Die Nutzung der Erweiterungsmöglichkeiten von ORDBVS für Client/Server-basierte Anwendungssysteme

Henrik Loeser
Universität Kaiserslautern, FB Informatik, AG DBIS
Postfach 3049, 67653 Kaiserslautern
E-Mail: loeser@informatik.uni-kl.de
WWW: http://www.uni-kl.de/AG-Haerder/Loeser/

Zusammenfassung: Objekt-relationale Datenbankverwaltungssysteme (ORDBVS), die seit wenigen Jahren kommerziell angeboten werden und sich noch auf dem Weg zur Marktreife befinden, bieten als Ergänzung zu rein relationalen DBVS neben ihren objektorientierten Konzepten auch Möglichkeiten der Erweiterung des DBS ("Extensibility"). Letzteres bedeutet, daß neue Datentypen, Anwendungslogik und eigene Indexstrukturen in den Datenbank-Server integriert sowie externe Datenquellen eingebunden werden können. Für den Entwurf von DB-basierten Anwendungssystemen ergeben sich hierdurch zahlreiche neue Optionen. In diesem Beitrag geben wir einen kurzen Überblick über die verschiedenen, derzeit angebotenen Erweiterungsmöglichkeiten, diskutieren, soweit in der Kürze möglich, zu überdenkende Aspekte beim Einsatz dieser Möglichkeiten in Client/Server-basierten Anwendungssystemen und stellen ein konkretes Anwendungsbeispiel für den Einsatz der Erweiterungsmöglichkeiten vor.

1. Einleitung

Objekt-relationale Datenbankverwaltungssysteme (ORDBVS, [Cha96, Kim96, Sto96]), die seit wenigen Jahren kommerziell angeboten werden und sich noch auf dem Weg zur Marktreife befinden, bieten als Ergänzung zu rein relationalen DBVS neben ihren objektorientierten Konzepten auch Möglichkeiten der Erweiterung des DBS ("Extensibility"). Hierunter versteht man die Möglichkeit, das ORDBS um benutzerdefinierte Datentypen ("user defined types", UDTs), in einer 3GL (third generation language) kodierte Funktionen ("user defined functions", UDFs) sowie neue Indexstrukturen so zu ergänzen, daß diese neuen Typen, Funktionen und Indexstrukturen vom Benutzer bzgl. der Sprachintegration und der Optimiererunterstützung wie die eingebauten, d. h. die bereits vom (nicht-erweiterten) DBVS bereitgestellten, verwendet werden können. Faßt man diese drei Arten zusätzlicher Funktionalität anwendungs(bereichs)bezogen zusammen, z. B. für die Speicherung und Bearbeitung von Bildern, so spricht man, je nach DBVS-Hersteller, von DataBlade (Informix), Extender (IBM) oder Cartridge (Oracle). Als Ergänzung dieser Konzepte können über sogenannte "Table Functions" (IBM) bzw. über das "Virtual Table Interface" (Informix) externe Datenquellen in Form einer Tabelle zur Verfügung gestellt werden, auf die dann mit Hilfe von SQL zugegriffen werden kann.
Diente der Datenbank-Server in zwei- und mehrstufigen DB-basierten Anwendungssystemen bisher nur zur reinen Datenhaltung, so können nun basierend auf seiner Erweiterungsfähigkeit weitere Aufgaben an ihn deligiert bzw. eine engere Verknüpfung von Applikations- und DB-Server erreicht werden. Mit diesen neuen Optionen hinsichtlich des Entwurfs von Client/Server-basierten Anwendungssystemen sind aber auch neue Probleme und Fragen verbunden, die es zu lösen bzw. zu beantworten gilt.
Im folgenden geben wir zunächst einen sehr knappen Überblick über die von ORDBVS angebotenen Erweiterungsmöglichkeiten. Im Anschluß daran (Kapitel 3) diskutieren wir, soweit in der gebotenen Kürze möglich, die für ihren richtigen Einsatz im Rahmen von Client/Server-basierten Anwendungssystemen (AS) zu bedenkenden Aspekte und stellen als Beispiel für den Einsatz der ganzen Breite der "Extensibility" ein Projekt vor. Den Abschluß bilden eine Zusammenfassung sowie ein Ausblick auf weitere Arbeiten.

