GESIS

InformationsZentrum Sozialwissenschaften

IZ-Arbeitsbericht Nr. 5

Relationale Datenbanksysteme im Vergleich: Eine Zwischenbilanz

Peter Mutschke

Dezember 1995

InformationsZentrum
Sozialwissenschaften

Lennéstraße 30
D-53113 Bonn
Tel.: 0228/2281-135
Fax.: 0228/2281-120
email: mutschke@bonn.iz-soz.de
Internet: http://www.social-science-gesis.de

ISSN: 1431-6943

Herausgeber: Informationszentrum Sozialwissenschaften der Arbeitsgemeinschaft Sozialwissenschaftlicher Institute e.V. (ASI)

Druck u. Vertrieb: Informationszentrum Sozialwissenschaften, Bonn
Printed in Germany

Das IZ ist Mitglied der Gesellschaft Sozialwissenschaftlicher Infrastruktureinrichtungen e.V. (GESIS), einer Einrichtung der Wissenschaftsgemeinschaft Blaue Liste (WBL)

Inhalt

Abstract:

Die Wahl einer bestimmte Datenbanksoftware ist eine Entscheidung von grundsätzlicher strategischer Bedeutung. Sowohl die Datenbankmaschine selbst als auch die Entwicklungswerkzeuge bilden die technologische Plattform für die Umsetzung eines integrierten, informationswissenschaftlich elaborierten Gesamtkonzepts für den Umgang mit allen relevanten Unternehmensdaten. Vor diesem Hintergrund werden in der vorliegenden Studie die gängigsten relationalen Datenbanksysteme für größere UNIX-Plattformen im Hinblick auf die informationstechnologischen Anforderungen des Informationszentrum Sozialwissenschaften untersucht und bewertet.

1 Anforderungen an eine neue Datenbanksoftware im IZ

1.1 Vorbemerkungen

Die vorliegende Studie geht davon aus, daß es das Ziel des IZ ist, ein integriertes, informationswissenschaftlich elaboriertes Gesamtkonzept für seine Informationsdienstleistungen und den Umgang mit *allen* relevanten Unternehmensdaten bereitzustellen. Da die neu einzuführende Datenbanksoftware die technologische Plattform zur Erreichung dieses Zieles sein wird, ist die Wahl für ein bestimmtes Datenbanksystem daher eine Entscheidung von grundsätzlicher strategischer Bedeutung.

Mit dieser Zielperspektive stellen sich die Anforderungen an eine neue - relationale - Datenbanksoftware anders als an ein reines Datenspeicherungssystem (die bisherige Nutzungsphilosophie): Die Entscheidung für ein bestimmtes Produkt kann sich nicht an Funktionalität und Performance des Database Management Systems (DBMS) allein orientieren. Entscheidungsrelevant sind ebenso die zusätzlichen, sich um den Datenbankkern herum gruppierenden Komponenten und Werkzeuge, die eine integrierte Anwendungsentwicklung ermöglichen oder die Voraussetzung bilden für die Umsetzung eines unternehmensweiten integrierten Client/Server-Konzepts. Dies gilt um so mehr als sich die Datenbanksysteme hinsichtlich der Kernfunktionalität weitgehend ähneln.

Eine zentrale Grundanforderung eines sich an den Bedarfen aller Abteilungen orientierenden Anforderungskatalogs ist deshalb ein relationales, nach Möglichkeit sogar objektorientiertes Datenmodell.

1.2 Vor- und Nachteile relationaler Datenbanktechnologie

Mit der Relationentheorie von Codd[1] wurde die Datenorganisation und -manipulation auf eine mathematisch präzise Grundlage gestellt (Mengenlehre, Normalisierungslehre, Synthesealgorithmen). Relationale Datenbanken erlauben die Implementierung von redundanzfreien und damit konsistenten Datenmodellen und eine mengenorientierte Datenverarbeitung. Darin unterscheiden sich relationale Datenbanken grundsätzlich von hierarchischen und netzwerkorientierten Datenbanken. Mit dem Relationenmodell als konzeptionelles Datenbankschema sind folgende Vorteile verbunden:

- Sehr einfaches Datenmodell
- Garantierte Datenkonsistenz (bei normalisierten Modellen), da Änderungen nur an einer Stelle vorgenommen werden
- Beliebige Einstiegspunkte in die Datenbank, da alle Relationen „gleichberechtigt“ sind (keine ausgewiesenen Einstiegspunkte wie bei hierarchischen und netzwerkorientierten Datenbanken)
- Verknüpfung über Inhalte
- Da jedes Objekt in Einzelteile zerlegt wird und die Daten mengenorientiert verarbeitet werden, sind komplizierte Abfragen auf große Datenmengen und mächtige Operationen gegen die Daten möglich (s. Quantoren)
- Hohes Maß an Datenunabhängigkeit: kein Navigieren in der Datenbank durch den Benutzer; die „Navigation“ obliegt völlig dem Datenbanksystem; die Sicht der Benutzer auf die Daten wird abstrakter; es sind weniger Kenntnisse über die physische Datenorganisation erforderlich
- Standardisierte einheitliche Datenbanksprache (SQL).

[1] Vgl. hierzu: Schlageter/Stucky (1987), Ullmann (1988), Sauer (1992).

Gerade aufgrund der garantierten Datenkonsistenz und der flexiblen Abfragemöglichkeiten haben sich relationale Datenbanksysteme zum de-facto-Industriestandard entwickelt.[2]

Die Vorteile des relationalen Ansatzes sind aber zugleich auch seine Schwächen:

- Beziehungstypen werden wie Entity-Typen in Relationen (Tabellen) abgelegt. Dies ist eine komfortable Vereinfachung, bedeutet aber, daß semantische Beziehungen zwischen Entitäten, wie sie im Entity-Relationship-Modell definiert werden, im Relationenmodell nicht explizit ausgedrückt werden können („semantische Lücke“).
- Is-a- und Part-of-Beziehungen lassen sich ebenfalls nicht ausdrücken, da es im klassischen Relationenmodell nicht möglich ist, den Tupeln wiederum Sub-Tupel zuzuordnen.
- Die bei der Normalisierung erforderliche Zerlegung der Daten in Einzelteile erschwert inhaltlich und unter Performance-Gesichtspunkten die Suche nach komplexeren Zusammenhängen: Daten, die getrennt voneinander gespeichert werden, aber gemeinsam zu benutzen sind, um Informationsobjekte zu erzeugen, so wie der Anwender sie kennt, müssen auf der Anwendungsebene in Form von Joins wieder zusammengeführt werden.
- Es läßt sich wenig Wissen über die Regeln und Funktionen speichern, die mit den Daten verbunden sind.

1.3 Vor- und Nachteile objektorientierter Datenbanken

Mit objektorientierten Datenmodellen werden dagegen drei grundlegende Konzepte in die Datenbankentechnologie eingeführt, die die Nachteile des Relationenmodells weitgehend auffangen[3]:

- Gegenstände und Aspekte, d.h. Objekte eines zu modellierenden Weltausschnitts sind sehr oft selbst aus anderen Objekten in beliebiger Weise zusammengesetzt (Teil-von-Beziehung bzw. *use*-Relation zwischen Objekten). Um diese Zusammenhänge möglichst vollständig, d.h. 1:1 in der Datenbank abbilden zu können, ist eine hohe Ausdrucksmächtigkeit des Datenmodells erforderlich, die herkömmliche Datenbanksysteme mit ihrer geringen Strukturierungstiefe nicht oder nur eingeschränkt erfüllen können.

---

[2] Vgl. auch Gerkens (1994).

[3] Vgl. hierzu: Drucks (1992), Bauer (1995a), Schmatz/Weikert (1995).

Objektorientierte Datenbanken gestatten dagegen die Definition von beliebig komplexen Objekten und erlauben damit eine realitätsnahe Modellierung mit hoher Strukturkomplexität. Auf diese Weise wird eine vereinfachte Sicht der Welt abgebildet, die die semantische Interpretierbarkeit des Modells erhöht.

- Mit den Objekten zusammen können Objektmethoden (Wissen über die Daten) hinterlegt werden, durch deren Anstoßen Verarbeitungsoperationen, die zusammen mit den Daten definiert werden, automatisch ablaufen.
- Klassifikation kann explizit ausgedrückt werden: Eigenschaften bestehender Klassen können an neue (Unter-)Klassen vererbt werden, so daß Operationen nur einmal kodiert werden müssen.

Aber auch mit objektorientierten Datenbanken sind spezifische Probleme verbunden:

- Objektorientierte Datenbanksysteme sind noch nicht marktreif.
- Es gibt bislang noch keine mathematische Fundierung objektorientierter Datenmodellierung, die mit der Relationentheorie vergleichbar wäre, sondern nur allgemeine Gestaltungsprinzipien. Die Qualität und der semantische Informationsgehalt objektorientierter Datenbanken steht und fällt damit mit dem Klassenmodell, das - in weitaus stärkerem Maße als das Relationenmodell - eine individualistische Konstruktion ist. Zusammenhänge zwischen Entitäten müssen fest verpointert werden, womit viele Probleme semantischer Netzwerke v.a. hinsichtlich der Modifizierbarkeit des Modells wieder eingeführt werden (Nethacking).
- Die Flexibilität der Abfragemöglichkeiten des Relationenmodells ist hier wieder aufgegeben worden, da die Abfragemöglichkeiten auf die Objekte eingeschränkt sind, die auch modelliert wurden.
- Objektorientierte Datenbanken haben notwendigerweise einen ungleich höheren Speicherbedarf als relationale Datenbanken, da mehr Information hinterlegt werden muß.
- Es gibt Probleme mit der Identität von Objekten.

Eine Variante, die die Vorteile relationaler und objektorientierter Modellierungstechniken integrieren würde, sind sog. *objekt-relationale* Systeme, bei denen auf ein Relationenmodell eine objektorientierte Schicht aufgesetzt wird[4]. Bei der Auswahl eines Datenbanksystems ist daher auch nach den Weiterentwicklungsoptionen in Richtung Objektorientierung zu fragen.

---

[4] Vgl. auch Drucks (1992), Vorwerk (1995).

1.4 Anforderungen an ein relationales Datenbanksystem im IZ

Über ein relationales Datenmodell und bestimmte Kernfunktionalitäten des DBMS (Datenintegrität und -sicherheit, Performance etc.) hinaus sind folgende informationstechnologische Anforderungen für das IZ in besonderer Weise relevant:

- Verteilte Datenhaltung und Zugriff auf verteilte, heterogene Datenbanken:

  Die geographische und inhaltliche Aufgabenverteilung innerhalb des IZ (s. Außenstelle) und der GESIS erfordern komfortable und mächtige Möglichkeiten der verteilten Datenhaltung und ggf. sogar Möglichkeiten eines homogenen Zugriffs auf verteilte, heterogene Datenquellen.

- Interaktive graphische Browsing-, Datenanalyse- und Reportingwerkzeuge:
  Die fehlende Möglichkeit, in benutzerfreundlicher Form beliebige ad-hoc-Anfragen an die Datenbank zu stellen, die nicht auf festdefinierte Masken eingeschränkt sind, sowie statistische Analysen und Berichte erstellen zu können, ist das Grunddilemma der zu Zeit eingesetzten Datenbanktechnologie.

- Integrierte Textretrievalfähigkeit:
  Im Zusammenhang mit der (virtuellen) Integration der GESIS-Datenbestände, aber auch innerhalb des IZ (s. Textfelder, „Bürodokumente“), stellt sich das Problem der Kombination von strukturierten und unstrukturierten Daten. So muß es z.B. möglich sein, Textdokumente innerhalb der gleichen Datenbankumgebung zu speichern, in der sich auch die strukturierten Daten befinden, so daß sich innerhalb eines SQL-Befehls Abfragen auf strukturierten Datensätzen mit Volltext-Recherche verbinden lassen.[5] Dieses Problem stellt sich auch im Hinblick auf den Aufbau eines *Information Warehouse*, das aggregierte Information für den alltäglichen Informationsbedarf zur Verfügung stellt[6].

- Ausnutzung paralleler Rechnerarchitekturen, insbesondere die parallele Verarbeitung von Datenbankanforderungen

- Mächtige graphische Frontend-Tools, die eine visuelle Modellierung von Datenbanken und Arbeitsabläufen (inkl. Anwendungsgenerierung) sowie

---

[5] Vgl. auch Ritter (1995).
[6] Vgl. Haarmann (1995).

eine integrierte und portable Entwicklung von Datenbankoberflächen und (komplexeren) Berichten erlauben.

- Hardware- und Betriebssystemunabhängigkeit
- Unterstützung von Workflowsystemen
- Internet-Anbindung, WWW-Fähigkeit
- Support und Zukunftssicherheit (technologische und wirtschaftliche Potenz des Herstellers).

1.5 Die Evaluationsstrategie der Studie

Ziel (und Auftrag) der Studie ist es, die gängigsten relationalen Datenbanksysteme für größere UNIX-Plattformen im Hinblick auf die informationstechnologischen Anforderungen des IZ zu bewerten. Das vorliegende Papier und die Beurteilung der Produkte basiert auf der Lektüre der einschlägigen Fachliteratur und Fachpresse, technischer Produktbeschreibungen und Handbücher, vielen Gesprächen mit Herstellern und Anwendern und diversen Präsentationen.

Das Thema der Studie ist gigantisch - dies gilt nicht nur für die große Zahl zu untersuchender Aspekte, sondern auch hinsichtlich der Komplexität jedes dieser Einzelprobleme. Der Autor hat es sich daher zum Ziel gesetzt, nur *die* relevanten Informationen zusammenzutragen, die benötigt werden, um eine vernünftige, sich an den Bedarfen des IZ orientierende Entscheidung treffen zu können. Der systematische Vergleich der Datenbankprodukte wurde deshalb nur so weit vorangetrieben, wie notwendig war, um auf der Basis der so gewonnenen Erkenntnisse zu einer Entscheidung kommen zu können. Grundsätzlich könnte eine derartige Datenbankstudie noch weiter ausgedehnt und vertieft werden, indem man z.B. eine systematische Auswertung der Systemhandbücher und Marktstudien durchführt (oder durchführen läßt) oder die Leistungsmerkmale der Systeme empirisch testet (bzw. testen läßt). Der Autor glaubt jedoch, daß man aufgrund dieser „Zwischenbilanz“, die einen *Gesamteindruck* der Produkte vermitteln soll, zu einer tragfähigen Entscheidung kommen kann.

Die Studie konzentriert sich auf die gängigsten relationalen Datenbanksysteme, die auf dem Server-Betriebssystemen UNIX eingesetzt werden. Dazu zählen[7]:

[7] Da Ingres vom deutschen Markt weggebrochen ist, wurde dieses Produkt nicht in Betracht gezogen.

- ORACLE 7.1
- SYBASE SQLServer 10
- INFORMIX-OnLine Dynamic Server 7.1
- ADABAS-D 3.1.2 (Software AG)

In Kapitel 2 werden die in einer Client/Server-Umgebung relevanten Anforderungen und Leistungsmerkmale dieser Produkte ausführlich dargestellt und miteinander verglichen. Am Ende eines jeden Kapitels befindet sich eine tabellarische Zusammenfassung, die (dem ungeduldigen Leser) einen Überblick vermitteln soll[8].

2 Evaluation der Datenbankprodukte

2.1 SQL-Funktionalität (Relationalität)

2.1.1 SQL-Sprachumfang und -kompatibilität

Die mengenorientierte Verarbeitung von Datensätzen (Tabellenzeilen) wird bei relationalen Datenbanksystemen durch die (einheitliche) Sprachschnittstelle SQL (*Structured Query Language*) unterstützt. SQL ist eine konsequent nichtprozedurale Sprache mit einem extrem limitierten Sprachumfang, die jedoch die Mächtigkeit des relationalen Modells voll ausnutzt. Inzwischen gibt es einen international verbindlichen SQL-Standard, der folgende Komponenten umfaßt[9]:

- ISO-SQL-89: normiert die Basisoperationen und (teilweise) die Definition von referentiellen und Entity-Integritätsregeln.

Mit der Verabschiedung des ISO-SQL2-Standards 1992 (ISO-SQL-92) wurde eine Fülle wichtiger Verbesserungen und Erweiterungen eingebracht, v.a. hinsichtlich Strukturierung des Systemkatalogs, relationale Operatoren und Datentypen, Änderungsmöglichkeiten bei der Definition von Datenbankobjekten und Dynamic SQL. SQL-92 definiert drei Sprachebenen, um den Datenbank-

---

[8] Zutreffen von Merkmalen ist mit einem '+'-Zeichen gekennzeichnet, Nicht-Zutreffen mit '-'; '~' bedeutet 'nach oben hin unbegrenzt'; bei leeren Feldern blieben die Fragen unbeantwortet - dies war insbesondere bei SYBASE der Fall - oder konnten nicht abschließend geklärt werden.

[9] Vgl. Sauer (1992), Jenz & Partner (1994).

herstellern die Gelegenheit zu geben, ihre SQL-Sprachimplementierung schrittweise auf den vollen Sprachumfang von SQL-92 zu bringen:

- Ebene 1 (*Entry Level*): beinhaltet (größtenteils) die Grundfunktionalität von SQL-89; es fehlen jedoch Sprachkonstrukte u.a. für Dynamic SQL, die Änderung des Datenbankschemas und die Behandlung von Isolationsebenen bei Transaktionen.
- Ebene 2 (*Intermediate Level*): enthält eine Untermenge der Ebene 3, es fehlen jedoch Elemente zur Deklaration temporärer Tabellen, zur Behandlung Rollender Cursor oder eines kaskadierenden Update.
- Ebene 3 (*Full SQL*): der komplette Sprachumfang.

Dieser Standard wird zur Zeit von keinem Datenbanksystem vollständig abgedeckt. So entsprechen die meisten SQL-Sprachimplementierungen im wesentlichen nur dem Entry Level (ORACLE, ADABAS-D, SYBASE, INFORMIX). Intermediate und Full Level sind bei allen Anbietern in der Entwicklung. Für die Implementierung dieser Ausbaustufen dürften jedoch noch einige Jahre vergehen.

