Realisierung eines föderierten Datenbanksystems auf Basis der Standards CORBA und ODMG-93

S. Sander und W. Hasselbring
Universität Dortmund, Informatik 10 (Software-Technologie), D-44221 Dortmund
E-mail: {sander|willi}@ls10.informatik.uni-dortmund.de

1 Einleitung

Der zunehmende Einsatz von Datenbanksystemen (DBS) führte in den letzten Jahren dazu, daß heute in vielen Organisationen unterschiedliche Datenbanksysteme nebeneinander im Einsatz sind. Die Gründe hierfür liegen einerseits darin, daß verschiedene Datenbankmanagementsysteme (DBMS) sich in unterschiedlicher Weise gut für bestimmte Anwendungsbereiche eignen, andererseits unterstützen Hersteller von spezieller Anwendungssoftware im allgemeinen nur bestimmte DBMS. Hieraus resultiert, daß der die Organisation betreffende Realweltauschnitt auf eine Menge von unterschiedlichen Datenhaltungssystemen abgebildet wird. Es zeigt sich, daß diese Systeme sehr oft inhaltlich verwandte Daten verwalten [Rah94]. Diese Situation verschärft sich auch in dem Maße, in dem Organisationen sich zusammenschließen, etwa bei der Fusionierung von Unternehmen oder durch Restruktierung von Behörden.

Das Betreiben dieser voneinander isolierten DBS innerhalb einer Organisation impliziert eine Reihe von gravierenden Problemen. Die redundante Haltung der gleichen Information in verschiedenen, nicht kooperienden Systemen legt dabei den Benutzern zusätzliche Mehrarbeit auf. Das Verknüpfen von Daten unterschiedlicher DBS wird nicht unterstützt und ist den Benutzern selbst überlassen. Für diese ist die Verteilung der Daten auf die unterschiedlichen Systeme voll sichtbar. Ein weiteres Problem ergibt sich durch die bereits bestehenden Anwendungen, welche einen großen Teil des Unternehmens-Know-how darstellen. Diese müssen modifiziert oder neu implementiert werden. Dieser Weg wird deshalb häufig aus ökonomischen Gründe erst gar nicht in Betracht gezogen. Außerdem hat sich gezeigt, daß unternehmensweite Datenmodelle in ihrer Komplexität sehr schlecht zu handhaben sind. Betrachtet man zum Beispiel das betriebswirtschafliche Standardsoftwarepaket SAP R3, so zeigt sich, daß allein die Einführung eines unternehmensweiten Datenmodells — bei schon vorhandener Software — enorme Probleme impliziert, hohe Kosten verursacht und großen Zeitaufwand in Anspruch nimmt [Hol96]. Es erscheint also sinnvoll, nach Alternativen zur Integration der Datenbestände einer Organisation zu suchen.

Ein vielversprechender Ansatz zur Integration von heterogenen Datenbeständen innerhalb einer Organisation stellen föderierte Datenbanksysteme (FDBS) dar [SL90]. Dabei werden bestehende Datenbanksysteme zu einem Gesamtsystem föderiert. Dies geschieht durch Einführung einer zusätzlicher Softwareschicht, welche den einheitlichen Zugriff auf alle Daten realisiert. Die an der Föderation beteiligten DBS werden als Komponenten-DBS bezeichnet (KDBS). Die KDBS bewahren im Rahmen der Föderation weitgehend ihre Autonomie. Bestehende Anwendungen (legacy systems) können weiterhin auf die lokalen Komponenten-DBS zugreifen. Bereits getätigte Investitionen in Informationssysteme bleiben somit gesichert.

In [SL90] wird zwischen lose und eng gekoppelten FDBS unterschieden. Bei lose gekoppelten FDBS greifen die Benutzer mittels einer Multidatenbanksprache auf die verschiedenen Komponentendatenbanksysteme zu. Die Aufgabe der Schemaintegration ist dabei dem jeweiligen Administrator der KDBS überlassen. Im Gegensatz dazu stellt sich ein eng gekoppeltes FDBS dem Benutzer wie ein einziges logisches Datenbanksystem dar. Die Redundanzproblematik der in einer Organisation vorhandenen Datenbestände läßt sich durch ein eng gekoppeltes FDBS lösen. Nur in eng gekoppelten Systemen ist es auch möglich, systemübergreifende Integritätsbedingungen zu gewährleisten.