2. Erweiterungsfähigkeit von ORDBVS

Neben ihren zusätzlichen objektorientierten Konzepten zeichnen sich ORDBVS insbesondere durch ihre Erweiterungsfähigkeit aus. Hierunter versteht man die Möglichkeit zur Definition von eigenen Datentypen, der Erweiterung des DB-Servers um Anwendungslogik, die Integration neuer Indexstrukturen in die Abläufe des DB-Servers sowie die Einbindung externer Datenquellen. Im Gegensatz zu bisher durchaus schon vorhandenen Ansätzen im Bereich der objektorientierten DBVS ist nun die vollständige Einbindung in den Rahmen der Anfragesprache SQL gegeben, d. h., am DB-Server vorgenommene Erweiterungen stehen in SQL bzw. intern genauso wie die systemseitig vorhandenen Datentypen und Indexstrukturen zur Verfügung. Zusätzlich können jedoch noch in einer SQL-Anweisung UDFs aufgerufen werden. Im folgenden gehen wir kurz auf die für das Thema dieses Beitrags relevanten Aspekte der Erweiterbarkeit von ORDBVS ein.

Benutzerdefinierte Datentypen

Bei den benutzerdefinierten Datentypen unterscheidet man zwischen verschiedenen Arten: distinct types, opaque types und row types. Während erstere nur eine Umbenennung eingebauter Datentypen bedeuten, um z. B. mehr Semantik und Typisierung zu erreichen, sind opaque types dem DBVS nur als "Black Box" bekannt und werden intern über sog. "support functions" angesprochen. Dies bedeutet, zum lesenden und schreibenden Zugriff werden vom Entwickler dieses neuen Typs bereitgestellte Funktionen verwendet, die u. a. Daten von der ASCII-Darstellung in das interne Format konvertieren und umgekehrt.

Basierend auf den eingebauten und den bisher erzeugten benutzerdefinierten Datentypen können nun neue row types bzw. Tabellentypen erzeugt werden. Auf ihnen aufbauend können dann Tabellen definiert werden, d. h., eine auf diese Art definierte Tabelle enthält genau die Attribute des zugrundeliegende Tabellentyps.

Benutzerdefinierte Funktionen

Anders als bei bisherigen *Stored Procedures* können nun Funktionen in einer 3GL wie C, C++ oder Java entwickelt werden, wobei je nach Hersteller verschiedene objektorientierte Konzepte unterstützt werden. Das heißt, die Bindung einer Funktion, bzw. dann Methode, an einen Typ, das Überladen von Funktionen sowie spätes Binden sind möglich. Funktionen können im Rahmen einer SQL-Anweisung aufgerufen werden, es sei denn, sie sind explizit nur für den internen Gebrauch, d. h. für den Aufruf von anderen Funktionen aus vorgesehen.

Durch ihre Art der Einbindung in eine SQL-Anweisung werden drei verschiedene Typen unterschieden: skalare Funktionen, Aggregatfunktionen und Tabellenfunktionen. Während skalare Funktionen bei jedem Funktionsaufruf einen Wert zurückliefern und von jedem DBVS unterstützt werden, findet sich für Aggregatfunktionen, die basierend auf allen qualifizierten Werten einer Spalte *ein* Ergebnis zurückgeben, derzeit keine Unterstützung seitens der kommerziellen Anbieter. Während grundlegende Voraussetzungen, wie z. B. das Anlegen eines den Funktionsaufruf überdauernden privaten Speichers zum Ablegen von Zwischenresultaten (z. B. Anzahl bisheriger Werte und die Summe für ein eigenes AVG), vorhanden sind, fehlt bisher die Einbindung in die interne Anfrageausführung. Tabellenfunktionen, die, wie ihr Name schon andeutet, Tabellen als Ergebnis liefern, stellen einen Sonderfall dar. Auf sie werden wir im Rahmen der Diskussion des Zugriffs auf externe Daten (s. u.) noch genauer eingehen.

Benutzerdefinierte Indexstrukturen

Neben der Definition eigener Datentypen und Funktionen bieten ORDBVS auch die Möglichkeit zur Integration eigener Indexstrukturen in den DB-Server. So ist es z. B. möglich, für VITA-Daten (Video, Image, Text, Audio) spezielle neue Indextypen wie R-Baum, D-Baum etc. zu integrieren und zu nutzen.