Bei diesen Normierungen handelt es sich allerdings um einen minimalen Standard, der der SQL-Praxis hinterherhinkt. Viele wichtige Architekturmerkmale relationaler Datenbanken und für die Praxis notwendige Sprachkonstrukte sind im SQL92-Standard nicht oder nur teilweise definiert: Löschen von Tabellen, Ändern von Tabellenstrukturen, Prädikate, Funktionen, referentielle Integrität, Returncodes, Textretrieval, Trigger, Stored Procedures, Index-Konzepte, Embedded SQL. Obwohl der SQL92-Standard von den Herstellern noch nicht vollständig abgedeckt wird, bietet jeder DBMS-Anbieter daher eigene Spracherweiterungen an, die über den Standard weit hinausgehen. Die meisten Produkte realisieren sogar schon Leistungsmerkmale (z.B. Trigger), die erst für die dritte SQL-Hauptversion 1995/96 vorgesehen sind. Eine wirkliche DBMS-Unabhängigkeit auf der SQL-Ebene ist in der Praxis daher nicht gegeben.

Die z.T. gravierenden Unterschiede im SQL-Sprachumfang sind vor allem im Hinblick auf die Portabilität von Datenbankanwendungen ein grundsätzliches Problem, da „der vermeintliche Ausweg, die Beschränkung auf elementarste Sprachelemente, ... wegen der geringen Funktionalität den heutigen Anforderungen nicht (standhält)“[10]. Trotz eines international verbindlichen SQL-

[10] Herzog, U. und Lang, S.M.; vgl. zu diesem Problemkreis auch: Koch, Jürgen (1995), Nußdorfer (1994), Bauer (1995), Kuppinger (1995).

Standards ergibt sich somit eine enge Bindung an den Hersteller. Der technologischen und wirtschaftlichen Potenz des Herstellers, in dem Bereich SQL-Standardisierung mitzuziehen oder sogar Akzente zu setzen, kommt daher eine strategische Bedeutung zu. Dies dürfte zumindest bei ADABAS-D ein Problem sein.

Da es für die Praxis nicht oder nur schwer möglich ist, sich auf den allen Datenbanksystemen gemeinsamen (normierten) SQL-Sprachvorrat zu beschränken, wird im folgenden auch nach Sprachelementen und Funktionen gefragt werden, die diesem Standard nicht entsprechen, sich aber in der Praxis als unerläßlich herausgestellt haben.

2.1.2 Datendefinition

Über die Basis-Datentypen hinaus, die alle DBMS-Hersteller anbieten, sind hier folgende Datentypen und Indexarten von Interesse:

- varchar (suchbare Textfelder variabler Länge): Bei ORACLE kann dieses Feld bis zu 2000 Zeichen enthalten, bei ADABAS-D, SYBASE und INFORMIX nur 254 bzw. 255 Zeichen.
- long (bis zu zwei GB große nicht-suchbare Textfelder): Bei ORACLE kann nur ein long-Feld pro Tabelle angelegt werden, bei den anderen Anbietern dagegen auch mehrere.
- raw (Binärdaten), long raw
- Unique-Index: wird automatisch bei allen primary-key- und unique-Spalten angelegt
- Non-Unique-Index: sollte für alle Foreign-Keys und für Spalten mit einer großen Selektivität erzeugt werden
- Zusammengesetzte Indices: aus mehreren Spalten bestehend.

Daß bei ORACLE z.Zt. nur ein long-Feld pro Tabelle angelegt werden kann, ist eine Schwäche dieses Produkts, da mehrere, funktional von einem Primärschlüssel abhängige Textfelder in einem normalisierten Datenmodell dann auf entsprechend viele Tabellen verteilt werden müßten. Bei einer entsprechenden Anfrage werden dann unnötige Join-Operationen notwendig. Da in den Datenbeständen des IZ davon jedoch nur Textfelder betroffen sind, die ohnehin keine große Selektivität haben (fremdsprachige Abstracts in SOLIS, Datengewinnung etc. in FORIS), und die deutlich relevanteren Abstracts ja der „Haupttabelle“ zugeordnet werden können, ist dieses Problem als nicht so gravierend einzustufen. Eine Verbesserung des long-Felder-Konzepts hat ORACLE für Version 8 angekündigt. Um inhaltlich unsinnige Tabellen-

verknüpfungen zu vermeiden, könnte bis dahin das Problem der zusätzlichen long-Felder dadurch gelöst werden, daß alle größeren Textfelder bis auf das Abstract intern in einer dreistelligen Relation der Form (<DokID>, <Feldname>, <Text>) abgespeichert werden. Dies ist eine nicht sehr elegante, aber praktikable Lösung, da auf diese Weise alle long-Felder in nur einer zusätzlichen Tabelle abgelegt werden müssen, wobei die Verteilung der Daten auf zwei physische Tabellen hinter einer View, die eine logische Sicht auf eine oder mehrere Tabellen darstellt, „versteckt“ werden könnte.

Hält man sich allerdings vor Augen, daß in long-Feldern, die eine reine Datenbehälterfunktion haben, nicht gesucht werden kann, kommt der Dimensionierung der varchar-Felder und einer integrierten Textretrievalfunktionalität (s.u.) für die Behandlung längerer Textfelder eine weitaus größere Bedeutung zu: In einer ORACLE-Datenbank besteht mit ungleich größeren varchar-Feldern als bei ADABAS-D, SYBASE und INFORMIX die Möglichkeit, auch längere Textdaten aufzunehmen (z.B. längere Titel oder kürzere Abstracts bis zu 2000 Zeichen Länge). INFORMIX bietet zwar auch die Möglichkeit, in einem zusätzlichen Zeichenfeld noch größere Textfelder zu speichern. Da dieses Feld jedoch eine feste Länge von 32K hat, ist dies keine Lösung für das oben genannte Problem.

Eine elegantere Lösung des Problems, die auch nur ORACLE für diese Fälle zu bieten hat, ist der Einsatz einer integrierten Textretrievalkomponente, die eine kombinierte Suche in strukturierten und unstrukturierten Datenbeständen mit voller Textretrievalfunktionalität erlaubt (s.u.). Grundsätzlich ist der kritische Bereich der Textdatenverwaltung somit nicht (nur) ein Problem der maximalen Anzahl von long-Feldern pro Tabelle, sondern der Größe der varchar-Felder und der Möglichkeit, die relationale Algebra um Textretrievalfähigkeit zu erweitern.

Kritischer ist da schon die Beschränkung der internen (komprimierten) Datensatzlänge auf 4K bei ADABAS-D, die zu ernsthaften Problemen führen kann, und die Tatsache, daß ADABAS-D offenbar kein relationales, sondern ein gekapseltes Data Dictionary hat. Damit wäre bei ADABAS-D eine wichtige Grundanforderung an relationale DBMSe nicht erfüllt (4. Codd-Regel). Die Kapselung des Data Dictionaries führt u.a. dazu, daß Schema-Änderungen während des laufenden Betriebs nicht oder nur mit Einschränkungen durchgeführt werden können: Damit z.B. Felder gelöscht werden können, muß bei ADABAS-D die Datenbank heruntergefahren werden.

2.1.3 Datenmanipulation

Die grundlegenden relationalen Operatoren (Joins, Mengen-Operationen, Quantoren) und Funktionen (Stringverarbeitung, arithmetische Funktionen etc.) werden von allen untersuchten Datenbanksystemen unterstützt, bis auf die folgenden gravierenden Ausnahmen:

- Der Auto-Join, d.h. die Verknüpfung einer Relation mit sich selbst, und der bei einer Verknüpfung größerer Tabellen performance-mäßig überlegene Sort-Merge-Join sind bei ADABAS-D nicht vorhanden.
- Die Auflösung hierarchischer Strukturen (s. transitive Beziehungen zwischen Deskriptoren, Personen etc.) wird *nur* von ORACLE unterstützt.
- Datenänderungen auf Views sind bei allen Systemen möglich; INFORMIX unterstützt allerdings kein Update auf Join-Views, welches bei verteilter Datenhaltung gebraucht wird.

2.1.4 Integrierte Textretrievalfunktionalität

Als einziger Hersteller bietet ORACLE eine integrierte Textretrieval-komponente an (ORACLE TextServer3), mit der eine ORACLE-Datenbank um die Fähigkeit zur Volltextrecherche erweitert werden kann. Der TextServer ermöglicht die Speicherung beliebig großer Dokumente, die sowohl im ursprünglichen Dateisystem als auch innerhalb der Datenbank (Datentyp *long*) abgelegt sein können. Dabei können bestehende Tabellen strukturierter Daten nachträglich durch Textspalten ergänzt werden. Umgekehrt können zu einer Texttabelle strukturierte Datenfelder hinzugefügt werden. Da auf diese Weise alle Informationen (Dokumente, Daten, Indizes, Thesauri) in Tabellen einer Datenbank gespeichert werden, können auch alle Verwaltungs- und Sicherheitsfunktionen des DBMS (Backup, Recovery, Zugriffsschutz, Lesekonsistenz, Transaktionsschutz bei gleichzeitigen Zugriff im Mehrbenutzerbetrieb etc.) genutzt werden.

Der TextServer erlaubt eine kombinierte SQL-Abfrage in strukturierten Datensätzen und unstrukturierten Textdatenbeständen (contains-Operator). Dadurch können Dokumente nicht nur über Spaltenwerte einzelner Tabellen (z.B. Autor, Erscheinungsjahr etc.) gefunden werden, sondern auch über freie Textpassagen in großen Textdokumenten. Der Zugriff erfolgt über einen Wort- und einen Bitmap-Index (Location List), der für alle Dokumente in der Datenbank angelegt wird. In einem Bitstring der Location List können bis zu 500.000 Dokumente referenziert werden, d.h das Vorkommen eines Wortes kann mit

einer einzigen I/O-Operation in bis zu 500.000 Dokumenten festgestellt werden. Bei mehr Dokumenten müßte für das betreffende Wort ein neuer Bitstring, d.h. eine neue Zeile in der Location List angefangen werden. Ein weiterer Index ermöglicht die Abspeicherung zusätzlicher Information (Position eines Wortes innerhalb eines Dokuments, Häufigkeit des Wortes).

Zur Partitionierung der Indices können zusätzlich Virtuelle Texttabellen angelegt werden, die beliebig viele, ggf. auch verteilt gehaltene Texttabellen sowie weitere Virtuelle Texttabellen umfassen können. Auf diese Weise entsteht eine baumartige Struktur, die bei einer Suche rekursiv durchlaufen wird. Dadurch können nicht nur große Textdatenbestände mit einer einzigen Abfrage erfaßt werden. Es können Abfragen auf Textdatenbestände auch parallelisiert werden, so daß schnelle Antwortzeiten auch bei großen Datenmengen möglich sind. Die Technologie der Virtuellen Texttabellen bietet sich an, wenn mehr als 500.000 Dokumente indiziert werden müssen, so daß für ein Wort in der Location List nicht mehrere Bitstrings abgefragt werden müssen. Durch eine Verteilung auf mehrere Texttabellen, auf die über eine Virtuelle Texttabelle zugegriffen wird, wird so gewährleistet, daß es für jedes Wort pro Bitmap Index genau einen Bitstring gibt. Fraglich ist allerdings, ob bei einem Einsatz Virtueller Texttabellen auch komplexe Join-Operationen mit strukturierten Datensätzen performant abgearbeitet werden.

Die Suchmöglichkeiten des TextServers umfassen alle gängigen Textretrievalfunktionen:

- Suchbar sind einzelne Begriffe, zusammengesetzte Wörter oder Satzteile
- Verwendung von Platzhaltern, Rechts- und Linkstrunkierung, Abstandssuche; alle Möglichkeiten können zusammen verwendet werden.
- Durch die Integration in SQL (Boolesche Logik) können auch komplexe, geschachtelte Abfragen verarbeitet werden (*contains*-Operator).
- Abfragen können gespeichert und wiederverwendet werden.
- Thesaurus-Funktionalität
- Erzeugung einer Trefferliste.

Der Text wird in einem Textfenster im ASCII-Format dargestellt; die gesuchten Textstellen werden dabei hervorgehoben. Das Textverarbeitungssystem, mit dem das Dokument erstellt wurde, kann von ORACLE aus über einen Formatmanager aufgerufen werden, so daß das Dokument im Originalformat bearbeitet werden kann. Eine Aktualisierung des Dokuments und die Indizierung kann direkt online in der Datenbank oder später im Batch-Modus nach-

vollzogen werden. Eigene Formatmanager sind über eine offene Schnittstelle (API) in den TextServer integrierbar. Dialoganwendungen mit dem TextServer können in Forms- oder Pro*C-Applikationen eingebunden werden.

2.1.5 Weiterentwicklungsperspektiven hinsichtlich SQL-Funktionalität

Alle betrachteten DBMS-Hersteller beabsichtigen, in Richtung Objektorientierung weiterzugehen, allerdings ohne dabei die relationale Kernel-Technologie aufzugeben. So sind bei ORACLE, ADABAS-D und INFORMIX für die nächsten Versionen objektorientierte bzw. objektrelationale Datenbankkonzepte in der Entwicklung. Zum strategischen Konzept von ORACLE gehört darüber hinaus die volle Integration des TextServers in den Datenbankkern.

2.1.6 Tabellarische Zusammenfassung

	ORACLE	ADABAS-D	SYBASE	INFORMIX
Schema-Modifikation (deklarativ, im lfd. Betrieb)				
Feld hinzufügen	+	+		+
Feld löschen	+	-		+
Ändern Datentypen	+	z.T.		+
Ändern Tabellennamen	+	+		-
Ändern Integritätsbedingungen	+	+		+[11]
DDL-Statements im lfd. Betrieb	+	+		+[12]
Relationales Data Dictionary	+	-		+
DD-Views	+			+
DDL auch in 3GL anwendbar (z.B. 'create table')	+			+
Datentypen				
Zeichen (char)	+	+	+	+
Datum und Zeit (date / time)	+	+/+		+/+

[11] 2-stufig.
[12] Sperre der Tabelle notwendig.

	ORACLE	ADABAS-D	SYBASE	INFORMIX
Ganzzahl (integer)	+	+		+
Festkommazahl (numeric, fixed)	+	+		+
Gleitkommazahl (float, real)	+	+		+
Zeichenketten variabler Länge (varchar): max. Länge	2000	254	255	255[13]
long (2 GB Textfeld) / mehrere pro Tabelle	+/i.E.	+[14]/+	+/	+/+
Binärdatenfeld (raw, image) / long raw (2 GB)	+/+	s. long	+	+
Schlüsselfeld (row-id)	+	+?		+
benutzerdefinierte Datentypen	-	-	+	-
Konvertierungsfunktionen (für alle Datentypen)	+	+	+	+
Index-Konzepte				
Unique-Index	+	+		+
Non-Unique-Index	+	+		+
Zusammengesetzte Indices	+	16		16
SQL-Kompatibilität				
ANSI SQL 89	+	+	+	+
SQL-2 Entry Level	+	+	+	+
SQL-2 Intermediate Level	i.E.	i.E.		i.E.
SQL-2 Full	i.E.	i.E.		i.E.
SQL- Queries (Operatoren)				
Auto-Join	+	i.E.		+
Outer-Join	+	+		+
Sort-Merge-Join	+	i.E.		+
Nested-Loop-Join	+	+		+
Zeichenkettenfunktionen	+	+	+	+
Arithmetische Funktionen	+	+	+	+
Datums- u. Zeitfunktionen	+	+	+	+
Sortierte Ausgaben (benutzerdefiniert) / Länge Sortierfeld	+/~	+/250		+/~

[13] Zusätzlich bietet INFORMIX ein nicht-variables Zeichenfeld von 32 K Länge.
[14] Wird vom ODBC-Treiber z.Zt. allerdings nicht unterstützt.

	ORACLE	ADABAS-D	SYBASE	INFORMIX
Set-Operationen	+	+		+
Quantoren	+	+		+
Auflösung hierarchischer Strukturen	+	-		-
DML-Operationen auf Views / Änderbare Join-Views	+/+	+/+	+/	+/-
max. Anzahl der Tabellen pro Abfrage	~	16	16	32.000
ANSI-SQL-Mode	+	+		+
Integrierte Textretrievalfunktionalität				
kombinierte SQL-Suche in unstrukturierten u. strukturierten Daten	+	-[15]	-[16]	-[17]
Integration in Server-DBMS	+	-	-	-
Nutzung Thesaurus	+	-	-	-
Abstandssuche	+	-	-	-
Trefferlisten	+	-	-	-
Trunkierung	+	-	-	-
Integration mit Textverarbeitungsprogrammen	+	-	-	-
Weiterentwicklung				
in Richtung Objektorientierung	+ (V8)	+		+

2.2 Datenbankorganisation, Tuning-Ebenen und Administration

2.2.1 Speicherobjekte und physische Struktur der Datenbank

An der Art und Weise der File-Organisation entscheidet sich im wesentlichen die Frage, wie effektiv die Speicherplatzverwaltung der Datenbank hinsichtlich Datensicherheit, Administration und Performance ist. Hier ist insbesondere die

[15] Die Software AG bietet eine Textretrieval-Komponente für ADABAS-D an. Diese ist jedoch nicht in das DBMS integriert.

[16] Nur Drittanbieter-Produkt („Fulcrum“): nicht integriert.

[17] Von einem Drittanbieter ist ein Workgroup-Server („Excalibur“) mit Textretrieval- und Multimediafähigkeit für INFORMIX in Entwicklung.

Segmentierung der Datenbank von Interesse, für die ORACLE, SYBASE und INFORMIX Lösungen anbieten. Am ausgeprägtesten ist in diesem Bereich das Tablespace-Konzept von ORACLE, so daß hierauf näher eingegangen werden soll.

In einer ORACLE-Datenbank können beliebig viele Datenbank-Files verwaltet werden. Davon gibt es mindestens zwei Redo-Log-Files, die alle Datenänderungen protokollieren, die innerhalb einer Transaktion durchgeführt werden, und mindestens zwei Kontroll-Files, die Grundinformationen über die Struktur einer ORACLE-Datenbank enthalten.

Sämtliche Datenbank-Files werden in sog. Tablespaces verwaltet. Bei der Erstellung einer ORACLE-Datenbank wird ein System-Tablespace angelegt, indem u.a. das Data Dictionary (DD) abgelegt ist. Jedes Datenbankobjekt wird genau einem Tablespace zugeordnet. Die Datenbankobjekte verbrauchen den Speicherplatz dieses Tablespaces, d.h. sie können innerhalb eines Tablespaces wachsen. Die Tablespaces können vergrößert werden. Ein Datenbank-File kann aus einem Tablespace jedoch nicht mehr entfernt werden (!); man muß dann den kompletten Tablespace löschen und neu organisieren. Es ist möglich, mehrere Tablespaces offline zu setzen.