Abschnitt 2 stellt kurz grundsätzliche Lösungsansätze für eng gekoppelte Systeme vor. Abschnitt 3 diskutiert den Standard ODMG-93 als kanonisches Datenmodell eines FDBS und Abschnitt 4 die Bewältigung der Heterogenität durch den Standard CORBA. Abschnitt 5 beschreibt dann die Architektur des angestrebten föderierten Datenbanksystems, das zur Evaluation der Eignung dieser Standards für diesen Zweck realisiert werden soll, und Abschnitt 6 faßt die Projektziele zusammen.

2 Grundsätzliche Lösungsansätze

Viele FDBS-Projekte verfolgen den Ansatz, die Datenbankspezifika der zu integrierenden Komponentendatenbanksysteme in *Datenbankadaptern* zu kapseln [HFBK94, HTJ+95, RS95]. Dieser Ansatz soll auch im vorgestellten Projekt weiterverfolgt werden.

Eine Kopplungsschicht realisiert dabei die Kommunikation zwischen dem FDBMS-Kern und den jeweiligen Datenbankadaptern. Die Adapter leisten die Transformation der Daten und Operationen der lokalen DBS in das kanonische Datenmodell der Föderierungsschicht. Dadurch ist der Kern des FDBS unabhängig von den einzelnen Komponentendatenbanksystemen. In einer Hilfs-DBS werden die Meta-Daten des föderierten Systems verwaltet. Die Integration eines weiteren DBS beschränkt sich darauf, den entsprechenden Datenbankadapter zur Verfügung zu stellen und an die Kopplungsschicht zu binden. Der generische Kern des FDBS bleibt unverändert. Der Administrator hat nun noch die Aufgabe, die Schemaintegration innerhalb des kanonischen Datenmodells durchzuführen. Die globalen Applikationen greifen auf das FDBS über externe Adapter zu. Diese realisieren die externen Schemata der 5-Ebenen Referenz-Architektur für FDBS [SL90].

Dieser Ansatz bietet auch eine Basis für eine schrittweise Reduktion der Anzahl, der in einer Organisation betriebenen Datenbanksysteme. Wie in [RS95] dargelegt, stellt Objekt-Replikation und -Migration ein probates Mittel für eine sanfte Migration von nicht mehr zeitgemäßen, weniger robusten Datenhaltungssystemen zu modernen Datenbanksystemen dar. Die Daten des Alt-Systems werden über den Replikationsmechanismus in ein modernes DBS repliziert. Wurde die Testphase erfolgreich abgeschlossen, kann das alte DBS außer Betrieb genommen werden.

3 ODMG-93 als kanonisches Datenmodell eines FDBS

Das Objektmodell ODMG-93 soll als kanonisches Datenmodell (KDM) der Föderierungsschicht dienen. Wesentliche Vorteile des ODMG-93 Objektmodells als KDM sind:

- Relationale Datenbankschemata werden heute häufig durch objektorientierte Techniken oder erweiterte E/R-Modelle modelliert. Viele Aspekte (z.B. Generalisierung / Spezialisierung) des ursprünglichen Entwurfs gehen bei der Transformation in das der Datenbank unterliegende relationale Modell verloren [Kro93]. Diese Aspekte stecken implizit im Code der Anwendungen, welche auf dem relationalen Schema basieren, und sind evtl. durch die ursprünglichen Entwurfs-Dokumente noch verfügbar. Diese können im korrespondierenden Komponenten-Schema wieder explizit gemacht werden. Dieser Prozeß wird auch als *semantischer Anreicherung* bezeichnet [HP96].
- Die Integration von objektorientierten Datenbankschemata läßt sich weitgehend ohne semantische Verluste durchführen.
- Ein nicht objektorientiertes Datenmodell würde zu einem "two level store" Ansatz [Cat96] für globale Anwendungen führen.

Einige Nachteile des ODMG-93 Objektmodells als KDM sind:

- Eine automatische Generierung eines Komponenten-Schemas aus einem lokalen Schema, welches in einem semantisch ärmeren Datenmodell vorliegt (etwa dem relationalen Datenmodell), ist nahezu unmöglich [HP96, Hil95], weil bestimmte Modellierungssituationen des zu integrierenden Schemas zumeist durch unterschiedliche objektorientierte Konzepte abgebildet werden können.
- Derzeit existiert in ODMG-93 kein Sichtenkonzept.
- Maßnahmen zur Realisierung einer Zugriffskontrolle sind bisher nicht vorgesehen.