Zugriff auf externe Daten

Neben der Erweiterung des DB-Servers bieten ORDBVS zudem die Möglichkeit des Zugriffs auf externe Daten. Diese sind danngenauso wie die intern verwalteten Daten vom Anwender über SQL ansprechbar.

Je nach Hersteller spricht man dabei von Tabellenfunktion (IBM DB2) oder einer virtuellen Tabelle (Informix). Der Grundmechanismus basiert aber gemeinsam auf einer (IBM) oder mehreren Funktionen (Informix), die bei Aufruf eine Tabelle, die aus den externen Daten erzeugt werden muß, als Ergebnis liefern. Im Falle des Einsatzes von nur einer Funktion muß diese die verschiedenen Aktionen (öffnen, lesen, schließen) behandeln, während dies im anderen Fall von je einer eigenen Funktion gehandhabt wird. Die Hersteller unterscheiden sich in der Mächtigkeit der bereitgestellten Möglichkeiten. So dürfen bei IBM z. B. innerhalb einer Tabellenfunktion keine SQL-Anfragen gestellt werden und es ist auch nur lesender Zugriff auf die erzeugte Tabelle möglich.

3. Nutzung der Erweiterungsfähigkeit

Nachdem wir nun kurz die Grundkonzepte der Erweiterungsfähigkeit von ORDBS betrachtet haben, wollen wir nun ihren Nutzen in Client/Server-basierten Anwendungssystemen diskutieren. Hierzu wollen wir zunächst allgemeine Aspekte erörtern, die für eine Entscheidung, ob und wie UDFs genutzt werden sollen, wichtig sind. Anschließend betrachten wir ein konkretes Anwendungsbeispiel für die Integration von externe Daten in Kombination mit UDFs, UDTs sowie eigenen Indexstrukturen: die Verwaltung von Web-Daten.

3.1 Die Nutzung von UDFs

Wenngleich ORDBVS in vielen Einsatzgebieten von DBVS zunächst geringe Vorteile bringen, die im allgemeinen von besser auf die jeweilige Anwendung abstimmbaren Datentypen und dem selektiven Einbringen von Anwendungslogik in den Server-Kern mit Hilfe benutzerdefinierter Funktionen herrühren [HLZ97], so sind aber auch wegen des aufgrund der Erweiterungsmöglichkeiten komplexer gewordenen Server-Kerns leichte Geschwindigkeitseinbußen zu verzeichnen. So stellte auch bereits [Sto96] fest: "... object-relational engines have typically not been optimized for transaction processing. Expect TPC-C to run slower on object-relational engines." Dies ist auch ein Aspekt, der auch vom in [CD+97] vorgeschlagenen *BUCKY object-relational Benchmark* behandelt wird. In ihm wird eine sowohl relational wie auch objekt-relational modellierbare Mini(hochschul)welt mit entsprechenden Testsituationen herangezogen, um Geschwindigkeitsunterschiede von relationalen zu objekt-relationalen Systemen bzw. zwischen den Modellierungsalternativen auf einem System zu messen. Neben anderen Ergebnissen konnte hiermit nachgewiesen werden, daß ORDBVS durch den gezielten Einsatz von UDFs die Verarbeitungsgeschwindigkeit steigern können.

Durch den Einsatz von UDFs besteht die Möglichkeit, setzt man eine 3-stufige Architektur voraus, Anwendungslogik vom Applikations-Server, wie in Abb. 1 a) und b) dargestellt, zunächst statisch in den DB-Server zu verlagern. Eine dynamische, lastabhängige Verlagerung der Anwendungslogik auf die beiden Server-Ebenen (Abb. 1c) oder sogar unter Einbeziehung der Client-Schicht (Abb. 1d) ist erst dann möglich, wenn benötigte (System-)Bibliotheken und geeignete Mechanismen auf allen Ebenen zur Verfügung stehen bzw. die ORDBVS weitere Programmiersprachen oder andere Arten der UDF-Einbindung unterstützen (siehe auch Diskussion unten). Interessant erscheinen in diesem Zusammenhang die Integration einer *Java Virtual Machine* in den DB-Server und auch die Integration der UDFs über Mechanismen von CORBA.

