenbank-Agenten und dem FDBS-Kern sind in der CORBA-Interface-Definition-Language (IDL) spezifiziert [Sie96]. Diese stellen die Kooperationsgrundlage zwischen dem FDBS-Kern und den Datenbank-Agenten dar. Als Grundlage

des kanonischen Datenmodells des FDBS dient der ODMG-93-Standard [Cat96]. Entsprechend der 5-Ebenen-Schema-Architektur nach Sheth und Larson [SL90] besteht die Aufgabe eines Datenbank-Adapters nun darin, den Übergang vom lokalen Datenmodell (lokales Schema) zum kanonischen Datenmodell (Komponenten-Schema) für den FDBS-Kern transparent zu machen. Dieser Prozeß wird als Schema-Transformation oder Homogenisierung bezeichnet [HK95].

Der gewählte Schema-Transformationsansatz für relationale Komponenten-Datenbanksysteme hat zum Ziel, einen Großteil der impliziten Semantik in relationalen Datenbank-Schemata in den Komponenten-Schemata wieder explizit zu machen, um so die Integration der heterogenen Datenbestände der KDBS zu erleichtern. Dieser Aspekt der Schema-Transformation wird als semantische Anreicherung [HK95] bezeichnet. Dies ist natürlich nur möglich, wenn die Ausdrucksstärke des kanonischen Datenmodells größer als die des relationalen Datenmodells ist [SCG91].

Relationale Datenbank-Schemata werden heute häufig mithilfe semantischer Datenmodelle, wie z.B. dem Entity-Relationship-Modell modelliert. Viele semantische Aspekte der ursprünglichen Modellierung, wie z.B. Generalisierungs- / Spezialisierungs- oder Assoziations-Beziehungen gehen bei der Transformation in ein relationales Datenbank-Schema verloren [Kro93], da das relationale Datenmodell nur einige wenige Datenstrukturierungs-Konzepte unterstützt. Ein weiterer Grund für diesen semantischen Verlust ist der Prozeß der Normalisierung [HK95], der zur Folge hat, daß die ursprünglichen Objekttypen des logischen DB-Schemas jeweils durch eine Vielzahl von Relationen repräsentiert werden und der ursprüngliche Entwurf sich häufig nicht mehr erkennen läßt. Ziel der Normalisierung ist es, Redundanz durch die Berücksichtigung von funktionalen Abhängigkeiten zu beseitigen [Dat95]. Demgegenüber steht, daß Redundanz manchmal bewußt in relationalen DB-Schemata eingesetzt wird, um den Lese-Zugriff auf Daten zu beschleunigen. Insgesamt läßt sich feststellen, daß es eine Vielzahl unterschiedlicher Auffassungen bezüglich der Modellierung von relationalen Datenbank-Schemata gibt. Hierauf muß der Schema-Transformationsansatz Rücksicht nehmen.

Bei der Entwicklung eines Schema-Transformationsansatzes stellt sich zunächst die Frage, welche Datenstrukturierungskonzepte des kanonischen Datenmodells überhaupt bei der Transformation der relationalen Schemata von Interesse sind. Erst dann sind Überlegungen anzustellen, wie die verschiedenen Modellierungs-Konzepte eines relationalen DB-Schemas darauf abgebildet werden können und wie diese Abbildung repräsentiert wird. Im folgenden werden nun die Konzepte des kanonischen Datenmodells, die im Schema-Transformationsansatz Berücksichtigung finden, in Bezug auf das relationale Modell erläutert.

Die *Objekttypen* des ODMG-93-Objektmodells entsprechen am ehesten den Relationen eines relationalen DB-Schemas. Jedoch entspricht nicht jede Relation einem *Objekttypen*. Darüber hinaus unterscheidet sich die Identifizierung von *DB-Objekten* und Tupeln. *Supertyp / Subtyp-Beziehungen* werden im relationalen Modell nicht direkt unterstützt. Vererbungsbeziehungen werden jedoch in relationalen Schemata häufig indirekt angewandt. Dabei existieren unterschiedliche Ansätze für die Repräsentation von Vererbungsbeziehungen in einem relationalen Schema [HK95]. *Beziehungen* zwischen Objekten (Tupeln) werden im relationalen Modell ebenfalls nicht direkt unterstützt. Sie werden jedoch mithilfe von Fremdschlüsseln in einem relationalen Schema umgesetzt. Das Navigieren entlang dieser "relationalen Beziehungen" ist ebenfalls nicht möglich, sondern erfolgt indirekt durch den Wertevergleich von Fremdschlüsseln und Primärschlüsseln (Join-Condition) innerhalb einer Datenbankabfrage. Es ist also die Aufgabe der Schema-Transformation, die eigentlichen Beziehungen wieder explizit zu machen. Die Wertebereiche der *Attribute* eines Objektes im ODMG-93-Objektmodell sind wiederum Objekte. Diese können, im Gegensatz zu den Attributen einer Relation, bei denen nur atomare Datentypen als Wertebereich zulässig sind (1. Normalform), unter Umständen auch strukturiert sein. Ein Objekt mit strukturierten Attributen wird auch als komplexes oder aggregierendes Objekt bezeichnet. Komplexe Objekte zerfallen im relationalen Modell in eine Menge von Tupeln, die über unterschiedliche Tabellen verstreut und durch Fremdschlüssel quasi zusammengekittet sind. Die Schema-Transformation könnte diese komplexen Objekte wieder explizit machen.