4 Bewältigung der Heterogenität

Eine Kernproblematik beim Entwurf und der Entwicklung eines föderierten Datenbanksystems liegt in der Heterogenität der beteiligten Komponenten-DBS begründet. Nach [Rah94] lassen sich in diesem Zusammenhang im wesentlichen drei Aspekte von Heterogenität unterscheiden:

- Heterogenität bezüglich der verschiedenen DBMS der Komponenten-DBS.

- Heterogenität in der Ablaufumgebung. Die einzelnen KDBS können auf verschiedenen Hardwareplattformen, Betriebssystemen und innerhalb verschiedener Kommunikationsnetze ablaufen.

- Semantische Heterogenität. Aufgrund der Entwurfsautonomie der an der Föderation beteiligten KDBS werden semantisch gleiche Objekte unterschiedlich benannt oder repräsentiert. Diese Schemakonflikte aufzulösen ist Aufgabe der Schemaintegration.

Bezüglich der Heterogenität in der Ablaufumgebung soll der Ansatz verfolgt werden, die Komponenten des angestrebten FDBS als verteilte, kooperierende Objekte zu konzipieren und zu realisieren. Diese kommunizieren über einen CORBA-kompatiblen Object Request Broker (ORB) [OHE96]. Hierdurch werden auf relativ einfache Weise die Grenzen der Ablaufumgebungen der an dem FDBS beteiligten Komponenten-DBS überwunden und ein einheitlicher Interaktionsmechnismus der FDBS-Komponenten realisiert. Die Schnittstellen der einzelnen Module sind dabei durch die IDL (Interface Definition Language) programmiersprachenunabhängig spezifiziert.

Ein weiterer Vorteil dieses Ansatzes ist eine stärkere Modularisierung des angestrebten Systems: auch innerhalb des FDBS-Kerns soll die Kommunikation über den ORB erfolgen. Es soll möglich sein, bestimmte Komponenten (verteilte Objekte) des FDBS durch andere zu ersetzen, ohne dabei den Rest des Systems modifizieren zu müssen (insbesondere neu übersetzen zu müssen). Beispielsweise ist es damit möglich, einen Two-Phase-Locking basierten Transaktions-Manager des FDBS-Kerns durch einen anderen zu ersetzen, welcher ein optimistisches Transaktions-Protokoll realisiert.

Die Heterogenität bezüglich der verschiedenen Datenbankmanagementsysteme wird durch die Transformation der lokalen Schemata in Komponenten-Schemata für des FDBS-Kern transparent gemacht. Auf dieser Stufe ist dann die Datenmodellheterogenität gelöst. Die Datenbankadapter übernehmen diese Aufgabe. In Bezug auf die semantische Heterogenität sollen Überlegungen angestellt werden, wie der Prozeß der Schemaintegration durch das FDBS (halb-automatisch) unterstützt werden kann.

5 Architektur des angestrebten föderierten Datenbanksystems

Eine kleine Beispielapplikation, welche die Fähigkeiten des FDBS-Prototyps aufzeigen soll, wird zur Evaluation realisiert. Geplant ist folgendes Szenario: Bestimmte Abteilungen der Stadtverwaltungen BO und WAT werden im Rahmen einer Eingemeindung zusammengelegt. Hiervon betroffen ist z.B. das Standesamt. Das Einwohnermeldeamt der Stadt BO verwendet zur Verwaltung der Bürgerdaten ein relationales DBMS. Im Einwohnermeldeamt der Stadt WAT ist für denselben Zweck ein objektorientiertes DBMS im Einsatz. Die Beispielapplikation soll, basierend auf dem FDBS, nach Integration der bestehenden Systeme, das EDV-System des nun zentralen Standesamtes BO realisieren. Z.B. soll eine Hochzeit unter Ehepartnern aus beiden Städten erfolgen können, ohne daß dabei ein Neuanlegen von Personendaten erforderlich ist. Die Architektur dieses FDBS ist in Abbildung 1 grob dargestellt. Das System ist in vier Schichten angelegt.

