Überlegungen zur Architektur von Datenbanksystemen für Ingenieuranwendungen

Bernhard Mitschang
Universität Kaiserslautern, Fachbereich Informatik
Erwin-Schrödinger-Straße
D-6750 Kaiserslautern

Überblick

Die stetige Erweiterung und Vergrößerung der rechnergestützten ingenieurwissenschaftlichen Anwendungen verursacht u.a. ein schnelles Wachstum der zu verwaltenden Datenbestände. Einhergehend mit dieser Expansion und Integration der Anwendungen steigt die Bedeutung und Notwendigkeit eines Datenbanksystem-Einsatzes.

In der vorliegenden Arbeit werden wichtige Anforderungen an Datenbanksysteme beim Einsatz in Spezialanwendungsbereichen wie Prozeßdatenverarbeitung, geographische Datenhaltung, Bild- und Textverarbeitung sowie CAD/CAM/CAE vorgestellt. Diese Anforderungen beeinflussen die Schnittstelle zwischen Anwendungssubsystem und Datenbanksystem sowie die Architektur des integrierten Gesamtsystems.

Hier wird ein Architekturkonzept für Datenbanksysteme erarbeitet, die speziell für den Einsatz in solchen Non-Standard-Anwendungsgebieten geeignet sind. Aufbauend auf einem allgemeinen Datenbanksystemkern wird eine Modellabbildungsschicht realisiert, die von einer vorgeschalteten Anwendungsanalyse beeinflußt ist. Der Umfang und die Auswirkungen der Anwendungsanalyse auf die zugrundegelegte zweigeteilte Datenbanksystem-Architektur werden angegeben sowie die Aufgaben der Modellabbildungsschicht und die Funktionen des Datenbanksystemkerns erörtert. Zusätzlich wird eine Abgrenzung und Bewertung der hier vorgeschlagenen Datenbanksystem-Architektur gegenüber jener bestehender Systeme und gegenüber denkbaren alternativen Architekturen angegeben.

1. Einleitung und Motivation

Die ersten Anwendungssysteme in den Ingenieurwissenschaften (schon aus den sechziger Jahren) waren durch die Verwendung eines einfachen Dateikonzeptes gekennzeichnet. Das Gesamtsystem bestand aus einzelnen voneinander abgegrenzten Teilsystemen, die jeweils ihre eigene Datenhaltung mit Hilfe von separaten Dateien abwickelten. In /Eb83/ wird als Fallbeispiel ein System zur Unterstützung der Konstruktion von Leiterplatten angegeben. Bei dem Entwurf solcher Systeme waren eine

- anwendungsfreundliche Benutzerschnittstelle (etwa Graphikterminalein- und -ausgabe),
- effiziente Algorithmen (z.B. Hidden-Line-Algorithmen oder Layout-Optimierungen) und eine
- angepaßte Datendarstellung (wie Begrenzungsflächendarstellung oder Zellzerlegung als Darstellungsschemata für geometrische Objekte) von übergeordneter Wichtigkeit.

Auf die gemeinsame Datenhaltung hingegen wurde kein besonderer Wert gelegt. Ein hoher Grad an Redundanz in der gespeicherten Information, schlechte dateiübergreifende Zugriffsmöglichkeiten sowie ein geringes Maß an Datenunabhängigkeit waren die wichtigsten Nachteile (siehe /Eb83/ und /Mi84/).

Mit der Erweiterung und Vergrößerung der Anwendungen wuchsen auch die zu verwaltenden Datenbestände und die Anzahl der notwendigen Verknüpfungen zwischen den einzelnen Dateien. Diese Entwicklung brachte zum einen verschärfte Modellierungs- und Integritätsprobleme und zum anderen auch erhöhte Sicherheits- und Konsistenzprobleme mit sich. Eine gemeinsame Benutzung von Daten wurde immer schwieriger und aufwendiger. Man war mit den so "gewachsenen", komplexen Systemen unzufrieden. In den seit den siebziger Jahren konzipierten Systemen der zweiten Generation sollten die erhöhten Anforderungen durch den Einsatz konventioneller Datenbanksysteme befriedigt werden. Man setzte die einzelnen Anwendungsmoduln einfach auf das zugrundeliegende Datenbanksystem (DBS) auf.

Der Nutzen dieser Methode lag in der deutlich verringerten Systementwicklungszeit und in der Übernahme der Vorteile, die konventionelle Datenbanksysteme anbieten, seien es nun hierarchische, netzwerkartige oder relationale (siehe /Da81/ und /Mi84/). Auf der anderen Seite hatte diese Methode jedoch auch eine ganze Reihe von schwerwiegenden Nachteilen. Die Daten, die man zuvor getrennt in privaten Dateien abgelegt hatte, mußten zusammengefaßt und mit Hilfe des zur Verfügung stehenden Datenmodells dargestellt werden. Anhand der zwei Beispiele, die wegen der Übersichtlichkeit im nachfolgenden Kapitel zusammengefaßt sind, soll ein Eindruck vom Umfang des Modellierungsaufwands vermittelt werden. Die dort aufgeführten Problempunkte sind keineswegs vollständig und erschöpfend, sollen jedoch dazu genügen, die Komplexität der Anfragebearbeitung und den Aufwand der Modellierung in den Non-Standard-Anwendungsgebieten aufzuzeigen. Außerdem erzwang die lose Bindung der Programme an die Daten einen erhöhten Aufwand beim Datenzugriff, was sich in einem schlechten Leistungsverhalten des Gesamtsystems äußerte.