Die Vorteile dieses Speicherungsprinzip liegen in der besseren logischen Strukturierung der Datenbank:

- Logisch zusammengehörende Tabellen und Indices können in einem Tablespace zusammengefaßt und somit tablespace-weise verwaltet werden (s. Zugriffsrechte): z.B. System-Tablespace nur für das Data Dictionary; temporäre Segmente und Rollbacksegmente in eigenen Tablespaces (mit eigenen Speicherungsparametern).
- Unterschiedliche Tablespaces können auf unterschiedliche Plattenlaufwerke verteilt werden, was sich positiv auf die Performance auswirkt: z.B. Redo-Log-Files und System-Tablespace auf die schnellsten Platten, getrennt von den Datenbank-Files; Daten- u. Indexsegmente des gleichen Datenbankobjekts u.U. in verschiedenen Tablespaces auf unterschiedlichen Platten.
- Teile einer Datenbank können unabhängig von anderen gesichert und nach Plattenproblemen wiederhergestellt werden (Online Recovery).
- Verteilung von Datenbank-Files bedeutet zusätzliche Fehlertoleranz gegenüber Plattenfehlern: Der Ausfall eines Datenbereiches bedeutet nicht das Ende des Datenbankbetriebs (wie bei Adabas-D).

Ein Tablespace besteht aus Daten-, Index- und Rollbacksegmenten. Jede Tabelle und jeder Index belegt genau ein Daten-, Index- und Rollbacksegment

innerhalb eines Tablespaces. Um die Fragmentierung der einzelnen Segmente zu minimieren, werden die Speicherplatzzuordnung und der Speicherplatzverbrauch über Speicherungsparameter gesteuert (Segmentausprägung). Ein Segment besteht aus einem oder mehreren *extents*. Dies sind Speicherbereiche in bestimmbarer Größe, die einem Segment zugeordnet werden können. Ein extent besteht aus einem Anfangsextent und beliebig vielen Folge-Extents, die immer dann vom System erzeugt werden, wenn der bisher allokierte Speicherplatz nicht ausreicht, um neue Datensätze aufzunehmen. Die Größe der Next-Extents ist definierbar (zusätzlicher Wachtumsparameter). Jedes *extent* besteht aus einer bestimmten Anzahl von ORACLE-Datenblöcken (meist 2K groß). Die Datenbank-Blöcke sind die kleinsten Einheiten eines Segments. Sie bestehen aus einem Datenbereich (Insert- und Freibereich) und einem Datenbank-Block-Kopf und sind ebenfalls konfigurierbar.

Bei ADABAS-D wachsen und schrumpfen die Tabellen und Indices dagegen dynamisch, so daß hier zwar keine Administrationsmaßnahmen anfallen. Der Ausfall eines Datenbereiches bedeutet bei diesem Konzept allerdings das Ende des Datenbankbetriebs. ADABAS-D empfiehlt dieses Problem durch eine Verteilte Datenhaltung auf *einem* Server zu entschärfen. Dies entspräche inhaltlich einer Tablespace-Konfiguration. ADABAS-D kennt auch keine Rollbacksegmente. Alle Transaktionen und Datenänderungen (*Before-After-Image*) werden in *einer* Log-Datei protokolliert, was keine gute Lösung des Problems der Transaktions- und Datensicherheit ist.

2.2.2 Tuning-Ebenen

2.2.2.1 Lastprofilkonfiguration

Grundsätzlich werden zwei verschiedene Konfigurationsvarianten für die Verteilung von Datenbanklasten auf Serverprozesse unterschieden:

- „Dedicated“ (ORACLE):

  Bei einer großen Zahl von Anwendungsprozessen, die jeweils eine große Datenbanklast produzieren, gibt es bei ORACLE, ADABAS-D und SYBASE die Möglichkeit, jedem Anwendungsprozeß einen dedizierten Serverprozeß zuzuordnen, der alle Datenbankanforderungen der jeweiligen Anwendung exklusiv bearbeitet. Dies wird von INFORMIX nicht unterstützt.
- Dynamic Multithreading:

  Bei vielen Online-Benutzern und moderater Datenbanklast pro Benutzer ist dagegen eine explizite Beziehung zwischen Anwendungs- und Server-

prozessen nicht sinnvoll. Als Alternative zum herkömmlichen „Dedicated"-Server hat sich das Multithreading als Stand der Technik etabliert. Beim Multithreading wird der Anwendungsprozeß von einem „Dispatcher"-Prozeß (ORACLE) in einer Request-Queue des Datenbankservers ablegt. Dieser Request wird dann von einem beliebigen Serverprozeß bearbeitet, der das Ergebnis über eine Response-Queue an das Anwendungsprogramm zurückgibt. Durch dieses Verfahren verringert sich die Prozeßanzahl innerhalb des Systems erheblich, so daß typische Online-Anwendungen im Multithreading-Modus betrieben werden sollten. Dieses Verfahren wird von allen Systemen unterstützt (bei SYBASE gibt es allerdings Performance-Verluste in einer symmetrischen Multiprozessorumgebung[18]).

Die Entscheidung, ob eine Session im „Dedicated" oder Multithreading-Modus gefahren werden soll, geschieht bei ORACLE zur 'connect'-Zeit, so daß auf Lastprofiländerungen sehr flexibel reagiert werden kann. Die minimale und maximale Anzahl von Shared Server- und Dispatcher-Prozessen kann bei der Konfiguration der ORACLE-Instanz definiert werden.

Bei ADABAS-D dagegen muß bei der Installation (!) angegeben werden, in welchen Modus das DBMS gefahren werden soll. Ein ADABAS-D-System ist damit hinsichtlich Lastprofilkonfiguration nicht mehr skalierbar.

2.2.2.2 Cache-Konfiguration

Durch die Vorhaltung von SQL-Befehlen und Datenblöcken (*result sets*) im Datenbank-Cache kann der zur Verfügung stehende Hauptspeicher effektiv ausgelastet werden, da identische SQL-Befehle oder Stored Procedures und mehrfach genutzte Datenbereiche nur einmal im Speicher vorhanden sind. Darüber hinaus wird die Zahl der Plattenzugriffe minimiert, was die Performance insbesondere von interaktiven Anwendungen erhöht. So kann man z.B. erreichen, daß nur so viele Datensätze über das Netz gesendet werden, wie das Frontend benötigt; den Rest des Result-Sets hält der Datenbankserver in einem Cache auf Abruf bereit. Die Konfigurierbarkeit des Cache ist damit für die Performance von besonderem Interesse, da die richtige Einstellung dieser Parameter erhebliche Auswirkungen auf die Gesamtperformance des Systems hat.

Bei ORACLE und INFORMIX besteht der Cache aus folgenden konfigurierbaren Bereichen (bei ADABAS-D ist dagegen nur die Gesamtgröße definierbar):

---

[18] Computerwoche 44/1995, S. 17.

- Datenbank-Block-Puffer:

  Je größer die Anzahl der Datenpuffer, desto mehr Datenblöcke können gleichzeitig im Cache gehalten werden, ohne daß physische Zugriffe auf die Platte notwendig werden[19]. Bei ORACLE werden Datenblock-Zugriffsstatistiken für Cache-Analyse-Zwecke in interne DD-Tabellen eingetragen. Die Datenblöcke im Blockpuffer unterliegen einem last recently used-Algorithmus (LRU), der dafür sorgt, daß die Blöcke, die am längsten nicht mehr benutzt worden sind, aus dem Cache entfernt und ggf. auf die Datenbank-Files zurückgeschrieben werden. Dieses Verfahren ermöglicht ein asynchrones Schreiben der Datenblöcke und wird von allen Systemen unterstützt.

- Shared Pool:

  Dieser Teil des Cache speichert neben DD-Informationen auch die geparsten Repräsentationen von SQL-Befehlen und deren Ausführungspläne sowie compilierte Prozeduren und Trigger. Auf diese Weise kann ein einmal geparster SQL-Befehl von beliebig vielen Prozessen ohne aufwendiges Neu-Parsen genutzt werden. In ORACLE gibt es neben dem Shared-SQL-Bereich, der von beliebigen Prozessen genutzt werden kann, auch einen privaten SQL-Bereich, der sessionorientierte (private) Informationen, wie Ausführungsstatus eines Befehls oder Werte der Bindevariablen u.ä. beinhaltet. Dieser private Bereich wird bei einer Multithreading-Konfiguration ebenfalls im Shared Pool verwaltet. Der Shared Pool unterliegt ebenfalls einem LRU-Algorithmus.

- Instanz-Informationen:

  Dieser Bereich enthält Informationen über gesperrte Ressourcen, den Status von laufenden Transaktionen und die Reihenfolge der Abarbeitung der Warteschlange.

2.2.2.3 SQL-Tuning

Ein weiterer zentraler Tuning-Bereich ist die Optimierung auf der SQL-Befehlsebene. Diese Komponente hat die Aufgabe, für einen gegebenen SQL-Befehl den kostengünstigsten Zugriffsweg zu ermitteln und damit den Durchsatz komplexer SQL-Befehle zu steigern. Hier werden folgende Verfahren unterschieden:

---

[19] Der Blockpuffer sollte so konfiguriert sein, daß auf 10-20 logische Lese-Operationen ein physischer Datenbankzugriff kommt.

- Regelbasiert:

  Internes Expertensystem, das für die Bestimmung des optimalen Zugriffspfades die Reihenfolge der Bedingungen in der 'where'-Klausel auf der Basis von Heuristiken auswertet (nur ORACLE und INFORMIX).

- Statistik:

  Berücksichtigung statistischer Informationen pro Tabelle (alle Systeme: Anzahl Datensätze, Anzahl Blöcke, Anzahl unbenutzter Blöcke, durchschnittlicher freier Platz innerhalb der Blöcke, durchschnittliche Datensatzlänge), pro Spalte (nur ORACLE und INFORMIX: Anzahl der eindeutigen Werte pro Spalte, kleinster und größter Wert) und pro Index (alle Systeme: Höhe des Indexbaumes, Anzahl der Indexblöcke auf der tiefsten Ebene eines Indexes (Blattebene), Anzahl der eindeutigen Werte innerhalb des Indexes, kleinster und größter Wert u.ä.). Sind keine Statistiken vorhanden, führt der SQL-Optimierer Schätzungen durch, woraus der Zugriffspfad ermittelt wird.

- Statistik + Hints (nur ORACLE): z.B. first-rows, hash, index, cluster, rowid, ordered, Nested-Loop Join, Sort-Merge-Join.

Um SQL-Befehle zu erkennen, die unverhältnismäßig viel Zeit und Ressourcen benötigen, bieten alle Systeme die Möglichkeit, eine Session zu analysieren (CPU-Zeiten für Ausführen und Parsen, Anzahl der physischen und logischen Befehlsausführungen, Parse-Vorgänge, Zugriffspläne). Die Zugriffspläne können auf diese Weise analysiert werden, um SQL-Befehle, die innerhalb einer Anwendung sehr häufig ausgeführt werden, zu optimieren.

ORACLE und INFORMIX sind die einzigen DBMSe, die ein Aktivieren der Optimierungsmethode nicht nur auf Server- und Sessionebene, sondern auch auf SQL-Befehlsebene zulassen.

2.2.2.4 Parallele Query-Verarbeitung

Um für last- und transaktionsintensive Anwendungen die Antwortzeiten möglichst gering zu halten, ist es notwendig, daß einzelne Datenbankabfragen oder Recovery- und Backup-Maßnahmen parallel durchgeführt werden können. Hier sollten mehrere Strategien der automatischen Parallelisierung gleichzeitig unterstützt werden (Skalierbarkeit)[20].

[20] Vgl. auch Ritter (1995).

2.2.2.4.1 Tabellenpartitionierung

- Statische Partitionierung (ORACLE, INFORMIX):
  Bei dieser Strategie wird jede Tabelle bei ihrer Erstellung gemäß bestimmter Schlüsselbereiche auf einzelne Plattenlaufwerke verteilt. Ein SQL-Befehl wird dann zur Laufzeit gemäß der vordefinierten Tabellenpartitionierung aufgeteilt und von mehreren CPUs parallel verarbeitet.
- Dynamische Partitionierung (ORACLE, INFORMIX über Hash-Algorithmus):
  Im Gegensatz zur vordefinierten (starren) Aufteilung der Daten wird bei der dynamischen Partitionierung die Tabelle zur Laufzeit gemäß dem Parallelisierungsgrad aufgeteilt, der für das Datenbankobjekt oder den SQL-Befehl mit Hilfe von Hints angegeben worden ist. Diese Technologie erfordert keine grundlegende Reorganisation der Datenbestände in bestimmte Partitionen.

2.2.2.4.2 Parallelisierungsarten für SQL-Befehle

- Horizontale oder intra-Operations-Parallelisierung (ORACLE, INFORMIX): Eine Operation wird in mehrere Sub-Operationen aufgeteilt (typisches Beispiel: full-table-scan).
- Vertikale oder inter-Operations-Parallelisierung (ORACLE, INFORMIX): Zwei Operationsarten werden parallel bearbeitet (Beispiel: full-table-scan und gleichzeitige Sortierung).

2.2.2.4.3 Einsatzebenen der Parallelisierung

- Definition des Parallelisierungsgrades eines Datenbankobjekts beim Anlegen, d.h. im ‘create table’ oder bei einer Schemamodifikation im laufenden Betrieb (nur ORACLE).
- Angabe des Parallelisierungsgrades auch innerhalb eines SQL-Befehls (ORACLE, INFORMIX). Bei beiden Systemen wird die Parallelisierung der Operationen vom DBMS automatisch durchgeführt. Bei ORACLE kann man allerdings durch entsprechende Hints im SQL-Befehl über den Abstufungsgrad der Parallelisierung hinaus auch die Parallelisierung bestimmter Operationen erzwingen (sofern diese parallelisiert werden können).

2.2.2.4.4 Applikationsebene

Positiv auf die Performance wirkt sich auch die Nutzung von deklarativen und prozeduralen Integritätsbedingungen und Stored Procedures aus, die hoch-

optimiert im Datenbankserver vorliegen (im Falle der Stored Procedures sogar in kompilierter Form) und zentral vom Datenbankserver ausgeführt werden, ohne daß nach jeder atomaren SQL-Operation die Kontrolle an das Anwendungsprogramm zurückgegeben werden muß. Dadurch können der Kommunikationsaufwand und somit auch die Antwortzeiten in einer Client/Server-Umgebung wesentlich verbessert werden. Dieses Verfahren hat außerdem den Vorteil, daß derartige Routinen bei ihrer Aktivierung in den Shared Pool innerhalb des Datenbank-Caches geladen werden und damit auch anderen Benutzern zur Verfügung stehen (zu Triggern und Stored Procedures siehe auch Kapitel 2.3.1.2 und 2.5.2.1).

2.2.3 Administration

Aus den Ausführungen geht hervor, daß es bei ADABAS-D nicht viel zu regeln gibt. Damit fällt natürlich auch der Administrationsaufwand geringer aus. Es gehört in der Tat zum Konzept von ADABAS-D, ein „möglichst einfach zu administrierendes SQL-System“ (Produktbroschüre) anzubieten. Diese Marketing-Strategie ist einleuchtend: Da die Software AG auf dem DBMS-Markt den Marktführern zeitlich und (somit auch) technologisch hinterherhinkt, ist die leichte Administrierbarkeit ein sinnvolles Verkaufsargument. Die Nachteile dieses Konzepts gehen aus dem Papier hervor.

Von INFORMIX wurde dem Autor in vielen Gesprächen wegen des komplexen Administrationsaufwandes dieses Datenbankservers abgeraten. Es wurde allerdings einhellig bestätigt, daß die Installation und Anfangsparametrierung des ORACLE-Servers ein aufwendiges Geschäft sei. Der Folgeaufwand falle jedoch, eine „optimale“ Ausgangskonfiguration vorausgesetzt, vergleichsweise gering aus. Bei ORACLE sind also größere Anlaufschwierigkeiten zu erwarten als bei einem Produkt wie ADABAS-D. Für die Installation und Konfigurierung einer ORACLE-Datenbank sollte daher genügend Zeit eingeplant werden.

Für alle Systeme werden Administrationswerkzeuge (z.T. als Clients unter Windows), Leistungsmonitore und andere Utilities (Laden, Backup, Restore etc.) angeboten.

2.2.4 Tabellarische Zusammenfassung

	ORACLE	ADABAS-D	SYBASE	INFORMIX
Physische Architektur				
DB-Organisation				
Tablespace-Konzept (Segmentierung)	+	-	+?[21]	+
Rollbacksegmente	+	-		+
File-Organisation konfigurierbar	+	-[22]		+
DB auf mehrere physische Platten verteilbar	+	+	+	+
Indexarten				
unique	+	+	+	+
non-unique	+	+	+	+
zusammengesetzte Indices (aus wievielen Spalten)	16	16	16	16
Sekundärspeicherorganisation				
B-Baum (B*) / Reorganisationfreiheit	+/+	+/+		+/+
Hash	+	-		-
Physische Begrenzungen				
Anzahl Knoten im Netzwerk	~	~		~
Anzahl Datenbanknutzer	~	~		~
Anzahl Datenbanken pro Knoten	~	~		~
Anzahl Schlüssel pro Tabelle / Indices	~	127/255		~/~
max. Schlüssellänge / Indexschlüssel	~	255/255		255/255
max. Satzlänge	~	4048[23]		32 K
max. Namenslänge	~[24]	18	30	18

[21] Database Devices.
[22] Datenbank-Files wachsen dynamisch; nur die Gesamtgröße der Datenbank muß festgelegt werden.
[23] Interne (komprimiert) Länge, ohne BLOBs.
[24] Frei definierbar (s. varchar-Feld).

	ORACLE	ADABAS-D	SYBASE	INFORMIX
max. Feldanzahl pro Satz	255	255	250	32 K
Anzahl Tabellen pro DB	~	~	~	~
Lastprofilkonfiguration				
„Dedicated“ (explizite Beziehung Client-Prozeß u. Server-Prozeß)	+	+	+	-
Dynamisches Multithreading	+	+	+	+
Entscheidung zur connect-Zeit möglich	+	-[25]	-?	-
Cache-Konfiguration				
DB-Block-Puffer / konfigurierbar	+/+	+/-[26]	+/	+/+
Shared Pool / konfigurierbar	+/+	+/-	+/	+/+
Instanzinformationen / konfigurierbar	+/+	+/-		+/+
Asynchrones Commit / Last Recently Used Verfahren	+/+	+/+	+	+/+
SQL-Optimierung				
regelbasiert	+	-	-?	+
statistisch				
pro Tabelle	+	+	+	+
pro Spalte	+	-	-?	+
pro Index	+	+	+	+
Statistik + Hints				
hash	+	-		-
index	+	-		-
cluster	+	-		+
first-rows / all_rows / full / rowid	+	-		-
ordered	+	-		-

[25] Entscheidung nur bei Installation möglich.
[26] Nicht konfigurierbar, nur die Gesamtgröße des Cache.