In der untersten Schicht sind die zu integrierenden Komponenten-DBS POET 4.0 [POE96] und SQL-Base 6.1 [Cen96] angesiedelt. In der nächst höheren Schicht sind die datenbankspezifischen Details durch die Datenbankadapter gekapselt. Alle Datenbankadapter stellen dem Kern des FDBS dieselbe Schnittstelle zur Verfügung. Dieser kooperiert dann über eine Menge von Zugriffsoperationen mit allen beteiligten Komponenten-DBS in der gleichen Art und Weise. Die Datenbankadapter realisieren dabei die Schematransformation vom lokalen Schema in das Komponenten-Schema. Der Kern des FDBS kommuniziert mit den Adaptern über den ORB. Die Adapter sind als verteilte Objekte realisiert und stellen ihre Dienste über das in der IDL spezifizierte Interface zur Verfügung.

Sämtliche Meta-Daten über das föderierte Schema, die Lokalisation einzelner Datenbankobjekte und die Komponenten-Schemata werden in einer Hilfs-Datenbank verwaltet. Diese Aufgabe soll das objektorientierte Datenbanksystem POET übernehmen. Die globalen Applikationen, die sich auf das FDBS stützen, enthalten als inhärenten Teil gemäß dem ODMG-93 C++ Binding bestimmte Klassen, um Zugriff auf das FDBS zu erhalten. Die Kommunikation mit dem FDBS wird über den ORB getätigt. Für einen Anwendungsentwickler stellt sich das FDBS wie ein ODMG-93-konformes Datenbanksystem dar, falls ein ODMG-93-konformer Adapter eingesetzt wird.

Der Kern des FDBS zerfällt in sechs Komponenten. Die Komponenten innerhalb des Kerns sind ebenfalls als verteilte Objekte realisiert und interagieren mit Hilfe des ORB. Der Dictionary-Manager stellt Operationen für die Verwaltung und den Zugriff auf das föderierte Schema und die Komponenten-Schemata zur Verfügung. Da im ODMG-93 Standard bisher kein Dictionary spezifiziert wurde, stellen die