Die grundlegenden Konzepte dieses Schema-Transformationsansatzes sind in Abb. 2 illustriert. Die Konzepte des relationalen Datenmodells und des kanonischen Datenmodells (ODMG-93) werden durch die Klassen eines objektorientierten ***Meta-Schemas*** modelliert. Die Instanzen dieser Klassen entsprechen dabei den Schema-Elementen des lokalen Schemas (relationales Datenmodell) und des Komponen-

ten-Schemas (ODMG-93). Dieses Meta-Schema zerfällt entsprechend der Unterscheidung zwischen dem Komponenten-Schema und dem lokalen Schema in zwei Teilbereiche. Die Beziehungen zwischen den Klassen dieser Teilbereiche des *Meta-Schemas* modellieren die ***Schema-Transformations-Abbildung***. Das *Meta-Schema* beschreibt also bereits, wie die Konzepte des relationalen Datenmodells auf die des ODMG-93-Objektmodells abgebildet werden können. Die Beziehungen zwischen Instanzen der Klassen des *Meta-Schemas* repräsentieren eine konkrete ***Schema-Transformations-Abbildung*** eines existierenden relationalen DB-Schemas auf ein Komponenten-Schema.

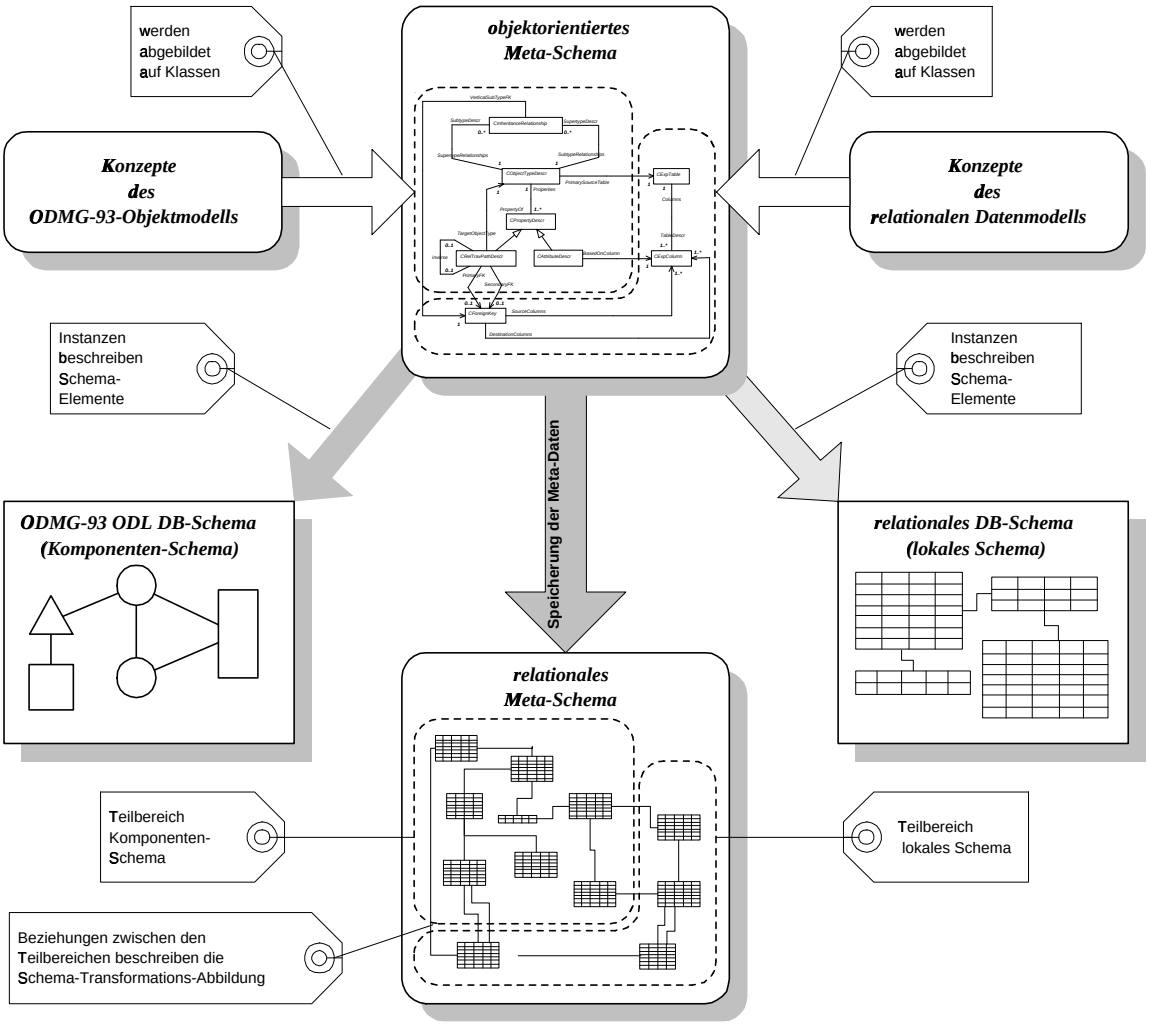

Abbildung 2: Übersicht über den Schema-Transformationsansatz.

Die persistente Speicherung der Meta-Daten bzw. der ***Schema-Transformations-Abbildung*** erfolgt im relationalen Komponenten-Datenbanksystem. Hierzu wird das objektorientierte *Meta-Schema* auf ein korrespondierendes relationales *Meta-Schema* abgebildet. Für diese "***two level store***"-Vorgehensweise gibt es mehrere Gründe. Um einen möglichst effizienten Datenbank-Agenten zu entwickeln, müssen die Möglichkeiten eines relationalen Komponenten-Datenbanksystem ausgenutzt werden. Dies betrifft insbesondere die Verlagerung eines Teils der Datenbank-Agenten-Logik auf den relationalen Datenbank-Server. Dies geschieht mithilfe von ***S**tored-Procedures* und *Triggern*. Da dieser Teil der Datenbank-Agenten-Logik ebenfalls auf den *Meta-Daten* bzw. der ***S**chema-Transformations-Abbildung* arbeitet, ist eine Speicherung der *Meta-Daten* im relationalen Komponenten-Datenbanksystem naheliegend.