	ORACLE	ADABAS-D	SYBASE	INFORMIX
Join-Varianten (Sort-Merge, Nested-Loop)	+	-		-
Analyse-Möglichkeiten für Anwender (Trace-File)				
auch der optimierten Suchabfragen	+	+		+
Ausführungspläne	+	+	+	+
CPU-Zeiten	+	+	+	+
Parse-Vorgänge	+	+	+	+
I/O-Ausführungen	+	+	+	+
Aktivieren der Methode				
auf Serverebene	+	+		+
auf Sessionebene	+	+		+
auf Befehlsebene	+	-		+
Vorübersetzung und Optimierung von ESQL	+	Compiler		+
Parallele Query-Verarbeitung				
statische Tabellenpartitionierung (zum Erstellungszeitpunkt)	+	i.E.	i.E.	+
dynamische Tabellenpartitionierung (zur Laufzeit)	+	i.E.	i.E.	+[27]
horizontale Parallelisierung (Aufteilung in Sub-Operationen)	+	i.E.	-	+
vertikale Parallelisierung (zwei Operationsarten parallel)	+	i.E.	-	+
Angabe Parallelisierungsgrad bei Tabellendefinition	+	i.E.	-	-
Angabe Parallelisierungsgrad innerhalb SQL-Befehl	+	i.E.	-	+[28]
Parallelisierbare Operationen				
full table scan	+	i.E.		+
Join-Operationen / Sort-Merge-Join	+/+	i.E.		+/+

[27] Über Hash-Algorithmus.

[28] Benutzer kann einen Parallelisierungsgrad von 1-100 vorgeben. Das System parallelisiert dann automatisch die entsprechenden Operationen, soweit diese parallelisierbar sind.

	ORACLE	ADABAS-D	SYBASE	INFORMIX
group-by	+	i.E.		+
order-by	+	i.E.		+
Set-Operationen	+	i.E.		+
distinct	+	i.E.		+
Sub-Queries	+	i.E.		+
Indexerzeugung	+	i.E.		+
Ladeoperationen	+	i.E.		+
Recovery	+	i.E.		+
Backup / Restore	+/	i.E.		+/+

2.3 Datenintegrität und Ablaufintegrität

2.3.1 Semantische Datenintegrität

Integritätsregeln sollen verhindern, daß fehlerhafte Eingaben die Datenbank zerstören. Konsistenzregeln und daraus abgeleitete Aktionen, die die Integrität der Daten sicherstellen, werden in der Praxis jedoch oftmals innerhalb der Anwendungsprogramme definiert. Auf diese Weise verteilt sich die Behandlung und damit auch die Verantwortung für die Konsistenz der Daten aber auf u.U. viele verschiedene Applikationen und deren Programmierer. Es ist daher schon seit längerem Stand der Technik, Integritätsregeln innerhalb der Datenbank - und somit zentral an einer Stelle - zu verwalten, so daß der Anwendungsprogrammierer vom Ballast der Konsistenzprüfung weitgehend befreit ist. Allerdings muß bei dieser Verfahrensweise, die entweder deklarativ oder prozedural implementiert werden kann, der Modellierung der Datenbank größere Aufmerksamkeit geschenkt werden.

2.3.1.1 Deklarative Definition von Integritätsregeln

Integritätsregeln können bei allen untersuchten Systemen in Form von *constraints* (ANSI-konform) in der Tabellendefinition, d.h. zum Erstellungszeitpunkt der Tabellen definiert werden. Bei der Aktivierung von Constraints wird die jeweilige Tabelle vollständig auf Constraint-Konformität geprüft.

Constraints können aber auch im laufenden Betrieb hinzugefügt und gelöscht, bei ORACLE und ADABAS-D sogar auch deaktiviert oder aktiviert werden.

2.3.1.1.1 Referentielle Integritätsregeln

Um die Integrität von Abhängigkeitsbeziehungen zwischen Tabellen oder zwischen Datensätzen derselben Tabelle sicherzustellen, die auf der Basis von Fremdschlüsseln bei der Tabellendefinition festgelegt werden (*foreign key reference*), müssen die Fremdschlüssel (*foreign keys*) mit den entsprechenden Primärschlüsseln (*primary keys*) der Mastertabelle übereinstimmen. Mit dieser referentiellen Integritätsprüfung wird sichergestellt, daß bei einer Veränderung von Fremdschlüsselspalten vom DBMS geprüft wird, ob dem neuen Foreign-Key-Wert ein entsprechender Wert in der Mastertabelle gegenübersteht. In der Mastertabelle können nur dann Primärschlüssel oder unique-Constraints geändert (update restrict) bzw. Datensätze gelöscht werden (delete restrict), wenn keine abhängigen Datensätze in der Foreign-Key-Tabelle mehr vorhanden sind.

Im Zusammenhang mit der Angabe referentieller Integritätsregeln können auch Aktionen definiert werden, die ausgeführt sollen, wenn innerhalb einer übergeordneten Tabelle ein Datensatz gelöscht wird. Wurden Foreign-Key-Beziehungen z.B. mit der on-delete-cascade-Option definiert, bewirkt dies bei einer Löschoperation in der Mastertabelle das automatische Löschen aller abhängigen Datensätze. Um hier Inkonsistenzen zu vermeiden, müssen referentielle Integritätsprüfungen in das Transaktionskonzept eingebettet sein. ORACLE hat dieses Problem durch shared-index-entry-Sperren gelöst, die das Löschen von Datensätzen verhindern, die aktuell von Transaktionen referenziert werden: Bei Änderungen in der Foreign-Key-Tabelle wird eine shared-Sperre für den Primary-Key-Wert innerhalb des Primary-Key-Indexes gesetzt. Dies ist ein noch kleineres Sperrgranulat als die Datensatzsperre, da der entsprechende Datensatz in der Mastertabelle weiterhin bearbeitet werden kann.

2.3.1.1.2 Entity- und Domainintegrität

Über die referentiellen Integritätsregeln hinaus gibt es bei allen Systemen standardmäßig die Möglichkeit, Integritätsprüfungen zu definieren, die die Tabellen und deren Daten selbst betreffen:

- Primärschlüssel (*primary key, unique*): stellen sicher, daß jeder Eintrag in einer primary-key- bzw. unique-Spalte einen anderen Wert aufweist.
- Wertebereiche (*check*)
- NULL-Wertverhalten *(not null*).

Bei SYBASE können über diese Standardmethoden hinaus auch wiederverwendbare benutzerdefinierte Datentypen mit eigenen Integritätsregeln vergeben werden. Bei allen Systemen gibt es DD-Views auf die Integritätsregeln.

2.3.1.2 Prozedurale Integritätsmethoden

Eine prozedurale Implementierung von Integritätsregeln in Form von *Triggern* oder *Stored Procedures* ist notwendig, wenn über die deklarative Regeldefinition hinaus zusätzliche Logik zur Gewährleistung der Datenintegrität definiert werden muß. Der Anwender hat damit die Möglichkeit, additiv zur normalen Datenmodellierung noch beliebig komplexe Aktionen auf der Datenbank zu formulieren, die dann allen Anwendungsprogrammen gemeinsam zur Verfügung stehen.[29] Auf diese Weise wird viel von der Anwendungslogik, die sonst in verschiedenen Programmen wiederholt implementiert werden müßte, zentral beim DBMS hinterlegt.

2.3.1.2.1 Trigger

Trigger sind komplexe Integritätsregeln, die als Folge einer insert-, update- oder delete-Operation auf einer Tabelle aktiviert und ausgeführt werden. Trigger werden nicht als kompiliertes Modul in der Datenbank abgelegt, sondern erst zur Laufzeit kompiliert (wie SQL-Befehle) und im Shared Pool zur Verfügung gestellt. Es werden für jedes Trigger-Ereignis (insert, update, delete) vier Triggerarten unterschieden:

- before Befehl: Ausführung vor dem auslösenden SQL-Befehl, unabhängig von der Zahl der betroffenen Datensätze (ADABAS-D und SYBASE nicht)
- before Datensatz: Ausführung jeweils vor der Änderung eines Datensatzes
- after Befehl: Ausführung nach dem auslösenden SQL-Befehl (ADABAS-D und SYBASE nicht)
- after Datensatz: Ausführung jeweils nach der Änderung eines Datensatzes.

Die Aktivierung von Triggern bewirkt keine vollständige Prüfung der Tabellenwerte, sondern (nur) die Bearbeitung des jeweiligen Befehls oder Datensatzes. Mit ORACLE 7.1 und bei ADABAS-D können pro Tabelle mehrere Trigger der gleichen Art implementiert werden, so daß bestehender Triggercode nicht geändert werden muß, wenn zusätzliche Funktionalität erforderlich ist. Bei allen Systemen können Trigger geschachtelt werden. Rekursive Trigger sind jedoch nur bei SYBASE möglich.

---

[29] Vgl. auch Bauer (1995b).

Sofern Integritätsprüfungen keine Datenbankzugriffe erfordern, können diese bei ORACLE und ADABAS-D auch auf der Client-Seite durchgeführt werden, so daß hier keine Performance-Verluste durch (unnötige) Speicherungsversuche auf dem Server entstehen. Dies ist bei INFORMIX nicht möglich; hier können Trigger immer nur auf der Server-Seite ablaufen.

2.3.1.2.2 Stored Procedures

Eine weitere Variante prozeduraler Verfahren, die zur Sicherstellung der Datenintegrität genutzt werden kann, sind Datenbankprozeduren bzw. *Stored Procedures*, die im Gegensatz zu Triggern in kompilierter Form in der Datenbank vorgehalten werden (s. hierzu Kapitel 2.5.2.1).

2.3.2 Ablaufintegrität

2.3.2.1 Transaktionsmodelle und Synchronisation paralleler Transaktionen

Grundsätzlich werden bei allen Systemen Änderungsoperationen stets im Cache innerhalb des Datenblocks durchgeführt, wobei die ursprünglichen Werte in einem Rollbacksegment (ORACLE, INFORMIX) bzw. in Log-Dateien abgelegt werden. Dieses Modell erlaubt die Durchführung beliebig komplexer lokaler oder verteilter Transaktionen. Alle Systeme (außer INFORMIX) bieten die Möglichkeit, Transaktion in kleinere Rollback-Einheiten aufzuteilen (Savepoints), so daß im Rollback-Fall weniger zurückgesetzt und wiederholt werden muß. Dies wirkt sich positiv auf die Performance aus. Darüber hinaus hat dieser Mechanismus den Vorteil, daß das Rollback-Geschäft vom Anwendungsprogrammierer gesteuert werden kann.

Ein weiterer wichtiger Aspekt ist in diesem Zusammenhang die Synchronisation paralleler Transaktionen, insbesondere von konkurrierenden Lese- und Schreibzugriffen. Gegenüber dem traditionellen dirty-read (ADABAS-D, SYBASE, INFORMIX) zeigt das Read-Consistency-Modell von ORACLE bei jedem Lesezugriff den konsistenten Zustand eines Datensatzes an, wie er zum Zeitpunkt der Ausführung eines ‘select’ bestanden hat, ohne daß die betroffenen Tabellen oder Datensätze für Änderungsoperationen gesperrt werden müssen. Hier zahlt sich bei ORACLE die ausführliche Nutzung von Rollbacksegmenten aus, die solange vorgehalten werden, bis die jeweiligen Änderungstransaktionen beendet sind und keine Daten aus dem Rollbacksegment mehr für Konsistenzzwecke bei Leseoperationen benötigt werden.

ADABAS-D kennt dagegen keine Rollbacksegmente, sondern speichert Transaktionen und die ursprünglichen Werte allesamt in einer (!) Log-Datei.

Damit konkurrierende Prozesse nacheinander ablaufen und sich nicht durch irgendwelche Wechselwirkungen beeinflussen (lost-update-Problem), muß es geeignete Sperrmechanismen geben, die parallele Transaktionen steuern. Hier werden folgende Sperrgranulate unterstützt:

- Datenbanksperre
- Tabellensperren
- Datenbank-Block-Sperren (ADABAS-D nicht)
- Datensatzsperren.

ORACLE unterstützt darüber hinaus noch die Index-Entry-Sperre (kleinstmögliches Sperrgranulat), d.h. bei Referenzierung eines Primary-Key-Wertes braucht nur dessen Index-Entry gesperrt zu werden.

2.3.2.2 Backup und Recovery

Inkonsistenzen durch Verlust von Transaktionen werden bei allen untersuchten Datenbanksystemen durch folgende Recovery-Verfahren vermieden:

- Prozeß-Recovery:

  Bei Absturz eines Anwenderprogramms wird automatisch die letzte offene Transaktion vollständig auf den ursprünglichen Zustand zurückgesetzt. Der alte Wert wird dabei aus dem Rollbacksegment bzw. der Log-Datei entnommen. Alle gesperrten Ressourcen werden automatisch wieder freigegeben.

- Instanz-Recovery:

  Beim erneuten Starten des DBMS wird für jede positiv abgeschlossene Transaktion ein Rollforward, für jede nicht abgeschlossene ein Rollback durchgeführt.

- Medium-Recovery:

  Die Wiederherstellung der Datenbank bei einem Verlust von Daten durch einen Plattenfehler setzt periodische Backups der Datenbank oder Teile derselben voraus.

Bei allen Systemen ist die Verfügbarkeit der Datenbank bei Backup- oder Recovery-Maßnahmen und damit die ununterbrochene Nutzung des Systems gewährleistet. Die Möglichkeit, diese Prozesse zu parallelisieren, ist nicht bei allen Anbietern gegeben: ORACLE unterstützt ein paralleles Recovery, Backup und Restore; ADABAS-D unterstützt nur paralleles Backup; INFORMIX unterstützt kein Parallel Recovery.

2.3.2.3 Datenschutz

Neben der standardmäßig unterstützten Granulierung der Objekt- und Systemprivilegien bietet ORACLE als einziger Anbieter ein ausgefeiltes Rollenkonzept an, das insbesondere bei großen Benutzerzahlen von erheblicher Bedeutung hinsichtlich des Verwaltungsaufwandes ist. Dabei können beliebige Objekt- und Systemprivilegien in einer Rolle bzw. Aufgabe konzentriert werden, die auch andere Rollen enthalten kann (Schachtelung von Rollen). Dieses Konzept ermöglicht die Modellierung von komplexen Anwenderklassen.

2.3.3 Tabellarische Zusammenfassung

	ORACLE	ADABAS-D	SYBASE	INFORMIX
Semantische Datenintegrität				
referentielle Integritätsregeln	+	+	+	+
Integritätsaktionen (Trigger, Rules)	+	+	+	+
Entity-Integrität	+	+	+	+
Domain-Integrität	+	+	+	+
wiederverwendbare benutzerdefinierte Datentypen plus Regeln	-[30]	-	+	-
Constraint-Arten				
primary key	+	+	+	+
unique	+	+	+	+
check	+	+	+	+
default	+	+	+	+
not null	+	+	+	+
foreign key reference	+	+	+	+
kaskadierendes Löschen (on delete cascade)	+	+		+

[30] User-defined Functions.

	ORACLE	ADABAS-D	SYBASE	INFORMIX
Constraint-Operationen				
Anlegen mit Tabellendefinition	+	+	+	+
Hinzufügen (im lfd. Betrieb)	+	+	+	+
Löschen (im lfd. Betrieb)	+	+	+	+
Aktivieren (im lfd. Betrieb)	+	+	-?	-
Deaktivieren (im lfd. Betrieb)	+	+	-?	-
Trigger-Arten				
before Befehl	+	-	-?	+
before Datensatz	+	+	+	+[31]
after Befehl	+	-	-?	+
after Datensatz	+	+	+	+[32]
wahlweise im Client oder auf Server	+	+		-[33]
pro Tabelle mehrere Trigger der gleichen Art	+	+	-[34]	z.T.[35]
Nested Trigger	+	+	+[36]	+
Rekursive Trigger	-	-	+	-
DD-Views Integritätsregeln	+	+		+
Datenschutz				
Objektprivilegien	+	+	+	+
Systemprivilegien	+	+	+	+
Rollenkonzept (Aufgabenbezogen)	+	-	-	+[37]

[31] „for each row“.
[32] „for each row“.
[33] Immer auf Server.
[34] Jeweils nur 1 update-, 1 insert- und 1 delete-Trigger.
[35] Nur für update-Trigger.
[36] Maximale Schachtelungstiefe: 16.
[37] Nur Groups (ein Objekt- oder Systemprivileg wird mehreren Nutzern als Gruppe zugeordnet); n:m-Beziehung zwischen Benutzern und Gruppen ist möglich, aber keine Schachtelung; Rollenkonzept im nächsten Release.

	ORACLE	ADABAS-D	SYBASE	INFORMIX
Transaktionssicherheit				
Sperrgranulate				
Index-Entry	+	-	-?	-
Satz	+	+	+?	+
Block, Page	+	-	+	+
Tabelle	+	+	+	+
Datenbank	+[38]	+	+?	+
Synchronisation paralleler Transaktionen (Isolationsebenen)				
Daten ändern, während andere lesen	+	z.T.[39]		+
Konsistentes Lesen, während andere updaten	+	z.T.[40]	-?	-
Automatische, kostenbasierte Deadlock-Erkennung	+	+	+?	+
Backup und Recovery				
Log-Dateien / Rollbacksegmente	+/+	+/-	+/	+/+
konsistente Aufsetzpunkte (Savepoints)	+	+	+	-[41]
Prozeß-Recovery	+	+	+?	+
Instanz-Recovery	+	+		+
Medium-Recovery (Backup)	+	+	+	+
Paralleles Recovery	+	-	-?	+
Paralleles Backup	+	+	-?	+
Paralleles Restore	+	-	-?	+
Partieller Backup	+	+	+	+
Verfügbarkeit bei Backup und Recovery	+	+		+
Gespiegelte Platten	+	+	+	+

[38] Durch Offline-Setzen (geht auch für Teile der Datenbank).
[39] Abhängig vom Lockmechanismus (s. Isolation Level).
[40] Abhängig vom Lockmechanismus (s. Isolation Level).
[41] Nicht vom Anwender steuerbar.