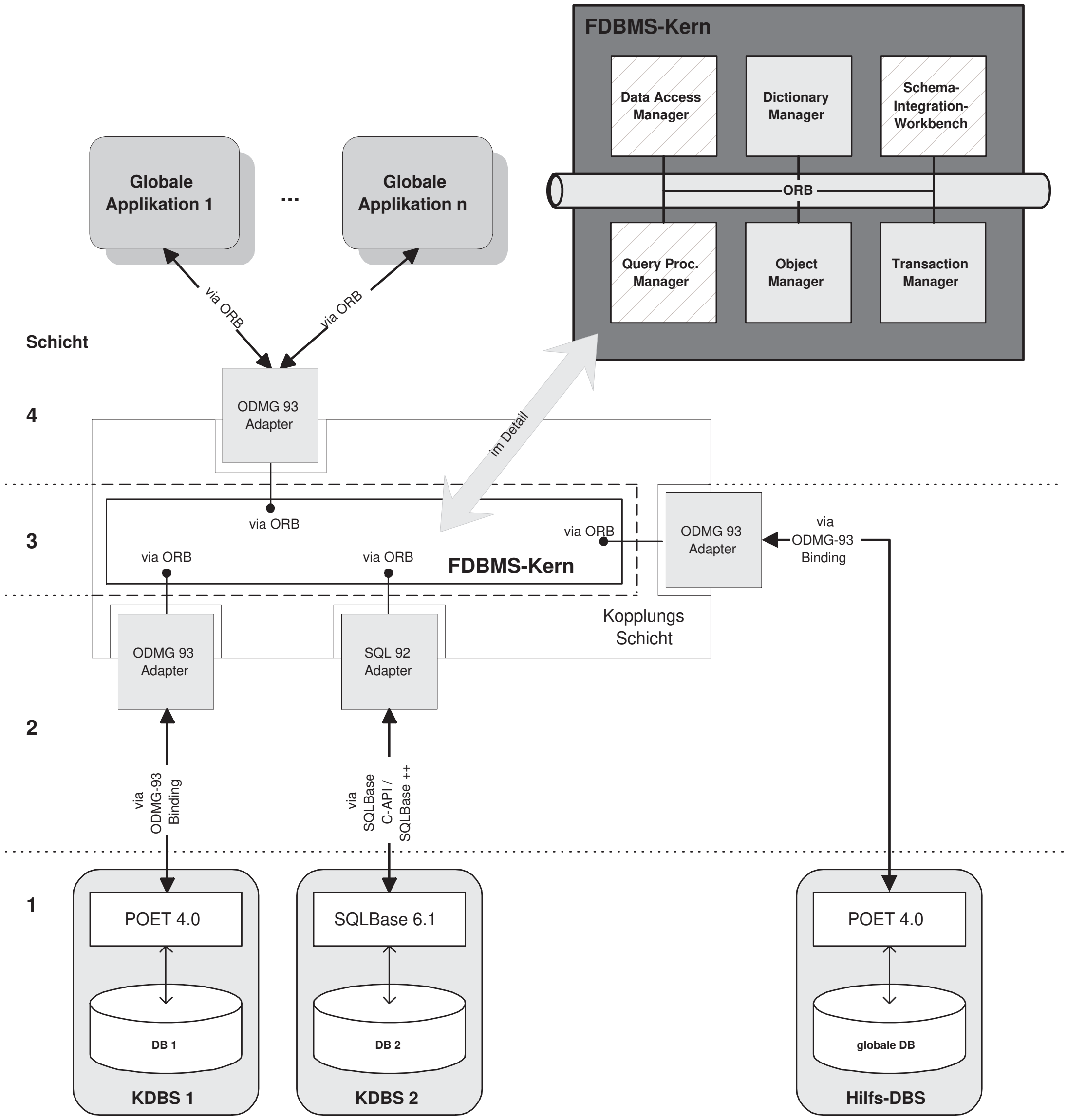

Abbildung 1: Architektur des zu entwickelnden FDBS.

Anbieter von ODBMS zumeist eine Hierarchie von Klassen zur Verfügung, mit deren Hilfe ein Dictionary verwaltet werden kann. Diese werden häufig als Meta-Klassen bezeichnet (dies ist z.B. der Fall bei POET [POE96] und O_2 [BDK92]). Im Projekt wird sich zeigen, ob sich das von POET zur Verfügung gestellte Meta-Klassen Konzept im Dictionary-Manager verwenden läßt. Der Object-Manager verwaltet für alle dem FDBS zugänglichen Datenbankobjekte eindeutige Object-Identifikatoren (OID). In [SS95, EK91] werden zwei Dimensionen von Objekt-Identität unterschieden. Zum einen die Umgebung innerhalb derer eine OID eindeutig ist, zum anderen der zeitliche Rahmen innerhalb dessen eine OID unverändert bleibt.

Der Object-Manager soll allen dem FDBS zugänglichen Datenbank-Objekten eindeutige und dauerhafte OIDs über die Lebenszeit des FDBS zuordnen. Hierzu verwaltet er die Lokalisation der DB-Objekte und eine Abbildung zwischen den Objekt-Identifikatoren der lokalen DBS und den OIDs. Als Objekt-Identifikatoren seien die ein Datenbank-Objekt, im Kontext eines Komponenten-DBS, eindeutig referenzierenden Eigenschaften bezeichnet. Ein wesentliches Problem dieser Abbildung besteht in der Autonomie der an der Föderation beteiligten Komponenten-DBS. Um die Aktualität der Abbildung zu gewährleisten, müssen bestimmte Ereignisse innerhalb der Komponenten-DBS dem Object-Manager unmittelbar mitgeteilt werden. Unter diese kritischen Ereignisse fallen das Neuanlegen oder Löschen eines Objekts bzw. das Verändern des Objekt-Identifikators eines lokalen Datenbank-Objekts. Wie in [SS95, EK91] herausgestellt wird, ist eine effiziente Realisierung eines Object-Managers nur bei einer Reduzierung der Autonomie der Komponenten-DBS möglich. Im konkreten Fall des zu integrierenden DBS SQLBase sind Trigger die Methode der Wahl. Diese Methode ist im Gegensatz zu einer Polling-Technik, bei welcher der Inhalt des Komponenten-DBS in zyklischen Abständen auf kritische Ereignisse hin überprüft wird, wesentlich effizienter. Außerdem erhöht sich hierdurch die Aktualität der von dem Object-Manager verwalteten Abbildung. Hierdurch wird jedoch die Entwurfsautonomie des lokalen Systems einschränkt. Das ODBMS POET bietet in diesem Kontext die Möglichkeit *Watches*, eine Art Trigger, zu definieren. Dabei spezifiziert eine Client-Anwendung das genaue Ereignis (z.B. Neuanlegen, Löschen oder Verändern eines Objektes), über welches es informiert werden möchte und übergibt zusätzlich eine Referenz auf eine *Memberfunktion*, die das Auftreten des Ereignisses behandeln soll. Die zuletzt betrachteten Mechanismen sollen bei der Realisierung des Object-Managers Verwendung finden.