Möchte man nun die Möglichkeiten der Erweiterungsfähigkeit nutzen, so ist es nun nicht einfach damit getan, alle sich nun anbietenden Möglichkeiten heranzuziehen. Vielmehr besteht durch die größere Zahl der Modellierungsoptionen nun viel eher die Gefahr, das Gesamtsystem eher schlecht als gut zu entwerfen. Für den Einsatz von UDFs sind daher vom Systementwickler für die Entscheidung, welche und wieviel Funktionalität wann wohin verlagert werden soll, u. a. folgende Aspekte zu bedenken:

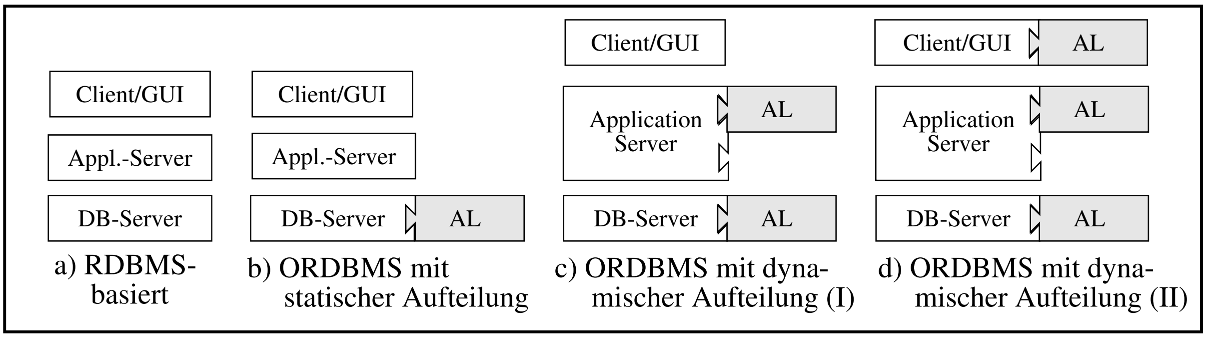

Abb. 1: Wandel von Client/Server-basierten DB-Anwendungssystemarchitekturen

- *Art der Einbindung von UDFs* [SB97]: Bei der Einbindung von in einer 3GL erstellten UDFs kann man derzeit drei verschiedene Ansätze unterscheiden, die unterschiedlich starken Einfluß auf das Laufzeitverhalten haben. Die von der Geschwindigkeit beste aber bzgl. der Fehlertoleranz schlechteste Lösung ist die Abarbeitung der UDFs im eigentlichen Server-Prozeß. Verursacht die UDF einen Fehler, stürzt evtl. das gesamte ORDBVS ab. Dafür werden aber die geringsten Kosten beim Funktionsaufruf verursacht. Die bzgl. Sicherheit und Geschwindigkeit ausgewogenste Lösung ist die Abarbeitung von UDFs in einem separaten Prozeß (“fenced”). Stürzt dieser Prozeß ab, so kann das eigentliche System weiterarbeiten. Die von den Laufzeitkosten her teuerste Lösung ist die Einbindung der UDFs über CORBA-Mechanismen. Neben den Kosten eines Prozeßwechsels entsteht zusätzlicher Aufwand durch ein komplexeres Protokoll.

- *Rechenzeitintensivität der UDFs*: Ein Aspekt, der starken Einfluß auf die Zahl der gleichzeitigen DB-Benutzer und die Antwortzeiten hat, ist die Rechenzeitintensität einer UDF. Werden nur (von der Rechenzeit her) einfache Aufgaben durchgeführt, die allen höheren Schichten zugute kommen, z. B. Konvertierungen, so wird das Risiko eines “Fat Servers” vermieden. Werden dagegen rechenzeitintensive Aufgaben auf den DB-Server verlagert, so besteht schnell die Gefahr, daß dieser zu einem Engpaß wird und die Leistung des Gesamtsystems drastisch leidet.
- *Verhältnis “verarbeitete/in Applikation benötigte” Daten*: Werden von einer UDF viele Daten “angefaßt” und durch die Auswertung die Datenmenge deutlich reduziert, so ist der Einsatz von UDFs lohnenswert, da durch die Abarbeitung der Auswerteroutinen auf dem DB-Server hohe Datentransportkosten zwischen DB- und Applikations-Server vermieden werden können. Andererseits sind aber die bzgl. der Rechenzeitintensität angestellten Überlegungen auch nicht zu vernachlässigen.
- *Komplexität und Größe der Datensätze*: Die Komplexität der Daten, d. h. der Grad der internen Vernetzung mit anderen Daten, und auch die Größe eines Datensatzes sind in die Überlegungen bzgl. der Verlagerung von Funktionalität einzubeziehen. Können durch den Einsatz von UDFs die Komplexität und Datensatzgröße verringert werden, lassen sich sowohl Datentransportkosten als auch eventuelle Zusatzanfragen zum Einladen abhängiger Datensätze vermeiden.
- *Systemkennzahlen*: Architektur und Leistungsfähigkeit der eingesetzten Hard- und Software-Komponenten, die Zahl der gleichzeitigen Benutzer und auch die Anforderungen bzgl. Durchsatz, Antwortzeit etc. haben Einfluß auf die Entscheidung, welche und wieviel Anwendungslogik in Form von UDFs zum DB-Server zu delegieren ist. Ist z. B. die DB-Server-Maschine bedeutend leistungsfähiger als die des Appl.-Servers, so kann auf erstere u. U. deutlich mehr Anwendungslogik verlagert werden.