2.4 Verteilte Datenhaltung

2.4.1 Server-Autonomie und ortsunabhängiger Zugriff

Da INFORMIX und ORACLE (lediglich) lokale Systemkataloge unterstützen, wird die lokale Autonomie der Datenbankserver bei diesen Datenbanksystemen vollständig gewährleistet. Die Datenbankobjekte sind in der lokalen Datenbank verzeichnet. Es gibt damit bei diesen Systemen auch keinen Single Point of Failure. Der Zugang zu entfernten Datenbanken erfolgt über Pfaddefinitionen (Datenbank-Links). Damit Anwendungen die Sicht auf eine einzige logische Datenbank vermittelt werden kann, können bei ORACLE Synonyme (für Tabellenname und Datenbank-Link) angegeben werden, die die Pfaddefinitionen vor dem Anwender verbergen.

Bei SYBASE muß jeder an einer Transaktion beteiligte Server explizit angesprochen werden. Ein globaler Systemkatalog ist ebenfalls nicht vorhanden. Eine Verlagerung einer Tabelle auf einen anderen Server muß daher in allen Anwendungen nachvollzogen werden. Um der Anwendung die Sicht auf einen globalen Systemkatalog zu vermitteln, ist bei SYBASE die Implementierung eines „OmniSQL-Gateways“ erforderlich, mit dem ein globales Schema eingerichtet werden kann. Schemaänderungen in den lokalen Katalogen können in den OmniSQL-Katalog übertragen werden. Globale Katalogmodifikationen sind jedoch nicht möglich. Der globale Katalog stellt darüber hinaus einen Single Point of Failure dar, da bei Ausfall des Knotens, auf dem das OmniSQL-Gateway installiert ist, kein Zugriff auf andere Datenbanken mehr möglich ist.[42]

Bei ADABAS-D haben alle Server-Datenbanken eine Kopie des verteilten globalen Katalogs, so daß es keine „Master“-Datenbank gibt, die das globale Wissen hat und damit einen Single Point of Failure darstellt. Alle Server-Datenbanken sind darüber hinaus lokal autonom, d.h. unabhängig vom Zustand anderer Datenbankserver. Gleichzeitig können - im Gegensatz zu den anderen Systemen - verteilte referentielle Integritätsbedingungen definiert werden. ADABAS-D bezahlt diesen Vorteil allerdings mit einem gekapselten, d.h. nicht-relationalen Data Dictionary. ADABAS-D erfüllt damit ein zentrales Kriterium für die Relationalität eines Datenbanksystems nicht.

Zur Implementierung verteilter referentieller Integritätsbedingungen werden bei ORACLE und INFORMIX entsprechende Trigger eingesetzt.

---

[42] Vgl. Jenz & Partner (1995).

2.4.2 Verteiltes Transaktionsmanagement

Während in nicht-verteilten Datenbankumgebungen lediglich der Datentransport zwischen Client und Server zu lösen ist, muß bei datenbankübergreifenden Transaktionen die vollständige Durchführung derselben sichergestellt werden. Fällt ein beteiligter Knoten aus, muß die Integrität der verteilten Datenbank nach Wiederanlauf des ausgefallenen Knotens über alle Knoten hinweg garantiert sein. Bei allen Systemen kann jedes an einer Transaktion beteiligte Datenbanksystem über ein 2-Phase-Commit-Protokoll mit dem steuernden Datenbanksystem kommunizieren und somit die Konsistenz der Transaktion sicherstellen.

2.4.3 Fragmentierung

ORACLE, ADABAS-D und INFORMIX unterstützen eine horizontale und vertikale Verteilung der Daten, d.h. Tabellen können sowohl spalten- als auch datensatzweise fragmentiert und auf verschiedene Server-Datenbanken verteilt werden. Die Teiltabellen lassen sich durch Join-Views wieder zu einer Gesamttabelle zusammensetzen. Ein Update auf derartige Join-Views ist bei INFORMIX jedoch nicht möglich.

2.4.4 Replikationsunabhängigkeit

Um viele teure Zugriffe auf eine zentrale Tabelle zu vermeiden, werden von allen Datenbanksystemen folgende Replikationskonzepte unterstützt:

- Asynchrone Replikation (*loose consistency*): Die Tabellenkopie (Replikat) auf den „Filial“-Servern wird periodisch auf neuesten Stand gehalten (*snapshot*-Verfahren). Standardmäßig repräsentiert ein Replikat genau eine Master-Tabelle. Bei ORACLE sind auch komplexe *snapshots* möglich, d.h. ein Replikat kann sich aus mehreren Master-Tabellen zusammensetzen (durch Join). Es gibt bei ORACLE darüber hinaus zwei Refresh-Modi: *complete* (vollständiger Neu-Aufbau des Replikates) oder *fast* (nur Änderungen in der Master-Tabelle werden repliziert).
- synchrone read-only-Replikation: Änderung in der Original-Tabelle werden unmittelbar im Replikat nachvollzogen.
- synchrone read/write-Replikation: Es sind auch Änderungen im Replikat möglich, die dann in der Master-Tabelle nachvollzogen werden. Bei diesem Verfahren gibt es keine Master-Slave-Beziehung zwischen den beteiligten Datenbankobjekten mehr.

- Symmetrische Replikationen (nur ORACLE und INFORMIX): Die oben beschriebenen Verfahren haben das Problem, daß Replikationen nur innerhalb der auslösenden Transaktion bearbeitet werden können, so daß die Dauer einer ‘commit’-Operation mit jedem Replikat steigt. Das symmetrische Replikationsverfahren basiert dagegen auf einem asynchronen *Remote-Procedure-Call*-Mechanismus (RPC), bei dem eine remote-Operation nicht transaktionsgenau zur Auslösetransaktion abgearbeitet werden muß, sondern zu einem späteren Zeitpunkt durchgeführt werden kann. Dadurch können lokale Transaktionen mit ‘commit’ abgeschlossen werden, obwohl ein Teil der verteilten Transaktionen noch in der RPC-Queue abgelegt ist.

2.4.5 Plattformen-Unabhängigkeit

Die untersuchten Datenbanksysteme unterstützen alle gängigen Hardware- und Betriebssystemplattformen und Netzwerkprotokolle. Bei SYBASE gibt es allerdings Performance-Verluste in einer symmetrischen Multiprozessorumgebung[43]. Für den Zugriff auf heterogene Datenquellen bieten ORACLE, INFORMIX und SYBASE transparente Daten-Gateways für alle gängigen DBMS-Plattformen an (ADABAS-D nur für DB2 und ADABAS-C), die die Einbeziehung beliebiger Datenquellen in verteilte Transaktionen erlauben.

[43] Computerwoche 44/1995, S. 17.

2.4.6 Tabellarische Zusammenfassung

	ORACLE	ADABAS-D	SYBASE	INFORMIX
Verteilte Datenhaltung				
Schema-Verteilung	+	+	+	+
horizontale Verteilung	+	+	+?	+
vertikale Verteilung	+	+	-?	Views
verteilte referentielle Integrität	-	+	-?	-
Erfüllungsgrad Date-Regeln				
Lokale Autonomie	+	+		+
kein Verlaß auf einen zentralen Knoten	+	+		+
unterbrechungsfreier Betrieb	+	+		+
Standortunabhängigkeit	+	+		+
Fragmentierungsunabhängigkeit	+	+		+
Replikationsunabhängigkeit	+	+	+	+
Verteilte Query / Optimierung	+/+	+/+	+/+	+/+
Verteiltes Transaktionsmanagement (2-Phase-Commit)	+	+	+	+
Hardware-Unabhängigkeit	+	+		+
Betriebssystem-Unabhängigkeit	+	+		+
Netzwerk-Unabhängigkeit	+	+		+
DBMS-Unabhängigkeit (Transparent Gateways)	+	-[44]	+	+
Replikation				
asynchrone Replikation	+	+	+?	Trigger
(quasi-)synchron, read-only	+	+		+
(quasi-)synchron, read-write	+	+		+
symmetrische Replikation	+	-		+

[44] Nur für DB2 und ADABAS-C.

2.5 Frontend-Systeme und Programmierunterstützung

2.5.1 Graphische 4GL-Entwicklungswerkzeuge

2.5.1.1 Datenbankabfragen und -analysen

Alle Produkte bieten für Anwender ohne SQL-Kenntnisse graphische 4GL-Werkzeuge zur einfachen und flexiblen Durchführung von ad-hoc-Abfragen, zur Definition von View-Objekten und zur Erstellung von einfachen Statistiken und Geschäftsgraphiken. Bei ORACLE verteilen sich diese Funktionalitäten auf mehrere Tools einer Produktfamilie (Discoverer/2000), die jedoch voll integrierbar sind. Bei SYBASE sind diese Funktionalitäten in PowerBuilder 4.0 integriert, der seit der Übernahme von Powersoft durch SYBASE als Frontend-Tool für SYBASE-Datenbanken fungiert.

Mit allen Browsing-Tools können Abfragen graphisch dargestellt werden, womit auch eine intuitiver Aufbau der Abfragen per Maussteuerung unterstützt wird. Das Browsing läuft direkt gegen das Data Dictionary, so daß alle relevanten Informationen (Tabellenaufbau, Datentypen, foreign-key-Beziehungen), beim ORACLE Browser 2.0 auch komplexe verteilte Datenstrukturen, automatisch dargestellt und für die Generierung der SQL-Statements genutzt werden. Dabei wird der volle SQL-Sprachumfang ausgenutzt, aber ohne daß SQL vom Benutzer bewußt angewendet werden muß. Abfrageergebnisse können sortiert und mit entsprechenden Formatierungsfunktionen aufbereitet werden.

Beim ORACLE Browser können für weniger erfahrene Anwender individuelle benutzerspezifische Arbeitsumgebungen mit vereinfachten Strukturen und ggf. auch eingeschränktem Zugriff entworfen werden. Über den rein lesenden Datenbankzugriff hinaus sind, in der Extended Edition des Browsers, auch Schemaänderungen möglich. Datenänderungen werden von allen Browsing-Werkzeugen unterstützt. Graphische Abfragen und Ergebnisprotokolle können bei allen Tools, u.a. als Views, abgespeichert werden (bei ORACLE in der Datenbank oder in einem Dateisystem). Beim ORACLE Browser können darüber hinaus auch SQL-Befehle importiert (und dann graphisch aufbereitet) oder exportiert werden, so daß Daten und Abfragen mit anderen Oracle Tools weiterbearbeitet werden können. Dynamic Data Exchange (DDE) und Object Embedding and Linking (OLE 2.0) werden nur vom ORACLE-Browser voll unterstützt. Bei allen Browsern ist ein Zugriff auf Fremddatenbanken (unterschiedlicher Hersteller) über ODBC oder Open Gateways möglich.

Da diese Browsing-Tools eine genaue Kenntnis des Datenmodells voraussetzen, werden von allen Herstellern darüber hinaus Werkzeuge angeboten, die die Erstellung von komplexen View-Objekten und Geschäftsberichten, -graphiken und -analysen erlauben. Mit Hilfe dieser Views lassen sich durch Verbergen relationaler Details der Datenbank und komplexer SQL-Ausdrücke vereinfachte logische Sichten auf die Datenbank definieren, so wie der Anwender sie kennt und benötigt (logische Objekte).

Mit dem ORACLE Data Query 4.0 können logische Objekte („Derived Items“) zu Geschäftsbereichen („Business Areas“) gruppiert werden. Auf der Basis von vordefinierten Verbindungen zwischen derartigen Objekten können mit dem ORACLE-Werkzeug und mit PowerBuilder Detailinformationen (Drill-Down) oder allgemeinere Informationen (Drill-Up) per Maussteuerung abgefragt werden, ohne daß der Anwender Kenntnisse über Tabellen und Spalten besitzen muß. Bestandteil von Derived Items (bei ORACLE) können auch statistische Analysen, die mit Hilfe eines Formeleditors erstellt werden, oder Berichte im Tabellen-, Matrix- oder Master/Detail-Format sein. Änderungen der Daten sind damit jedoch nicht möglich. Das Erstellen von Statistiken mit einem Formeleditor wird von allen Werkzeugen unterstützt.

Abfrageergebnisse können bei ORACLE deklarativ mit Hilfe eines Graphikprogramms (Graphics 2.5) in Form von interaktiven Geschäftsgraphiken (Linien, Balken, Säulen- oder Tortendiagramme) visualisiert werden, die mit Hilfe der integrierten prozeduralen Programmiersprache (PL/SQL) auch mit Drag&Drop-Funktionalität und Ablaufkontrolle versehen werden können. Durch Modifizierung graphischer Attribute (Farbe, Größe, Form etc.) lassen sich Änderungen von Daten widerspiegeln. Darüber hinaus können multimediale Elemente (Bilder, digitale Tonsequenzen) integriert werden. PL/SQL-Code kann dabei zwischen Client und Server aufgeteilt werden (*Application Partitioning*), was die Performance der Anwendung erhöht.

Das Erstellen von Graphiken ist bei ADABAS-D nicht möglich. Bei PowerBuilder müssen interaktive Graphiken in Verbindung mit der Applikationsentwicklung programmiert werden; ein Application Partitioning ist jedoch nicht möglich. Bei INFORMIX können interaktive Graphiken und multimediale Elemente nur über add-on-Bibliotheken von Drittanbietern integriert werden.

2.5.1.2 Reporting

Von allen Anbietern wird die Entwicklung von Datenbankberichten unterstützt.

Das Leistungsspektrum von ORACLE Reports 2.5 reicht von Listenerstellung, Textgestaltung, Serienbrief- und Etikettendruck bis hin zu interaktiven Berich-

ten mit Multimedia- und Drill-down-Funktionalität. Mit Hilfe von PL/SQL können konditionale Berichte erzeugt werden, deren Aussehen in Abhängigkeit von einzelnen Datenmengen -oder werten dynamisch angepaßt werden kann. Auch hier sind Portabilität über verschiedene Window-Plattformen hinweg und die Integration mit anderen ORACLE-Werkzeugen (CASE, Forms) gewährleistet.

Mit PowerBuilder können ebenfalls konditionale Berichte und Berichte mit Drill-down-Funktionalität erzeugt werden; ein Application Partitioning ist jedoch nicht möglich. Bei ADABAS-D und INFORMIX ist dagegen die Integration von Graphik- und Multimedia-Werkzeugen nur über Fremdprodukte möglich. INFORMIX unterstützt darüber hinaus keine drill-down-Funktionalität und keine konditionalen Berichte.

Bei allen Systemen ist die Integration mit dem GUI-Werkzeug und der Datenzugriff zu Fremddatenbanken über ODBC und Open Gateways gewährleistet. Die Einbindung von Windows DLLs ist bei allen Systemen (außer INFORMIX) möglich. OLE2-Container werden nur von ORACLE unterstützt.

2.5.1.3 CASE-Generatoren

Für alle untersuchten Datenbanksysteme gibt es Tools zur Prozeß- und Datenmodellierung; für SYBASE („S-Designor“) und INFORMIX allerdings nur über Drittanbieter.

Der S-Designor, der mit SYBASE ausgeliefert wird, und die Designer/2000-Tools von ORACLE sind so ausgelegt, daß alle Datenbankdefinitionen, inkl. referentielle Integritätsbedingungen, im Entity-Relationship-Modell abgelegt werden können. Die Datenbank kann mit dem Modell synchron geführt werden. Das Design läßt sich beim S-Designor in ein konzeptionelles und physikalisches Modell trennen, so daß beim Übergang vom logischen zum physikalischen Modell alle Definitionen an das Ziel-Datenbanksystem angepaßt werden können. Proprietäre Eigenschaften des Datenbanksystems müssen dann im physikalischen Modell in Form von Triggern nachgearbeitet werden.

Aus diesen Vorgaben können mit dem S-Designor Standard-Bildschirmmasken, mit dem Designer/2000 sogar auch ablauffähige Reports und Forms-Applikationen generiert werden. Bei beiden Systemen ist ein Reverse-Engineering möglich: Änderungen des Datenmodells werden sofort in der Datenbank und den darauf aufsetzenden Applikationen nachvollzogen und umgekehrt (ist bei ADABAS-D nicht möglich). Dadurch ist gewährleistet, daß die Applikationen permanent auf der aktuellen Version der Datenbank auf-

setzen. Bei ORACLE erstreckt sich das Reverse-Engineering auch auf Forms- und Reports-Anwendungen: Änderungen auf der Ebene der Applikationen werden automatisch in den Daten-, Prozeß- und Funktionsmodellen nachvollzogen.

2.5.1.4 Entwicklung von Datenbankoberflächen

Für die Entwicklung graphischer Benutzeroberflächen werden von allen Herstellern proprietäre Werkzeuge angeboten, die eine deklarative oder sogar graphische Anwendungsentwicklung bei gleichzeitiger Ausnutzung der DBMS-Funktionalitäten ermöglichen[45].

2.5.1.4.1 ORACLE Forms 4.5

Herausragendes Merkmal des ORACLE Entwicklungswerkzeuges (Forms 4.5) ist der deklarative Ansatz der Anwendungsentwicklung, d.h. der Modellierung des Anwendungsverhaltens und des Oberflächendesigns per Maussteuerung. Für die Realisierung von Anwendungen kleinerer und mittlerer Komplexität ist keine prozedurale Programmierung erforderlich, da Data Dictionary Informationen (Foreign-Key-Beziehungen, Master-Detail-Beziehungen) voll ausgenutzt werden.

In Forms sind einige objektorientierte Eigenschaften integriert, die dieses Entwicklungswerkzeug allerdings noch nicht zu einem voll objektorientierten Entwicklungssystem machen. Hier ist v.a. die Einführung von „Property-Klassen“ zu nennen, in denen sich Eigenschaften definieren lassen, die für die Objekte der Anwendung („Blöcke“, Elemente der graphischen Oberfläche) gelten sollen. In die Property-Klassen können auch Trigger integriert werden, so daß Objekte derselben Property-Klasse das gleiche Verhalten zeigen. Property-Klassen können hierarchisch angeordnet sein, wobei vererbte Eigenschaften sich vom Entwickler eines Objekts wieder verändern lassen. Das Prinzip der Vererbung ist auch bei Triggern möglich. Alle für ein bestimmtes Ereignis gültigen Trigger können nacheinander ausgelöst werden - unabhängig von der Ebene, auf der sie definiert sind (z.B. Forms-Trigger, Block-Trigger, Objekt-Trigger). Die Reihenfolge der Abarbeitung kann bei der Programmierung festgelegt werden.