Der Transaction-Manager bildet Transaktionen der Anwendungen, die sich auf das FDBS stützen, auf Subtransaktionen der involvierten Komponenten-DBS ab. Der Data Access Manager realisiert die Aufgabe der Zugriffskontrolle. Der Query Processing Manager verarbeitet globale Anfragen in OQL, zerlegt diese in Teilanfragen für die involvierten Komponenten-DBS und optimiert die Bearbeitungsreihenfolge. Eine Schema-Integration-Workbench unterstützt die Datenbankadministratoren bei der Aufgabe der Schemaintegration.

Die Komponenten Data Access Manager, Query Processing Manager und Schema-Integration-Workbench sollen in der ersten Phase nicht realisiert werden. Sie sind jedoch in der konkreten FDBS-Architektur aufgeführt, um die Vollständigkeit zu gewährleisten. An dieser Stelle könnte sich die Frage stellen, wie ein Retrieval von Datenbank-Objekten ohne das Vorhandensein einer Datenanfragesprache möglich ist. In einem ODMG-93-konformen Datenbanksystem sind all die Objekte zugreifbar, welche direkt oder indirekt (über Objektreferenzen) mit einem *Named Object* verbunden sind. Ein Named Object ist dabei ein persistentes Objekt, welches zu einem Namen assoziiert ist (diese Assoziationen werden im Schema verwaltet). Diese Namen stellen somit den Zugangspunkt zu den persistenten Objekten in der Datenbank dar und werden deshalb auch häufig als die *Wurzeln der Persistenz* bezeichnet [Cat96]. Sie sind in etwa vergleichbar mit den Tabellennamen in einem relationalen DBS. Named Objects können dabei Instanzen von anwendungsdefinierten Klassen aber auch Kollektionen von Objekten sein. Ein besonderer Fall von Named Objects stellen die Extensionen einer Klasse dar. Hierunter versteht man die Menge aller Instanzen einer Klasse. Ausgehend von Named Objects ist nun ein navigierender Zugriff auf alle Objekte der Datenbank möglich.

6 Zusammenfassung der Projektziele

Im vorgestellten Projekt soll eine konkrete Architektur für ein föderiertes Datenbanksystem entwickelt werden, welche sich aus der in [SL90] dargestellten Referenz-Architektur ableiten läßt und sich auf die Standards ODMG-93 sowie CORBA stützt. Einige Komponenten dieser FDBS-Architektur sollen dabei prototypisch implementiert werden, um eine genauere Bewertung des gewählten Ansatzes zu ermöglichen. Die System- bzw. Entwicklungsumgebung des unter Windows NT zu realisierenden Prototypen umfaßt den ORB Chorus Cool [JJC95], das ODMG-93-konforme objektorientierte Datenbanksystem POET 4.0 [POE96], das relationales Datenbanksystem SQLBase 6.1 [Cen96], das E/R-Entwurfswerkzeug ERwin [Log95], das objektorientierte Entwurfswerkzeug Rational Rose [Rat95] sowie Visual C++ [Mic95]. Die Abstützung auf die Standards CORBA und ODMG-93 soll es erlauben, den Kern des FDBS auch unter Solaris mit dem ORB Chorus Cool [JJC95] und der ODMG-93-konformen Datenbank O_2 [BDK92] zu verwenden.[1]

Hauptziel ist die abschließende Bewertung des gewählten Ansatzes und der bei der Implementierung gewonnenen Erfahrungen. Ein Ergebnis wird sein, wie sich ein ODMG-93-konformes Datenbanksystem, welches von den ODMG-Mitgliedern als eine Erweiterung des “OMA Persistent Object Service” gesehen wird [OHE96], in eine ORB Umgebung einfügt. Zudem wird die Performance des angestrebten Systems ein wichtiger Aspekt der Untersuchung sein. Die Eignung von CORBA als Integrationsansatz in einem FDBS-Projekt und die Eignung von ODMG-93 als kanonisches Datenmodell eines föderierten Datenbanksystems stellen die beiden Hauptbetrachtungspunkte dar.