Wie bereits deutlich wurde, besteht der FDBS-Kern aus einer Reihe von Systemkomponenten, die als verteilte Objekte realisiert wurden. Der äußere Rahmen des FDBS-Kerns wird dabei ebenfalls als verteiltes Objekt realisiert. Dieses Objekt ist eine Instanz der Klasse *CFDBSKernel*. Es organisiert die Initialisierung der weiteren Systemkomponenten des FDBS-Kerns beim Hochfahren des Systems und ermöglicht das An- und Abkoppeln der Datenbank-Adapter. Beim Systemstart des FDBS erzeugt das *CFDBSKernel-Objekt* jeweils ein Objekt der Klassen *CFDBSDataDictManager, CFDBSObjectManager* und *CFDBSEventHandler*, welche die weitere Initialisierung des Systems vornehmen.

Zum An- und Abkoppeln der Datenbank-Adapter stellt das *CFDBSKernel-Objekt* die Operationen *PluginDBAdapter*, *ConnectDBAdapter* und *DisconnectDBAdapter* zur Verfügung. Bevor jedoch ein Komponenten-Datenbanksystem an der Föderation teilnehmen kann, muß dessen Komponenten-Schema vom FDBS-Kern importiert und integriert werden. Abb. 3 illustriert die Interaktion der beim Schema-Import und beim Ankoppeln eines Datenbank-Adapters beteiligten Objekte durch Sequenzdiagramme. Der Datenbank-Adapter besorgt sich zunächst vom *ORB* bzw. dessen *Naming Service* die Objekt-Referenz des *CFDBSKernel-Objekts*. Der Import eines Komponenten-Schemas wird durch die Operation *PluginDBAdapter* eingeleitet. Das *CFDBSKernel-Objekt* delegiert diese Aufgabe an den *DataDictinaryManager* (Instanz der Klasse *CFDBSDataDictManager*), indem er dessen Operation *Add_DataSource* aufruft. Dieser fragt den Datenbank-Adapter zunächst nach der Objekt-Referenz seines lokalen *DataDictinaryManager*. Nun findet der eigentliche Import des Komponenten-Schemas statt.