Ein weiteres Konzept ist die Definition von Objektgruppen, in die verschiedene Objekte einer Anwendung gebündelt werden können, deren Inhalte an andere

---

[45] Die Software AG bietet für ADABAS-D zwar auch eine eigene Entwicklungsumgebung an („Natural“). Da es zu diesem Produkt jedoch kein Informationsmaterial gibt, wurde auf eine Vorstellung des Tools in diesem Papier verzichtet.

Anwendungen kopiert oder die referenziert werden sollen. Auch diese Methode unterstützt die Wiederbenutzung von Codestrecken und gewährleistet somit eine effizientere Entwicklung. Mit Hilfe der Drag&Drop-Technik können alle Objekte einer Anwendung beliebig verschoben oder kopiert werden (ein Trigger, der auf Objektebene definiert wurde, läßt sich z.B. problemlos auf die Blockebene verschieben). Darüber hinaus lassen sich PL/SQL-Programme per Drag&Drop zwischen Anwendung und Datenbank verschieben. Auf diese Weise kann während der Anwendungsentwicklung festgelegt werden, ob eine bestimmte Funktionalität auf dem Client oder auf dem Server ausgeführt werden soll. Durch dieses Partitionieren von Applikationen ist die Entwicklung von echten Client/Server-Applikationen möglich.

Für die Entwicklung von PL/SQL-Modulen wird eine integrierte graphische Entwicklungsumgebung angeboten (Oracle Procedure Builder), die auch eine hierarchische Darstellung aller Objekte auf Client- und Server-Seite und der Abhängigkeiten zwischen Objekten ermöglicht (Oracle Object Navigator). Mit dem sog. Performance Event Collection System (PECS) ist eine genaue Performance-Analyse einer Anwendung möglich, so daß der Entwickler erkennen kann, welche Operationen welche Ressourcen benötigen. Auf diese Weise lassen sich Performance-Probleme schon während der Entwicklung identifizieren und lösen. Da auch eine Abweichungsanalyse durchgeführt werden kann, läßt sich sogar testen, ob Code-Änderungen negative Auswirkungen auf die Performance haben. Für umfangreiche Anwendungen kann darüber hinaus eine Statistik über die Code-Überdeckung eines Tests geführt werden, wobei jede Zeile des Codes mindestens einmal ausgeführt wird.

Graphics- und Reports-Anwendungen lassen sich „nahtlos“ in Forms-Applikationen einbinden, so daß die Entwicklung integrierter Anwendungen möglich ist. Über die einfache Einbindung von Graphic Charts hinaus ist es möglich, bidirektional Datenströme zwischen Graphics und Forms auszutauschen, so daß z.B. eine Änderung in einer Tabelle sofort in der Graphik angezeigt wird und umgekehrt.

Da alle gängigen Oberflächen-Plattformen unterstützt werden, ist auch die Erstellung portabler Anwendungen möglich. Hinsichtlich der Entwicklung unter Windows verfügt Forms allerdings über eine Vielzahl von Elementen, die spezielle Windows-Funktionalitäten ausnutzen: Object Linking and Embedding (OLE 2.0) wird in vollem Umfang unterstützt. Damit ist es z.B. möglich, ein WinWord-Dokument in eine Forms-Anwendung einzubinden, wobei das Dokument auch innerhalb der Forms-Anwendung mit WinWord bearbeitet werden kann. In ähnlicher Weise wird der Datenaustausch zwischen Forms

und anderen Windows-Anwendungen (Dynamic Data Exchange (DDE)) und die Integration von VBX-Controls (Visual Basic Custom Controls) unterstützt. Eine DDE-Verbindung läßt sich allerdings immer nur von Forms initiieren und kann somit nur Client-Funktionalität haben. Um eine direkte Windows-Steuerung aus einer Forms-Anwendungen heraus zu ermöglichen, ist eine Schnittstelle zum SDK von Microsoft implementiert, deren Funktionen aus jedem PL/SQL-Modul oder über C-User-Exits aufgerufen werden können. Diese Funktionalitäten machen Forms zu einer mächtigen Entwicklungsplattform unter Windows, schränken die Portabilität der Anwendung jedoch entsprechend ein.

Das Multiple Document Interface von Forms erlaubt es, mehrere Forms-Anwendungen parallel und mit nur einem Connect zur Datenbank zu fahren (offenes Transaktionsmodell). Alle Transaktionen laufen dabei parallel nebeneinander her. Gleichzeitig ermöglicht Forms einen Zugriff auf fremde Datenbanksysteme über ODBC. Der Zugang zu Fremddatenbanken kann aber auch über die ORACLE Open Gateway Produkte oder sog. Transaktions-Trigger realisiert werden, d.h. eine offene Schnittstelle, die die Einbindung eigener C-Routinen erlaubt.

2.5.1.4.2 INFORMIX NewEra 1.0 für Microsoft Windows

NewEra ist - wie Forms auch - eine graphische Entwicklungsumgebung, die die visuelle Gestaltung der Oberfläche per Drag&Drop einschließlich Code-Generierung unterstützt. Eine integrierte objektorientierte Programmiersprache ermöglicht die Entwicklung von Klassenbibliotheken, die die Grundkomponenten einer NewEra-Anwendung bilden. Bibliotheken für Bound Controls, Datenbankzugriffe (u.a. über ODBC) und plattformenübergreifende Oberflächenprogrammierung (Windows, Motif) werden mit ausgeliefert. Mit NewEra ist sowohl die Entwicklung wiederverwendbarer Klassenbibliotheken als auch die Integration von fremden C-Funktions- oder C++-Klassenbibliotheken möglich. Die NewEra Language ist allerdings eine High-Level-Programmiersprache, die - wie PowerBuilder und Forms auch - v.a. stark ereignisgesteuerte Anwendungen unterstützen soll.

Die Behandlung von Datenbankzugriffen läßt sich in NewEra mit dem Konzept der *SuperTables* weitgehend automatisieren. Eine SuperTable ist ein „Datenbank-orientiertes Objekt“, in dem Oberflächen- und Datenbankelemente miteinander verknüpft werden. Der zur Durchführung der Datenbankoperationen benötigte SQL-Code wird unter Ausnutzung der im Data Dictionary hinterlegten Informationen (z.B. Join-Beziehungen) automatisch erzeugt. Darüber hinaus können vordefinierte SQL-Sequenzen und Ereignisse

für die Behandlung gängiger Datenbanktransaktionen genutzt werden. Da SuperTables als Klassen implementiert sind, können sie jederzeit erweitert werden (Vererbung, Wiederverwendbarkeit). Ein Repository-System, das alle Informationen über Datenstrukturen und deren Beziehungen, visuelle Attribute u.ä. enthält, unterstützt eine teamorientierte Entwicklung. Über ODBC werden auch andere Datenbankprodukte (wie ORACLE oder SYBASE) unterstützt. Ein Application Partitioning ist möglich.

Im Gegensatz zu Forms und PowerBuilder kann bei NewEra allerdings zwischen zwei Compiler-Systemen gewählt werden: Über den zur Laufzeit interpretierten P-Code hinaus kann auch C-Code erzeugt werden, der von einem C-Compiler (Microsoft Visual C++) zu ausführbaren Programmen kompiliert werden kann, die wesentlich schneller ablaufen als jede interpretierte Version eines Programms. Diese Offenheit und das ausgeprägte Konzept der Klassenbibliotheken machen NewEra vom Konzept her zu einer leistungsstärkeren und flexibleren Entwicklungsumgebung als Forms und PowerBuilder, die - laut Herstellerangaben - die Entwicklung von komplexeren zeitkritischen Anwendungen im Auge hat und sich eher an professionellere Softwareentwickler wendet. Dem widerspricht allerdings, daß NewEra kein offenes Transaktionsmodell kennt, d.h. es ist in einer NewEra-Anwendung immer nur eine Transaktion zur Zeit gegen die Datenbank möglich.

2.5.1.4.3 PowerBuilder 4.0 (SYBASE)

Für SYBASE gibt es zwar keine proprietäre Entwicklungsumgebung, seit der Übernahme von Powersoft wird jedoch der PowerBuilder als 4GL-Tool für SYBASE-Datenbanken angeboten. PowerBuilder ist ebenfalls ein graphisches Werkzeug mit objektorientierten Eigenschaften, das die Entwicklung ereignisgesteuerter Datenbankwendungen per Drag&Drop unterstützt. Da PowerBuilder jedoch ein Drittanbieter-Produkt ist, ist zu erwarten, daß Eigenschaften des Datenbanksystems nicht voll ausgenutzt werden. Der Umfang der automatischen Code-Generierung fällt dann dementsprechend geringer aus. Der Entwicklungsaufwand mit PowerBuilder dürfte deshalb größer sein als mit den anderen proprietären Tools.

PowerBuilder unterstützt allerdings die automatische Generierung von Join-Bedingungen, den bidirektionalen Datenaustausch mit eingebundenen Graphiken, das Einbinden von Windows DLLs, Windows SDK-Funktionen und VBX-Controls, OLE2 Container jedoch nicht, und den Datenzugriff über ODBC und Gateways. Ein Application Partitioning ist jedoch nicht möglich.

2.5.1.5 Das Problem der „richtigen“ Frontend-Strategie

Die Wahl der richtigen Frontend-Strategie (proprietäres oder DBMS-unabhängiges Werkzeug) hängt im wesentlichen von drei Faktoren ab:

- Die erforderliche bzw. gewünschte Entwicklungsperformance
- Die Komplexität der zu modellierenden Anwendungsprobleme
- Die Portabilität und Performance des Softwaresystems zur Lösung dieser Probleme.

Für eine Produktivitätssteigerung bei der Systementwicklung ist eine deklarative oder sogar graphische Entwicklungsumgebung, die DBMS-Funktionalitäten „optimal“ ausnutzt, sicherlich sinnvoll, so daß Werkzeuge, die mit dem DBMS ausgeliefert werden, zunehmend an Entscheidungsrelevanz gewinnen[46]. Inwiefern sich allerdings bei der Entscheidung für ein proprietäres Entwicklungstool die (nicht gewollte) DBMS-Abhängigkeit der Anwendungsprogramme verfestigt, hängt dann davon ab, wie eng das Werkzeug mit den Spezifika des DBMS verbunden ist. Dies ist bei Forms und NewEra sicherlich in weitaus größerem Maße als bei PowerBuilder oder C++-Werkzeugen der Fall. Forms und NewEra bieten zwar auch einen Datenzugriff auf fremde DBMSe über eine normierte Schnittstelle (ODBC), Forms-Anwendungen sind darüber hinaus auf Motif- und Macintosh-Plattformen portierbar, grundsätzlich muß bei allen proprietären Werkzeugen jedoch damit gerechnet werden, daß sich die Interoperabiliät der Anwendung auf das zugrundeliegende DBMS beschränkt. Dies spricht zunächst für ein DBMS-unabhängiges Werkzeug wie PowerBuilder, mit dem ebenfalls eine deklarative Entwicklung von (einfachen) Datenbankoberflächen möglich ist.

Ein grundsätzliches Problem bei der Entwicklung komplexerer Anwendungen ist jedoch der Übergang von der Prototyping-Phase in die Implementierungs- und Einsatzphase. Hier wird sich zeigen, daß C++-basierte Tools aufgrund ihrer Mächtigkeit und offenen Architektur sowie der Möglichkeit, sehr performante Programme zu erzeugen, längerfristig die professionelleren Werkzeuge sind.

Die Frage der DBMS-Abhängigkeit der Anwendungsprogramme hängt - neben der DBMS-Bindung der Werkzeuge - jedoch auch von der Offenheit der gewählten Schnittstelle bzw. den Connectivity-Konzepten ab, die bei der

---

[46] Vgl auch: Bauer (1995b), Borchers (1994).

Anwendungsentwicklung implementiert werden. Da die Abhängigkeit von der verwendeten Entwicklungsumgebung im allgemeinen größer ist als die von einem (relationalen) DBMS, liegt die eigentlich strategische Entscheidung bei der Wahl der richtigen Frontend-Strategie deshalb nicht nur in der Wahl der Werkzeuge, sondern auch in der Wahl der Schnittstelle.

2.5.2 Programmierschnittstellen und Connectivity-Konzepte

Die Frage der DBMS-Unabhängigkeit der Anwendungsprogramme betrifft im wesentlichen die Programmierschnittstelle, die die Kommunikation zwischen Programm und DBMS festlegt. Für den Datenbankzugriff sind grundsätzlich folgende Strategien denkbar:

- Die verstärkte Verlagerung von Datenbankzugriffs-Routinen auf den Server (in Form von Triggern und Stored Procedures)
- Die Nutzung einer proprietären Programmierschnittstelle
- Die Verwendung von standardisierten Schnittstellen (wie ODBC).

Dreh- und Angelpunkt ist im Hinblick auf die DBMS-Unabhängigkeit der Applikationen die Offenheit der verwendeten Schnittstelle. Um Anwendungsprogramme auf unterschiedliche Datenbanksysteme zugreifen lassen zu können, ist eine „Middleware“-Komponente erforderlich, die sich zwischen die Client-Anwendung und das DBMS schiebt, so daß die Kommunikation zwischen beiden Teilen für den Anwendungsentwickler unsichtbar und damit die Portabilität der Applikation gewährleistet ist. Diese offene Architektur wird grundsätzlich nur von der dritten Variante zur Verfügung gestellt[47]. Da die Reichweite der ODBC-Schnittstelle jedoch auf SQL-Funktionen eingeschränkt ist, die allen DBMS-Plattformen gemeinsam sind, und darüber hinaus die Portabilität einer ODBC-Anwendung über verschiedene Betriebssystem-Plattformen hinweg wegen der Beschränkung auf Microsoft Windows nicht gewährleistet ist, wird im folgenden auch auf die ersten beiden Alternativen eingegangen.

2.5.2.1 Trigger und Stored Procedures

In einer Client/Server-Umgebung stellen Datenbankzugriffe über das Netz ein erhebliches Performance-Problem dar. Eine Möglichkeit, dieses Problem zu lösen, besteht darin, einen Teil der Anwendungslogik von den Anwendungs-

[47] Vgl. zum Thema DBMS-Unabhängigkeit: Bauer (1995), Bauer (1994a), Bauer, M. (1994b), Nußdorfer (1994).

programmen in das DBMS zu verlagern (Application Partitioning), so daß die datenbezogene Verarbeitung auf dem Server abläuft und das Anwendungsprogramm nur noch die Interaktion mit dem Benutzer übernimmt. Hierfür gibt es zwei Varianten, die auch von allen untersuchten Datenbanksystemen unterstützt werden:

- Trigger: Konsistenzregeln, die die gültigen Zustände der Daten beschreiben und mit der Tabellendefinition implementiert werden.
- Stored Procedures: Prozedurale SQL-Routinen, die beim DB-Server in kompilierter Form vorgehalten und durch einen einzigen SQL-Befehl oder einen *Remote Procedure Call* (RPC) aufgerufen werden.

Bei den Stored Procedures sollten folgende Merkmale gegeben sein, die von ORACLE, ADABAS-D und INFORMIX im wesentlichen abgedeckt werden:

- Prozedurale Vollständigkeit, Rekursion
- Nutzung von Stored Procedures als SQL-Funktionen innerhalb von SQL-Befehlen
- Nutzung von dynamischem SQL innerhalb von Stored Procedures
- Erweiterbarkeit, d.h. Einbindung von C-Programmen (INFORMIX nicht): Dadurch ist es möglich, beliebig komplexe Operationen als Stored Procedures zu entwickeln und diese als SQL-Funktionen zu nutzen. Dies macht den SQL-Funktionsvorrat nahezu beliebig erweiterbar. Bei ORACLE kann diese Technologie auch bei der parallelen Query-Verarbeitung eingesetzt werden.

Für das Konzept des Application Partitioning sprechen in einer Client/Server-Umgebung folgende Argumente[48]:

- Der Kommunikationsaufwand zwischen Client und Server vermindert sich erheblich, was sich positiv auf die Performance auswirkt, da sich die Anzahl von SQL-Befehlen, die über das Netz versandt werden müssen, deutlich reduziert.
- Allgemeine Bereiche der „Geschäftslogik“, die immer wieder benutzte Datenverarbeitungsfunktionen darstellen, werden an einer zentralen Stelle hinterlegt und gepflegt, nämlich in der Datenbank selbst. Dadurch reduziert sich nicht nur der Entwicklungs- und Wartungsaufwand, sondern es stehen

[48] Vgl. hierzu auch: Bauer (1994a), Bauer (1994b), Bauer (1995), Herzog/Lang (1995), Demuth (1994).

die Funktionen auch allen Anwendungen bzw. Anwendungsentwicklern gemeinsam zur Verfügung. Einmal implementierte Routinen können in neuen Anwendungen wiederverwendet werden. Trigger laufen darüber hinaus automatisch für alle Programme ab, die auf die entsprechenden Daten zugreifen, so daß die Datenintegrität einheitlich sichergestellt wird.

- Die Client-Anwendungen werden schlanker.

Ein Nachteil dieses Verfahrens besteht allerdings darin, daß es für Trigger und v.a. für Stored Procedures noch keinen Standard gibt[49]. DBMS-Routinen müssen deshalb in der proprietären Programmiersprache des jeweiligen DBMS implementiert werden. Damit erhöht sich die DBMS-Abhängigkeit auf der Datenbankebene.

2.5.2.2 Proprietäre Schnittstellen

Als proprietäre Schnittstellen bieten sich an[50]:

- Proprietäre C- bzw. C++-Bibliotheken für Datenbankzugriffe:

  Bietet sehr performante Zugriffe und gewährleistet darüber hinaus ein hohes Maß an Portabilität bzgl. der Betriebssystem-Plattformen, aber auch ein hohes Maß an DBMS-Abhängigkeit.

- Embedded SQL (ESQL):

  Bietet ebenfalls ein hohes Maß an Effizienz und Portabilität (s. ANSI 19989 ESQL-Standard). Bei einem DBMS-Wechsel muß das Programm (lediglich) von einem anderen Precompiler erneut umwandeln werden, so daß Embedded SQL weniger DBMS-Abhängigkeit bedeutet als der Einsatz einer proprietären C-Bibliothek.