[1]Die angegebenen Produktnamen sind eingetragene Warenzeichen.

Literatur

[BDK92] F. Bancilhon, C. Delobel, und P. Kanellakis. *Building an Object-Oriented Database System: The Story of O_2*. Morgan Kaufman, 1992.

[Cat96] R. Cattell, Hrsg. *The Object Database Standard: ODMG-93, Release 1.2.* Morgan Kaufman, 1996.

[Cen96] Centura Software Corporation. *SQLBase Language Reference*, 1996.

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

[HFBK94] G. Huck, P. Fankhauser, R. Busse, und W. Klas. IRO-DB An Object-Oriented Approach towards Federated and Interoperable DBMS. In *Proc. International Workshop on Advances in Databases and Information Systems (ADBIS'94)*, Seiten 178–186, Moscow, Mai 1994.

[Hil95] E. Hildebrandt. Eignungsuntersuchung des ODMG-93 Objektmodells als Objektmodell einer Föderierungsschicht zur Integration heterogener, autonomer Datenbanksysteme. Diplomarbeit, Fakultät für Informatik, Universität Magdeburg, 1995.

[Hol96] T. Holzer. Objektorientierte Standardsoftware? *Objekt Spektrum*, Seiten 68–71, Jan/Feb 1996.

[HP96] U. Hohenstein und V. Plesser. Semantic Enrichment: A First Step to Provide Database Interoperability. In S. Conrad, M. Höding, S. Janssen, und G. Saake, Hrsg., *Kurzfassungen des Workshops "Föderierte Datenbanken"*, Seiten 3–17, Magdeburg, April 1996. Bericht 96–01, Institut für Technische Informationssysteme, Universität Magdeburg.

[HTJ+95] M. Höding, C. Türker, S. Janssen, K.-U. Sattler, S. Conrad, G. Saake, und I. Schmitt. Föderierte Datenbanksysteme - Grundlagen und Ziele des Projektes SIGMA_{FDB}. Preprint Nr. 12, Fakultät für Informatik, Universität Magdeburg, Dezember 1995.

[JJC95] C. Jacquemot, P. Strarup Jensen, und S. Carrez. CHORUS/COOL: CHORUS Object Oriented Technology. In *Object-Based Parallel and Distributed Computation (OBPDC '95)*, Band 1107 der Reihe *Lecture Notes in Computer Science*, Seiten 187–204. Springer-Verlag, 1995.

[Kro93] P. Kroha. *Objects and Databases.* McGraw-Hill, 1993.

[Log95] Logic Works Inc., Princeton. *Erwin User's Guide*, 1995.

[Mic95] Microsoft Corporation. *Visual C++ Online Documentation System*, 1995.

[OHE96] R. Orfali, D. Harkey, und J. Edwards. *The Essential Distributed Object Survival Guide.* Wiley, 1996.

[POE96] POET Software, Hamburg. *POET 4.0 Programmer's Guide*, 1996.

[Rah94] E. Rahm. *Mehrrechner-Datenbanksysteme: Grundlagen der verteilten und parallelen Datenbankverarbeitung.* Addison-Wesley, Bonn, 1994.

[Rat95] Rational Software Corporation, Santa Clara, CA. *Rational Rose User's Guide*, 1995.

[RS95] E. Radeke und M.H. Scholl. Functionality for object migration among distributed, heterogeneous, autonomous database sytems. In *Proc. 5th International Workshop on Research Issues in Data Engineering: Distributed Object Management (RIDE-DOM '95)*, Seiten 58–66, Taipei, Taiwan, März 1995. IEEE Computer Society Press.

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

[SS95] I. Schmitt und G. Saake. Managing Object Identity in Federated Database Systems. In M. Papazoglou, Hrsg., *Proc. 14th Int. Conf. on Object-Oriented and Entity-Relationship Modeling (OOER'95)*, Band 1021 der Reihe *Lecture Notes in Computer Science*, Seiten 400–411, Gold Coast, Australia, 1995. Springer-Verlag.