Ausgehend von diesen Fragestellungen ist dann zu entscheiden, wie das Mittel der UDFs innerhalb des Anwendungssystems genutzt werden soll. Da die Zusammenhänge dieser Einzelfaktoren recht komplexer Natur sind und sicherlich in Zukunft noch mehr Systemarchitekten beschäftigen werden, wollen wir schon aus Platzgründen nicht weiter darauf eingehen, sondern uns zum Abschluß einem konkreten Anwendungsgebiet für alle der angesprochenen Erweiterungsmöglichkeiten zuwenden.

3.2 Beispiel: Eine integrierte Web-DB

Neben der Verbesserung Client/Server-basierter Anwendungssysteme durch eine Verlagerung von Anwendungslogik vom Applikations- zum DB-Server können solche Systeme auch eine verbesserte Anpassung des DB-Servers an die Anwendung und ihre Daten optimiert werden. Als Beispiel wollen wir hier das WWW betrachten. Werden (OR)DBS bisher nur zur Bereitstellung von Daten zum Aufbau dynamischer HTML-Seiten verwendet [Loe97], so können diese nun durch die gezielte Bereitstellung der benötigten Datentypen auch zur Verwaltung aller Daten, d. h. auch der HTML-Seiten und Bilder dienen [Inf97]. Dies führt zu einer verbesserten, weil leichteren Administration sowie einer in sich konsistenten “Web-Site”.

Betrachtet man einen Web-Server mit seiner Infrastruktur, so stellt man fest, daß in der Regel HTML- und Grafikdateien im Dateisystem gehalten und von verschiedenen, unabhängigen Personen oder Gruppen erstellt werden, ein oder mehrere DBS für die Verwaltung zusätzlicher Daten eingesetzt werden und daß eine Suchmaschine, falls angeboten, noch einmal eine zusätzliche DB anlegt. Während bei Letzterem die Aktualisierung meist zu festgelegten Zeitpunkten erfolgt, wobei aber meist aus technischen Gründen nicht alle Dateien erfaßt werden und ein Schutz vor einem Durchsuchen bestimmter Dateien meist nicht regelbar ist, führt ersteres bedingt durch das Zusammenwirken mehrerer Personen bzw. Gruppen zwangsläufig zu Inkonsistenzen.

Durch die nahtlose Integration aller Daten(typen) mit Hilfe der Möglichkeiten der Erweiterung eines

ORDBS lassen sich diese Probleme beseitigen. Unter Nutzung der Möglichkeiten der Integration externer Daten können die Web-Dateien aus dem Dateisystem, angereichert durch zusätzliche Attribute, z. B. die Vereinheitlichung von Dateisuffixen zu Typen wie HTML oder Bild, für den Zugriff über SQL bereitgestellt werden. Werden im ORDBS die entsprechenden Datentypen mit gleicher Struktur wie die externen Daten angeboten, so ist zudem mit einer einzigen SQL-Anweisung eine Übertragung vom Datei- in das DB-System und umgekehrt möglich und das ORDBS kann als Repository für die Web-Dateien dienen. Durch die Integration von geeigneter Suchfunktionalität mittels UDFs und eigener Indextypen steht in Verbindung mit den Möglichkeiten von SQL und vorhandenen Attributen ein mächtiges Anfragepotential zur Verfügung, das weit über das gängiger (Web-Site-) Suchmaschinen hinausgeht. Aus Platzgründen können wir leider nicht genauer auf die Details und die Architektur eingehen, sondern müssen auf noch folgende, zugehörige Publikationen verweisen.