- Gateways:

  Bei unterschiedlichen Datenbanksystemen wird hier nur die formale Aufrufschnittstelle angepaßt, funktionale und semantische Differenzen zwischen den beteiligten Systemen werden aber nicht ausgeglichen, so daß bei einem Einsatz von Gateways für jedes unterschiedliche DBMS jeweils eine spezielle Software benötigt wird (s. auch Kapitel 2.4.5).

Für die Entwicklung von ESQL-basierenden Applikationen wird von allen Herstellern ein entsprechender Precompiler angeboten, der die Verwendung

---

[49] Beim SQL3-Standard ist allerdings auch eine einheitliche prozedurale Programmiersprache für relationale Datenbanken in der Diskussion (vgl. Bauer (1995)).

[50] Vgl. auch: Ammon (1994), Nußdorfer (1994).

von Embedded-SQL-Konstrukten in C-Programmen mit vollem Cursor-Support unterstützt. Zur Verbesserung der Performance kann eine Array-Schnittstelle für statisches und dynamisches SQL genutzt werden. Bei ORACLE ist die Integration mit Forms und Reports vollständig gewährleistet; Laufzeitstatistiken ermöglichen eine Performance-Analyse der Anwendung.

Darüber hinaus bietet ORACLE mit der OLE2-basierten Schnittstelle Oracle Objects für OLE 1.0. ein Middleware-Konzept an, das zwischen Anwendung und Datenbank gehängt werden kann, zugleich aber hohe Performance und volle Ausnutzung der Oracle7-Funktionalitäten vereint. Dieses Werkzeug fungiert als universelle Oracle-Schnittstelle, über die OLE-fähige Windows-Applikationen (wie WinWord oder Excel) auf lokale oder verteilte Oracle-Datenbanken zugreifen können. Um auch den Zugriff von eigenen C++-Applikationen über Oracle Objects für OLE zu ermöglichen, sind im Lieferumfang C++-Klassenbibliotheken für Visual C++ enthalten. Der Nachteil dieses Konzepts ist natürlich die Einschränkung auf Oracle-Datenbanken.

2.5.2.3 Offene Schnittstellen (ODBC)

Anders als bei Verwendung einer C-Bibliothek oder ESQL, wo für eine Anwendung, die auf unterschiedliche Datenbanksysteme zugreifen soll, auch mehrere Schnittstellen implementiert werden müßten, schirmt eine Standardschnittstelle die Applikation von den Unterschieden der Datenbanksysteme und der Netzsoftware ab. Bei derartigen generalisierten Schnittstellen enthält das Anwendungsprogramm neutrale, standardisierte SQL-Befehle, die von einem Umsetzer in die Befehlsstruktur des jeweiligen DBMS gebracht werden. Das Konzept der Standardschnittstelle für Datenbankzugriffe bietet genau die Vorteile, die für die Realisierung offener Software erforderlich ist:

- Plattformenunabhängigkeit und Wirtschaftlichkeit: Mit Standardschnittstellen entwickelte Module können ohne Neuprogrammierung für unterschiedliche DBMS-Plattformen eingesetzt werden, d.h. die Anwendung bleibt unverändert.
- Interoperabilität: Bei verteilten, plattformenübergreifenden Anwendungen erfolgt der Datenaustausch über offene Schnittstellen.

Eine solche Standardschnittstelle ist ODBC, eine C-API-Spezifikation für einen einheitlichen Zugriff von Windows-Anwendungen auf verschiedene Datenbanksysteme[51].

---

[51] Zu ODBC vgl. Kuppinger (1995), Brauchle (1995), Demuth (1994), Nußdorfer (1994).

Bei ODBC erfolgt der Zugriff über separate Programme (Datenbanktreiber, in der Regel als DLLs), die die SQL-Befehle der Anwendung im jeweiligen Dialekt an die Server-Datenbank weitergeben, so daß Eigenheiten der jeweiligen Datenbanksysteme vor der Anwendung verborgen werden. Dieses Konzept umfaßt auch die Netzwerkprotokolle, d.h. die Anwendung spricht diese Protokolle nicht mehr nativ, sondern über eine genormte Schnittstelle an. Die Anwendung ist damit mit beliebigen Protokollen ablauffähig. Ein ODBC-Treiber-Manager, der der Applikation als DLL hinzugefügt wird, lädt automatisch den entsprechenden Datenbanktreiber, über den dann der Datenbankzugriff erfolgt. Die Applikation verwendet intern die Aufrufe des ODBC-APIs. Dadurch wird der Datenbankzugriff auf der Client-Seite vereinheitlicht. Dieses Konzept gewährleistet eine hohe DBMS- und Netzwerk-Unabhängigkeit und Interoperabilität (Datenaustausch regeln die ODBC-Treiber). Bei einem Umstieg auf einen anderen Datenbankserver muß somit lediglich der Datenbanktreiber ausgetauscht werden. Die Anwendungsprogramme selbst müssen nicht verändert werden. Für alle untersuchten Datenbanksysteme gibt es einen ODBC-Treiber mit TCP/IP-Unterstützung und Cursor-Support.

ODBC ist allerdings grundsätzlich nur dann sinnvoll, wenn von einem bestimmten Client aus auf mehrere unterschiedliche Datenquellen zugegriffen werden muß. Bei betrieblichen Anwendungen, die nur für den internen Gebrauch entwickelt werden und wo das DBMS, auf das zugegriffen werden soll, in der Regel feststeht, ist der Einsatz einer ODBC-Schnittstelle nicht unbedingt erforderlich[52]. Darüber hinaus muß bei Release-Wechseln des DBMS ein funktionierender Treiber am Markt gefunden werden. Dieses Problem hat man bei Verwendung einer proprietären Schnittstelle (wie C-Bibliothek oder ESQL) nicht.

Aus softwaretechnologischen Gründen ist jedoch auch in diesen Situationen der Einsatz von ODBC sinnvoll:

- Grundsätzlich kann nicht davon ausgegangen werden, daß immer mit dem gleichen oder nur mit einem DBMS gearbeitet wird. Die Software sollte daher offen für die Arbeit mit unterschiedlichen Datenbanksystemen sein.
- Bei Verwendung einer normierten Schnittstelle müssen die Entwickler weniger über die Eigenarten verschiedener Datenbanksysteme wissen.

[52] Vgl. Kuppinger (1995).

Aufgrund dieser Vorteile hat sich ODBC als de-facto-Standard für Schnittstellen zwischen Client und Server etabliert. Konzeptbedingte Nachteile von ODBC sind:

- Die Beschränkung auf Windows (inzwischen gibt es allerdings auch ODBC-Treiber für Unix- und OS/2-Clients, so daß zu erwarten ist, daß Standardschnittstellen wie ODBC über Plattformgrenzen hinweg Verbreitung finden werden).
- Performance-Verluste wegen der zusätzlichen Übersetzungsschicht: Die Performance von ODBC-Treibern variiert u.U. erheblich. Vor einer Entscheidung für ODBC sollten daher die in Frage kommenden ODBC-Treiber getestet werden.
- Unterschiedliche Implementierungslevel: Core Level (nur die minimal erforderlichen Datenzugriffsfunktionen), Level 1 (Funktionen, mit denen nur Teile von Spaltenwerten gelesen werden), Level 2 (u.a. Verwendung von Cursorn). Bei der Auswahl der Treiber muß daher auf die verwendete SQL-Implementierung geachtet werden. Darüber hinaus sollten die Treiber als DLLs und nicht als TSR-Programme realisiert sein.
- Eingeschränkte Funktionaliät wegen der Beschränkung auf den SQL-Standard, d.h proprietäre, aber wichtige SQL-Funktionen können nicht genutzt werden. Eine Ausweichstrategie bildet die Nutzung der Funktion „pass through“, mit der SQL-Statements nativ an den Datenbankserver durchgereicht werden können. Damit ginge die angestrebte DBMS-Unabhängigkeit allerdings wieder verloren. Hinzu kommt die Unsicherheit über Weiterentwicklung des Standards.
- Kein gleichzeitiger Zugriff auf mehrere Datenbanksysteme.
- Kein Array-Interface.
- Das Call-Level-Interface-Konzept erfordert die Auflösung des SQL-Befehls zur Laufzeit (dynamisch). Dies bereitet manchen Datenbanksystemen Schwierigkeiten.

2.5.2.4 Die Wahl der „richtigen“ Programmierschnittstelle

Die Entscheidung, ob eine proprietäre oder offene Schnittstelle verwendet werden sollte, hängt nicht nur von der Beurteilung der Funktionalität und Performance der Schnittstelle und der Zukunftssicherheit des Konzepts (s. Normierungsdiskussion) ab, sondern auch von dem gewünschten Interoperabilitätsgrad der Anwendung. ODBC ist in dem Maße erforderlich, in dem die Portierbarkeit bzw. die Unabhängigkeit der Anwendung von der technischen

Umgebung im Vordergrund des Interesses steht. Dies entspricht im wesentlichen der Situation eines Software-Hauses, die Standardsoftware (für Windows!) entwickelt. Sind jedoch Performance und Nutzung proprietärer Funktionen zur Realisierung bestimmter Anforderungen wichtiger, wie es bei rein betrieblichen Anwendungen in der Regel der Fall ist, dann könnte sich ODBC als Rückschritt erweisen[53]. Bei komplexen industriellen Anwendungen bietet sich daher eher die Entwicklung mit C++-basierten Werkzeugen (C++-Klassenbibliothek) und Embedded SQL an.

Darüber hinaus ist die stärkere Nutzung des Application Partitioning von besonderem Interesse. Mit der Verwendung dieses Konzepts erhöht sich zwar die DBMS-Abhängigkeit auf der Datenbankebene, die Oberflächen-Programme selbst allerdings werden in dem Maße DBMS-unabhängiger, in dem sie sich ausschließlich auf den ereignisorientierten Teil der Applikation konzentrieren und Datenbankzugriffe (und somit auch viel von der oben beschriebenen Connectivity-Problematik) möglichst aus den Applikationen herausgenommen und in die Datenbank verlagert werden.

Die Frage ist hier deshalb nicht, ob man Stored Procedures nutzen sollte oder nicht, sondern eher, wo die Grenze zwischen Client und Server zu ziehen ist, d.h. das Problem der DBMS-Unabhängigkeit ist im wesentlichen eine Frage des Software-Designs (was schiebt man auf den Server und was nicht). Hier sollte darauf geachtet werden, daß (nach Möglichkeit) nur applikationsunabhängige Routinen als Stored Procedures implementiert werden, d.h. elementare Bausteine, die für alle (denkbaren) Anwendungen gleichermaßen gelten und deshalb eigentlich mehr zur Datenbank selbst gehören als zur Applikation.

Im Hinblick auf die Portabilität der Anwendungsprogramme stellt sich darüber hinaus die Frage, ob die Variante des „fetten“ Clients eine wirkliche Alternative ist, da - abgesehen von den genannten Performance-Problemen und dem höheren Entwicklungsaufwand bei mehreren Client-Anwendungen - die Portabilität hier nur bei vollständiger Beschränkung auf den SQL-Standard und normierte Schnittstellen (ODBC) gewährleistet ist. Dies ist angesichts des geringen Umfangs des SQL-Standards jedoch ein Problem für sich. Da eine Nach-Implementierung nicht-standardisierter SQL-Funktionen im Anwendungsprogramm jedoch erheblich aufwendiger ist, als die Änderung eines SQL-Befehls bei einem Umstieg auf ein anderes DBMS, verbietet sich geradezu die Außerachtlassung proprietärer SQL-Funktionen.

---

[53] Vgl. hierzu auch Nußdorfer (1994), Kuppinger (1995).

Für das IZ entscheidet sich diese Problematik letztendlich an der Frage, für wen und mit welchem Ziel Software im IZ entwickelt werden soll (domainenunabhängige Software oder Datenbankoberflächen zum „Hausgebrauch“). Da im wesentlichen letzteres der Fall ist, sollte das Konzept des Application Partitioning auf jeden Fall genutzt und deshalb vom DBMS auch unterstützt werden. Um hier den Umstellungsaufwand bei einem eventuellen DBMS-Wechsel gering zu halten, sollte DBMS-spezifischer Code dann allerdings leicht zu lokalisieren und sauber dokumentiert sein.

2.5.3 Workflow-Unterstützung und Internet-Anbindung

Im Bereich Workflow-Unterstützung werden von den Herstellern unterschiedliche Konzepte angeboten: NewEra stellt eine Klassenbibliothek für Lotus Notes (marktführendes Produkt im Groupware-Bereich) zur Verfügung. Für ADABAS-D gibt es ein eigenes Workflow-Produkt („Natural Workflow“). ORACLE und Lotus planen eine Kooperation, die auf eine Integration von Lotus Notes-Anwendungen mit Oracle-Datenbanken abzielt.

Für ORACLE, ADABAS-D und INFORMIX werden WebServer angeboten, mit denen die Datenbanken im World Wide Web zur Verfügung gestellt werden können.

2.5.4 Tabellarische Zusammenfassung

	ORACLE	ADABAS-D	SYBASE	INFORMIX
Graphische Frontend-Werkzeuge				
Administration				
Administrationstool	+	+	+	+
Leistungsmonitor	+	+	+	+
Lade-Utility	+	+	+	+
Integrierte Browsing- und Analyse-Werkzeuge				
Graphisch unterstützte Durchführung von ad-hoc-Abfragen	+	+	+	+[54]
Graphische Abbildung der Abfragen	+	+	+	+
Ausnutzung von foreign key-Beziehungen	+	+	+	-[55]

[54] „ViewPoint“.
[55] Hierzu müssen vom DBA entsprechende Views definiert sein.

	ORACLE	ADABAS-D	SYBASE	INFORMIX
Volle SQL-Unterstützung	+	+	z.T.	+
Schema- und Dateneditor für schreibenden Zugriff	+/+	+/+	-/+	+/+
Sortieren der Abfrageergebnisse	+	+	+	+
Formatierung der Abfrageergebnisse	+	+	+	+
DDE / OLE	+/+	+/-	/+	-/-
Datenzugriff über ODBC	+	+	+	+
Datenzugriff über Open Gateways	+	+	+	+
Speichern der Abfragen als Views	+	+		+
Erzeugen graphischer Abfragen aus SQL-Befehlen	+	-		-
Portabilität: MS-Windows 3.1 / Motif / Apple Macintosh	+/+/+	+/-/-	+/+/+	+/+/?
Erstellung von View-Objekte / Gruppierung von logischen Objekten	+/+	+	+/	+/+
Drill-Down / Drill-Up (bei View-Objekten)	+/+		+/+	-/-
Erstellen von Statistiken (Formeleditor)	+	+	+	+
Erstellen von Graphiken / Interaktive Graphiken / Multimedia-Fähigkeit	+/+/+	-	+/-[56]/+	+/+/-[57]
Integration mit anderen Entwicklungstools	+		+[58]	
Reporting				
Listendruck	+	+	+	+
Textgestaltung / Etiketten / Serienbriefe	+/+/+	+/-/-	+/+/+	+/
Interaktive Berichte / drill-down-Funktionalität	+/+	+/+	-[59]/+	+/-
Konditionale Berichte (integrierte Programmiersprache)	+	+	+[60]	-
Integration mit Graphik-Werkzeug / Multimedia-Fähigkeit	+/+	add-on	+/+	add-on
Integration mit CASE-Werkzeug	+		+[61]	add-on
Integration mit GUI-Werkzeug	+	+	+	+

[56] Muß innerhalb der Applikation programmiert werden.
[57] Nur add-on-Bibliothek von Drittanbieter.
[58] In PowerBuilder integriert.
[59] Muß innerhalb der Applikation programmiert werden.
[60] Muß innerhalb der Applikation programmiert werden.
[61] Integration mit „S-Designor“ (Drittanbieter-Produkt).

	ORACLE	ADABAS-D	SYBASE	INFORMIX
Einbinden von Windows DLLs	+	+	+	-
OLE2 Container	+	-	i.E.	-
Application Partitioning	+	-	-	+
Portabilität (MS-Windows / Motif / Apple Macintosh)	+/+/+	+/-/-	+/+/+	+/+/-
Zugriff auf Fremddatenbanken üb. ODBC / Open Gateways	+/+	s.o.	+/+	+[62]/+
Integrierte CASE-Generatoren (unter Windows)				
Prozeßmodellierung (Datenflußpläne)	+	+	+[63]	-[64]
Datenmodellierung (Entity-Relationship-Diagramme)	+	+	-	-
automatische Generierung Datenbank / Oberfläche / Reporting	+/+/+	+	+/+/-[65]	-
Reverse Engineering / auch von Applikationen	+/+	-	+/-	-
Integration anderer DBMS (Trennung log.u physikal. Modell)			+	-
Integrierte GUI-Entwicklung (unter Windows)				
Deklarative Anwendungsentwicklung / Bound Controls	+/+	+/+?	+/+	+/+[66]
Debugger	+	-	+	+
Erweiterbarkeit (User Exits, C-Schnittstelle)	+	+	+	+
Automatische Generierung von Join-Bedingungen	+	+	+	+
Offenes Transaktionsmodell	+	+	-	-[67]
Drag&Drop	+	+	+	i.E.
Einbindung v. Graphiken/bidirektionaler Datenaustausch	+/+	+	+/+	+/
Objektorientierung / Vererbung	z.T.	i.E.	+/z.T.	+/+
Inkrementelles Kompilieren / Generierung von C-Code	+/-	+/-	+/i.E.	+/+
ODBC / Open Gateways	+/+	+	+/+	+/
Einbinden von Windows-DLLs	+	+	+	+
Einbinden von Windows SDK-Funktionen	+	+	+	+
Einbinden von VBX-Controls	+	+	+	+

[62] S. ViewPoint.
[63] Nur Drittanbieter-Produkt („S-Designor“).
[64] Nur add-on-Bibliothek von Drittanbieter.
[65] In Verbindung mit PowerBuilder.
[66] Klassenbibliothek.
[67] Es kann immer nur *eine* Transaktion zur Zeit realisiert werden.