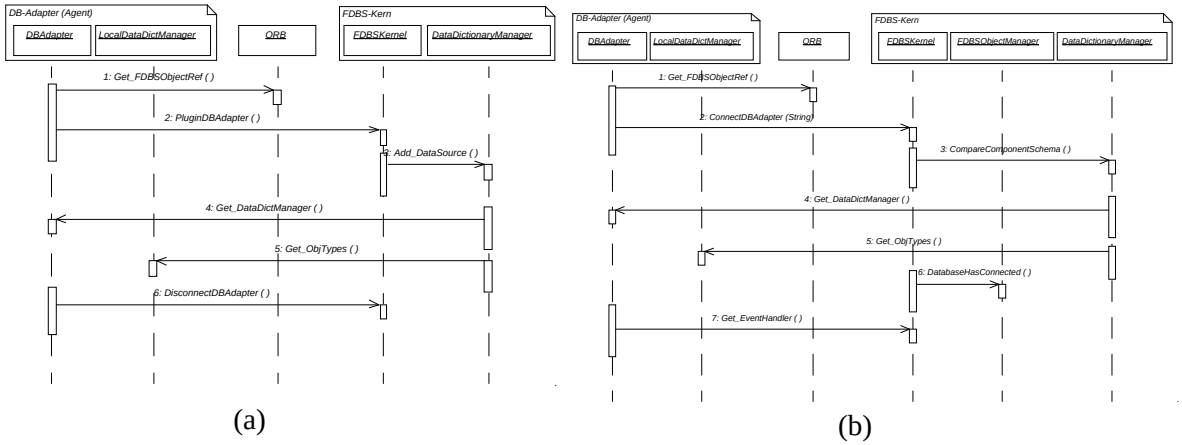

Abbildung 3: Import eines Komponenten-Schemas (a) und Ankoppeln eines Datenbank-Adapters (b).

Das Ankoppeln eines Komponenten-Datenbanksystems, dessen Komponenten-Schema bereits erfolgreich importiert und integriert wurde, geschieht mittels der Operation *ConnectDBAdapter* (Abb. 3b). Den Abgleich des Komponenten-Schemas erledigt der *DataDictionaryManager* des FDBS-Kerns in Kooperation mit seinem lokalen Pendant, wie es auch schon beim Schema-Import dargestellt wurde. Stimmen beide Schemata überein, so informiert das *CFDBSKernel-Objekt* zunächst den *FDBSObjectManager* darüber, daß ein KDBS erfolgreich angebunden wurde. Der *FDBSObjectManager* veranlaßt, daß alle zwischenzeitlichen Änderungen von DB-Objekten im Komponenten-Datenbanksystem nachgezogen werden. Nach der erfolgreichen Ankoppelung verschafft sich der Datenbank-Adapter die Objekt-Referenz des *FDBSEventHandlers* und weist seinen lokalen Event-Handler an, die Event-Verarbeitung aufzunehmen. Die Event-Verarbeitung erfolgt durch die Kooperation zwischen den lokalen Event-Handlern und dem *FDBSEventHandler*. Für eine detailliertere Darstellung sei auf [San97] verwiesen.

Literatur

[Cat96] R.G. Cattell. The Objectdatabase Standard: ODMG'93. Morgan Kaufmann Publisher, 1996.

[Con97] S. Conrad: Föderierte Datenbanksysteme, Springer-Verlag, 1997.

[Dat95] C. J. Date. An Introduction to Database Systems. Volume 1. Addison Wesley, 6. Edition, 1995.

[[EK91] F. Eliassen, und R. Karlsen. Interoperability and Object Identity. SIGMOD Record, 20 (4): 25 - 29, 1991.

[Has97] W. Hasselbring: Federated Integration of Replicated Information within Hospitals. International Journal on Digital Libraries 1(3): 192-208, November 1997.

[HK95] U. Hohenstein and C. Körner. Semantische Anreicherung relationaler Datenbanken. In G. Lausen, editor, *Proc. BTW'95, Dresden,* S. 130-149. Informatik aktuell, Springer-Verlag, Mai 1995.

[Kro93] P. Kroha. Objects and Databases. McGraw-Hill International Series in Software Engineering, 1993.

[San97] S. Sander: CORBA und ODMG-93 als Grundlage zur Realisierung eines föderierten Datenbanksystems, Diplomarbeit, FB Informatik, Universität Dortmund, 1997.

[SCG91] F. Saltor, M. Castellanos und M. Garcia-Solaco. Suitability of Data Models as Canonical Data Models in Federated Databases. SIGMOD Record 20 (2), S. 44-48, 1991.

[Sie96] J. Siegel. CORBA Fundamentals and Programming. John Wiley & Sons New York, 1996.

[SL90] A. Sheth und J. Larson. Federated database systems for managing distributed, heterogeneous,and autonomous databases. ACM Computing Surveys, 22(3):183-236,1990.