4. Zusammenfassung und Ausblick

In diesem Beitrag haben wir die Nutzung der Erweiterungsmöglichkeiten von ORDBVS für Client/Server-basierte Anwendungssysteme, insbesondere die vor ihrem Einsatz zu bedenkenden Aspekte diskutiert. Möchte man sie nutzen, so steigt die Komplexität des Anwendungssystementwurf drastisch. Auf der anderen Seite bieten sich jedoch auch ganz neue Entwurfs- und Optimierungsmöglichkeiten. Da ORDBVS noch relativ neu sind, fehlen bislang neben entsprechenden Entwurfs- und Modellierungsregeln auch noch ausgereifte unterstützende Werkzeuge, sowohl für die Schemamodellierung [Gri98] als auch für die eigentliche Implementierung. Zudem fehlt bislang, bedingt durch den sich noch langsam formierenden Begriff der ORDBVS, ein Verständnis für die Nutzungsmöglichkeiten von ORDBVS und die sich daraus ergebenden Vor- und Nachteile.

Ziel der sich noch am Anfang befindenden Forschungsprojekte muß es daher sein, zum einen nach geeigneten Regeln bzw. Kriterien für den DB-Schemaentwurf als auch die (Anwendungs-)Systemmodellierung unter Nutzung der Erweiterungsmöglichkeiten zu suchen, zum anderen aber auch für eine geeignete (Software-)Infrastruktur zu sorgen, die den Entwurf und die Implementierung, nach Möglichkeit DB-herstellerunabhängig, unterstützt. Dann ist auch ein Einsatz der ORDBVS über die bislang wenigen, ausgezeichnete Anwendungsgebiete [Hei98] hinaus möglich. Ihre allgemeine Tauglichkeit haben sie noch zu zeigen. Hierzu ist jedoch ein genaueres Verständnis ihrer Möglichkeiten sowie der damit zusammenhängenden Implikationen erforderlich. So meint denn auch auch [Gr98]: “Are ORDBMSes the Next Great Wave - or are they so complex they’re unusable?”

5. Literatur

CD+97 Michael J. Carey, David J. DeWitt, Jeffrey F. Naughton, Mohammad Asgarian, Paul Brown, Johannes Gehrke, Dhaval Shah: *The BUCKY Object-Relational Benchmark (Experience Paper)*, Proceedings ACM SIGMOD Conference 1997, Tucson, Arizona, S. 135-146.

Cha96 Don Chamberlin: *Using the New DB2: IBM‘s Object-Relational Database System*, Morgan Kaufman, 1996

Gri98 Seth Grimes: *Modeling Object-Relational Databases*, DBMS Magazine, April 1998

Hei98 Heinrich, Johannes: *Studie über das Einsatzpotential Objekt-Relationaler DBMS und Evaluation kommerzieller Produkte*, Diplomarbeit, FB Informatik, Universität Kaiserslautern, Februar 1998.

HLZ97 Härder, T., Loeser, H., Zhang, N.: *Supporting Adaptable Technical Information Systems in Heterogeneous Environments - Using WWW and ORDBMS* -, in: Proc. 8th Int. Workshop on Database and Expert Systems Applications (DEXA’97), Toulouse, Sept. 1997, S. 295-303.

Inf97 *Informix Web Integration Option for Informix Dynamic Server*, Informix White Paper, Informix Software Inc., 1997.

Kim96 Won Kim: *Object-Relational - The unification of object and relational database technology*, UniSQL White Paper, 1996

Loe97 Loeser, Henrik: *Datenbankanbindung an das WWW - Techniken, Tools und Trends*, in: Tagungsband der GI-Fachtagung “Datenbanksysteme in Büro, Technik und Wissenschaft” (BTW’97), K. R. Dittrich, A. Geppert (Hrsg.), Informatik aktuell, Ulm, März 1997, Springer-Verlag, S. 83-99.

SB97 Michael Stonebraker, Paul Brown: *Objects in Middleware: How Bad Can It Be*, Informix White Paper, Informix Inc., 1997

Sto96 Michael Stonebraker: *Object-Relational DBMSs - The Next Great Wave*, Morgan Kaufman, 1996