	ORACLE	ADABAS-D	SYBASE	INFORMIX
OLE2 Container	+	+	i.E.	i.E.
Plattformen-Unabhängigkeit (Portabilität)	+	+	+/-/-	+
Online-Hilfesystem	+	+	+	+
Integration mit CASE-/Graphik-/und Reportwerkzeugen	+/+/+	+/+	+	add-ons
Performance Analyse von DB-Prozeduren	+		?	+
Application Partitioning	+	+	-	+
Programmierunterstützung und -schnittstellen				
Stored Procedures / Arrays / Rekursion / Dynamic SQL	+/+/+/+	+/	+/	+/+/+/+
Message-basierte RPCs	+	+	+/+	
Nutzung v. Stored-Procedure-Funktionen i. SQL-Befehlen	+	+		+
Einbindung C-Programme in Stored Procedure-Routinen	+[68]	+		-
Embedded SQL-Precompiler für C / Dynamic SQL/Optimierung	+/+/+	+/+/	+	+/+/
Array-Schnittstelle (Array als Hostvariable)	+	+		+
ODBC (mit TCP/IP-Unterstützung)/Cursor Support/DLL	+/+?/+	+	+/?/+	+/+/+
OLE2-basierte Schnittstelle	+	-?	-?	-
C++-Klassenbibliothek für Datenbankzugriffe	+	C	C	C
Cursor-Support (Rollende Cursor)	+	+	+	+
WWW-Unterstützung	+	+		+
Workflow-Unterstützung	+	+[69]		+[70]
PowerBuilder-Unterstützung	+	+	+	+

2.6 Marktposition und Support

Neben den funktionalen Merkmalen der Datenbanksysteme sind auch strategische Überlegungen wie Plattformenunabhängigkeit, Marktbedeutung, Kundennähe und Support von besonderer Entscheidungsrelevanz.

ORACLE und INFORMIX unterstützen alle gängigen Hardware-Architekturen und Betriebssystemplattformen (v.a. UNIX). ORACLE ist in Deutschland dar-

[68] Prozedurale Gateway-Entwicklungsumgebung.
[69] Eigenes Produkt: „Natural Workflow“.
[70] Klassenbibliothek für Lotus Notes.

über hinaus das meistinstallierte Datenbanksystem für die im IZ favorisierte Hardware-Plattform (SNI).

ORACLE ist eindeutig der Marktführer und als solcher auch am stärksten in allen einschlägigen Normierungsgremien vertreten (ANSI, ISO, OMG, X-OPEN). Diese herausragende Marktbedeutung ORACLEs zeigt sich auch daran, daß es unter den öffentlichen Einrichtungen im Bonner Raum eine Reihe von ORACLE-Anwendern gibt, die als Kooperations- oder zumindest als Gesprächspartner für das IZ von Interesse sind. Und auch bei den GESIS-Partnern sind Bestrebungen erkennbar, ORACLE als Datenbanktechnologie einzusetzen.

Von allen ORACLE-Anwendern, die der Autor persönlich aufgesucht hat, wurde - unabhängig voneinander - der Support von ORACLE in höchsten Tönen gelobt. Dies hat sich auch in den Gesprächen und v.a. im Zusammenhang mit der Vorbereitung eines ORACLE-Workshops im IZ bestätigt. ORACLE war hier ohne weiteres bereit (und in der Lage), für eine typische Problemstellung des IZ (Kapitelverwaltung und Registererstellung für Themendokumentationen) in relativ kurzer Zeit eine mit Forms und Reports realisierte Datenbankapplikation auf der Basis von IZ-Daten vorzustellen. ORACLE hat darüber hinaus an beiden IZ-Standorten (Bonn, Berlin) eine Niederlassung, so daß bei diesem Anbieter auch eine große Kundennähe gegeben ist.

Bei SYBASE gestaltete sich die Kooperation dagegen als außerordentlich schwierig. Hier war es zum größten Teil nicht einmal möglich, die benötigten Informationen für eine Beurteilung des Produkts zu bekommen. Der vom Autor verfaßte Fragenkatalog mußte von der Düsseldorfer Niederlassung an einen Distributor im süddeutschen Raum weitergegeben werden, der dann wegen des Urlaubes eines Supporters (!) schließlich auch nicht in der Lage war, die gestellten Fragen zu beantworten. Da der Autor darauf verzichtet hat, den Herstellern „nachzulaufen“, konnten viele Fragen zu SYBASE deshalb nicht beantwortet werden.

ADABAS-D hat im Gegensatz zu den anderen Produkten eine ausgesprochen geringe Marktbedeutung. Die Software AG war nicht in der Lage, zu Anzahl der Installationen und Umsatzentwicklung bzgl. ADABAS-D auch nur irgendwelche Zahlen zu nennen.

INFORMIX hat, wie ORACLE auch, eine große Marktbedeutung und ebenfalls eine Niederlassung in Bonn, so daß dieser Hersteller als Gesprächspartner interessant bleibt.

2.6.1 Tabellarische Zusammenfassung

	ORACLE	ADABAS-D	SYBASE	INFORMIX
Hardware-Architekturen				
Massiv-parallele Systeme	+	+		+
Lose-gekoppelt (Cluster-)Systeme	+	?		+
Symmetrische Multi-Processing	+	+	i.E.	+
Single-Processing Systeme	+	+		+
Betriebssystem-Plattformen				
Unix	+	+	+	+
Windows NT	+	+		+
Marktposition und Support				
Support / Kundennähe	++/++	/+	/+	+/+
Manpower (Zahl Entwickler)	2000	300		2200
Beteiligung an Normierung	+	nur OMG		?
Anzahl Installationen auf UNIX (BRD)	?	k.A.		100.000/
Anzahl Installationen auf UNIX (weltweit)	?	k.A.		800.000
Umsatzentwicklung 1995 in Mio. US$ (BRD / % Zuwachs)	310/+40	k.A.		?
Umsatzentwicklung 1995 in Mio. US$ (weltweit / % Zuwachs)	2967/+48	k.A.		480/

3 Fazit

3.1 Funktionale und strategische Vorteile von ORACLE

Auf dem Hintergrund der Anforderungen und Produktbeschreibungen sprechen folgende entscheidungsrelevante Aspekte für einen Einsatz der Datenbanktechnologie von ORACLE im IZ:

Funktionale Vorteile:

- Integrierte Textretrievalfunktionalität (TextServer):
  Neben dem ansonsten hohen Grad an Relationalität hinsichtlich der DDL- und DML-Funktionalität bietet ORACLE als *einziger* Anbieter eine integrierte Lösung für die Behandlung von großen Textdaten in einer ORACLE-Datenbank. Dieses Konzept erlaubt nicht nur die Recherche in Textbeständen mit den gängigen Textretrievalfunktionen, sondern auch eine kombinierte Suche in strukturierten und unstrukturierten Datensätzen. Gleichzeitig werden alle Sicherheitskonzepte des Datenbanksystems für eine TextServer-Anwendung zur Verfügung gestellt. Eine integrierte Textretrievalfunktionalität ist sowohl für gegenwärtige Bedarfe des IZ (s. Recherche in den Abstracts), v.a. aber für künftige Anforderungen interessant, die in Richtung *Information Warehouse* und Integration der Textdatenbestände der GESIS gehen. Eine der wichtigsten strategischen Konzepte von ORACLE für die nächsten Versionen ist die volle Integration des TextServers (bisher „nur“ eine Zusatzkomponente) in den Datenbank-Kernel. Da dieses Werkzeug jedoch eine neuere Entwicklung ist, gibt es bisher wenig Erfahrungswerte insbesondere bzgl. des Performance-Verhaltens des TextServers in der Anwendung, so daß hier ganz grundsätzlich zunächst mit Schwierigkeiten gerechnet werden muß.
- Hohe Skalierbarkeit und ausführliches Tuning-Konzept:
  Die dynamische Lastprofilkonfiguration während der Laufzeit, die Ausnutzung des Cache und die parallele Query-Technologie sind bei ORACLE ausgesprochen performance-günstige Faktoren. Grundsätzlich ist Performance jedoch eine Frage der Gesamtumgebung (Hardware-Architekur des Servers und seine Bestückung, LAN-Performance, Ausstattung der Client-PCs, Performance der Client-Applikationen).
- Hohes Maß an Datensicherheit, Stabilität und Verfügbarkeit der Datenbank:
  Ausführliche Behandlung von Sicherheits- und Transaktionsmanagementproblemen, auch in einer verteilten Umgebung. Schema-Modifikationen und

Sicherheitsmaßnahmen (Backup, Recovery) können im laufenden Betrieb vorgenommen werden. ORACLE-Datenbanken laufen laut Anwenderangaben äußerst stabil.

- Mächtige Frontend-Tools und Applikationswerkzeuge, integrierte Anwendungsentwicklung von CASE bis Reporting:

  Für alle relevanten Bereiche der Benutzer-Interaktion mit der Datenbank (Browsing, Statistiken etc.) und der Entwicklung von Datenbankapplikationen (Oberflächen, Reports, Graphiken) werden von ORACLE leistungsfähige graphische Werkzeuge angeboten. Die Applikationswerkzeuge unterstützen eine deklarative Anwendungsentwicklung, so daß Standardprobleme einfacher bis mittlerer Komplexität ohne Programmieraufwand realisierbar sind. Ein besonderer Vorzug der Produktpalette von ORACLE ist, daß die Tools auf eine einfache Integration mit anderen ORACLE Werkzeugen, z.T. auch mit Drittanbieter-Produkten, ausgerichtet sind. Auf diese Weise können vollständig integrierte Lösungen auf der Basis von CASE-, Forms-, Reports- und Graphics-Anwendungen entwickelt werden. Darüber hinaus wird sowohl auf der Ebene der Datenbank als auch in allen Bereichen der Applikationsentwicklung (Forms, Reports etc.) die gleiche Programmiersprache benutzt (PL/SQL). Damit verringert sich nicht nur der Einarbeitungsaufwand für die Entwickler, es wird auch der Austausch von PL/SQL-Modulen zwischen den Anwendungen sowie zwischen Client und Server unterstützt. Alle Oracle-Produkte sind auf einer Vielzahl von Plattformen verfügbar und unterstützen den Zugriff auf Fremddatenbanken über ODBC oder Gateways und den Datenaustausch mit anderen Windows-Programmen (Excel etc.).
- Verteilte Datenhaltung, ausgezeichnetes Replikationskonzept
- Weiterentwicklung in Richtung Objektorientierung
- Workflow-Unterstützung: Integration mit Lotus Notes (geplant)
- Internet-Anbindung.

Strategische Vorteile:

- ORACLE ist der Marktführer und bietet daher am meisten Zukunfts- und Investitionssicherheit. ORACLE ist deshalb nicht nur ein Trendsetter auf dem DBMS-Markt (s. TextServer), sondern auch im Bereich SQL-Standardisierung. Hinsichtlich Plattformen-Unabhängigkeit (s. optimale Abstimmung der Release-Stände für UNIX und DBMS) und Portabilität der Anwendungen kommt ORACLE daher eine strategische Bedeutung zu.

- *Ein* Anbieter für DBMS, Textretrieval und Frontend-Entwicklung: Müssen dagegen Werkzeuge unterschiedlicher Hersteller benutzt werden, besteht die Gefahr, daß Probleme von einem auf den anderen geschoben werden.
- ORACLE ist in Deutschland das meist installierte DBMS für die im IZ favorisierte SNI-Hardwareplattform.
- Ausgezeichneter Support und Kundennähe (Niederlassungen in den IZ-Standorten Bonn und Berlin).
- Die mit der Migration beauftragte Firma aStec hat einschlägige (positive) Erfahrungen mit ORACLE-Datenbanken.

3.2 Anmerkungen zur Migration

Das eigentliche Problem bei der Migration stellt das historisch gewachsene nicht-relationale Datenmodell dar, das in großen Teilen heuristisch entstanden (s. Redundanzen) und (naturgemäß) in enger Anlehnung an die Besonderheiten von ADABAS organisiert ist (s. multiple Felder). Die Programme wurden ebenfalls auf diese Besonderheiten hin programmiert und so mit der ADABAS-spezifischen Datenorganisation verdrahtet. Ein derartiges Datenmodell ist für ein relationales DBMS nicht mehr zu gebrauchen.

Der Nutzen eines RDBMS (insbesondere im Sinne einer höheren Produktivität) erschließt sich aber erst bei einem sauberen Relationenmodell[71]. Um ein semantisch sauberes relationales Datenmodell zu erhalten, daß keine spezifischen Eigenschaften des DBMS ausnutzt, muß die ganze Datenbankanwendung daher völlig neu modelliert werden.

Das Standard-Migrationsverfahren von aStec erschlägt das Problem der multiplen Felder dadurch, daß dafür nicht eigene Tabellen gebildet werden, sondern für jede „Primärtabelle“ (z.B. SOLIS) eine sog. „Sekundärtabelle“ anlegt wird, wo jedes multiple Feld ein eigener Tabelleneintrag ist und Mehrfachbelegungen durch Vergabe von NULL-Werten vermieden werden. Damit entsteht eine Struktur, die für eine gut funktionierende relationale Datenbank ebenfalls nicht zu gebrauchen ist, da mit diesem Modell weder die Konsistenz und Integrität der Daten, noch die Transparenz und semantische Interpretierbarkeit des Modells gewährleistet ist. Für Datenkonsistenz müssen bei einem derartigen Modell die Anwender bzw. die Anwendungsprogramme selber sorgen, was dem Sinn relationaler Datenbanken diametral entgegensteht.

[71] Vgl. auch Bauer (1995b), Bauer (1992), Gerkens (1994).

Semantische Beziehungen zwischen Entitäten gehen in diesem Modell verloren, da bestehende Felder für unterschiedliche Aussagen verwendet werden. Darüber hinaus dürfte es bei diesem Modell wegen der zu erwartenden großen Zahl unnötiger NULL-Belegungen zu einer ungeheuerlichen Verschwendung von Speicherplatz und entsprechenden Performance-Verlusten kommen.

Dieser Lösungsansatz wird sich technisch und damit auch ökonomisch als Krücke erweisen, da es sich um eine reine Emulation des alten Modells auf einem RDBMS handelt. Allein der Sprachgebrauch („Primärtabelle“, “Sekundärtabelle“), der dem Konzept relationaler Datenmodellierung widerspricht, deutet darauf hin, daß es mit dieser Lösung Probleme geben wird - insbesondere dann, wenn mit unterschiedlichen Werkzeugen auf die Datenbank zugegriffen werden soll.

Die Migration macht jedoch nur Sinn, wenn auch die Vorteile eines relationalen Datenbanksystems ausgenutzt werden. Für die Entwicklung eines geeigneten relationalen Datenmodells, das Redundanzfreiheit, Datenintegrität und Transparenz des Modells auf der einen Seite und Performance-Gewinne auf der anderen Seite gleichermaßen berücksichtigt, sollte daher genügend Zeit eingeplant werden.

4 Literatur

AMMON, R.v. (1994): ODBC bietet einen neuen Ansatz des Datenbankzugriffs. UIMS sind eine Alternative zu DBMS-spezifischen Werkzeugen. *Computerwoche 3, 35-37.*

BAUER, M. (1992): Relationale Datenbanken - Kaum eingesetzt und beinahe schon veraltet. *Computerwoche 34, 27-30.*

BAUER, M. (1994a): Client/Server bringt DB-Szene in Bewegung. Verteilte und heterogene Umgebung stellt neue Forderungen an DB-Systeme. *FOCUS 1, 4-7.*

BAUER, M. (1994b): RPCs sorgen für problemlose Interprogramm-Kommunikation. *Computerwoche 1994, 23-24.*

BAUER, M. (1995a): Verkapselt und vererbt: Objektorientierte Datenbankkonzepte. *online 3, 58-65.*

BAUER, M. (1995b): Strategien für eine langfristige DBMS-Unabhängigkeit. *online 5, 46-50.*

BORCHERS, D. (1994): Im Markt der DB-Clients wird die Wahl meist zur Qual. *Computerwoche 31, 27-28.*

BRAUCHLE, Th. (1995): Integration heterogener Datenbestände: ODBC et al.. *Datenbanken in Theorie und Praxis, Heft 1, 53-73.*

DEMUTH, B. (1994): Offene Schnittstellen statt proprietärer Gateways. Datenverteilung aus der Sicht des Anwendungsprogrammierers. *Computerwoche 50, 50-52.*

DRUCKS, H.J. (1992): Die Zukunft liegt jenseits des Coddschen Datenmodells. *Computerwoche 34, 31-34.*

GERKENS, R. (1994): Migrationskonzepte von IBM, CA und System Performance. Wie sich der Übergang in die relationale Welt gestalten läßt. *Computerwoche 31, 29-31.*

HAARMANN, G. (1995): Vom Datenkunden zum Informationskönig. Durch Data-/Information-Warehousing sollen aus Daten entscheidungsrelevante Informationen werden. *FOCUS 2, 24-27.*

HERZOG, U. und LANG, S.M. (1995): Eine Technologie im Wandel. Bestandsaufnahme und Trends im Bereich Datenbanken. *FOCUS 2, 4-6.*

JENZ & PARTNER (1994): Relationale Datenbanksysteme: Ein Produktvergleich (Auszug).

JENZ & PARTNER (1995): Strategie-Studie Datenbanken in Netzwerken (Auszug).

KOCH, Jürgen (1995): Datenbanken und der Umgang mit Altsystemen. Der Weg in eine neue DBMS-Welt ist dornenreich. *FOCUS 2, 12-15.*

KUPPINGER, M. (1995): Keine Lösung für alle Fälle. Vor- und Nachteile von ODBC als betrieblicher Standard. *FOCUS 2, 16-18.*

NUßDORFER, R. (1994): Interoperabiblität gewährleistet Unabhängigkeit. Offene Schnittstellen statt offene Systeme gefordert. *FOCUS 1, 11-13.*

RITTER, U. (1995): Die traditionelle Aufgabe wird sich grundlegend ändern. Künftige Datenbanken erfordern die Kombination von strukturierten und unstrukturierten Daten. *FOCUS 2, 32-33.*

SAUER, H. (1992): Relationale Datenbanken. Theorie und Praxis. *Addison-Wesley. Bonn-München-Paris.*

SCHLAGETER, G. und STUCKY, W. (1987): Datenbanksysteme: Konzepte und Modelle. *Teubner Studienbücher.*

SCHMATZ, K.-D. und WEIKERT, P. (1995): Die Anforderungen der Anwender berücksichtigen. Auswahl und Bewertung von objektorientierten Datenbanksystemen. *FOCUS 2, 19-21.*

STÜRMER, G. (1993): Oracle7. Die semantisch verteilte Datenbank. *dbms publishing.*

ULLMANN, J.D. (1988): Principles of Databases and Knowledge-Base Systems. Volume I. *Computer Science Press.*

VORWERK, R. (1995): In der Praxis gibt es auch Grautöne. Relationale Datenbanken eignen sich durchaus zur Speicherung von Objekten. *FOCUS 2, 22-23.*