In /AF83/ wurden Anforderungen an Datenhaltungssysteme in der Prozeßdatenverarbeitung (PDV) analysiert. Dort wird ausgesagt, daß man heutzutage für die Datenhaltung in der PDV die datenabhängigen, leistungsstarken Dateiverwaltungssssteme den datenunabhängigen, langsameren DBS vorzieht. Der Leistungsaspekt wird also prioritätsmäßig höher eingestuft als die Datenunabhängigkeit. Ein expliziter Leistungsvergleich zwischen einem System der ersten Generation mit einem der zweiten Generation, allerdings aus dem Anwendungsgebiet Schaltkreisentwurf, wird in /GS82/ beschrieben. Dort wurde ein CAD-Paket, basierend auf dem Dateikonzept, mit einer äquivalenten, auf dem relationalen Datenbanksystem INGRES /SW76/ installierten Anwendung in bezug auf das Leistungsverhalten verglichen. Das CAD-Paket zeigte dabei ein deutlich besseres Verhalten (mindestens Faktor 3). Vergleichstests, die sich auf besondere Zugriffsanforderungen bezogen, wie z.B. auf die räumliche Suche (siehe Kapitel 2), offenbarten sogar Leistungsunterschiede vom Faktor 20. Aufgrund vereinfachter Aufwandsabschätzungen wird in /Eb83/ ein Faktor von 50 ermittelt. Diese extreme Leistungsdifferenz basiert hier noch zusätzlich darauf, daß komplexe und anwendungsnahe Operationen wie die graphische Darstellung eines durch ein CSG-Modell (oder Volumenmodell, siehe /Re80/) dargestellten Körpers

oder auch Flächenschnittverfahren zur Durchführung von Mengenoperationen auf Körpern, die im Begrenzungsflächenmodell dargestellt sind, betrachtet werden. Ein quantitativ gleichwertiges Ergebnis wird in /Fi83/ bei der Betrachtung der Kollisionsprüfung von Außenkonturen zweier Blechteile ermittelt. In /Eb83/ wird zusätzlich noch folgendes einfache, aber sehr aussagekräftige Beispiel aufgeführt: Das Lesen von Linien aus einem relationalen DBS mit anschließender Ausgabe mittels eines geräteunabhängigen Graphiksystems erreichte günstigstenfalls einen Durchsatz von 15 Linien pro Sekunde. Bei vergleichbarer Programmierung mit Hilfe konventioneller Dateibearbeitung und Verwendung einer niedrigen Schnittstelle zum Graphiksystem ließen sich etwa 100 Linien/Sekunde ausgeben.

Aus diesem Grunde müssen die Systeme der dritten Generation, die es jetzt zu entwerfen und zu konstruieren gilt, die Vorteile der Vorgängergeneration beibehalten und zudem noch ein akzeptables Leistungsverhalten aufzeigen. Dies bedeutet für die Datenverwaltungskomponente, daß die Schnittstelle zu den Anwendungsmoduln, die verfügbaren Repräsentationsmöglichkeiten des Datenbanksystems und die Systemarchitektur selbst zu überprüfen bzw. neu zu konzipieren sind. Aufgrund der zuvor geführten Diskussion und einschließlich der Ergebnisse aus Kapitel 2 scheint bisher ohne Zweifel festzustehen, daß die Modellierungsmöglichkeiten an die betreffende Anwendung angepaßt und die Zugriffsanforderungen durch neuartige und verbesserte Speicherungsstrukturen unterstützt werden müssen.

Die Vergleichsmessungen in /Fi83/ scheinen diese Forderungen zu bestätigen. Dort wurde die dateiorientierte Datenhaltung eines Geometrischen Modellierungssystems durch ein bzgl. Zugriffsmöglichkeiten und Speicherungsstrukturen zugeschnittenes netzwerkartiges DBMS ausgetauscht und annähernd das gleiche Leistungsverhalten erzielt.

Zur Konstruktion von solchen spezialisierten, d.h. auf die betreffende Anwendungsklasse zugeschnittenen, Non-Standard-Datenbanksystemen (NDBS) gibt es mehrere Möglichkeiten, wie in Kapitel 3 aufgezeigt wird. Nach einer kurzen Bewertung der verschiedenen Architekturvorschläge wird im darauffolgenden Kapitel der am besten erscheinende Ansatz aufgegriffen und detaillierter diskutiert.

Zuvor wird jedoch in Kapitel 2 auf die schon angesprochene Modellierungsproblematik sowie auf damit direkt zusammenhängenden Problempunkte wie Anfragekomplexität, Konsistenzerhaltung und Leistungsverhalten eingegangen.

2. Beispiele zur Problemerkennung bei unangepaßter Modellierung

Beispiel 1: geographische Datenhaltung (/HR83/, /Fr83/, /GP83/)

Das erste Beispiel zeigt die Sicht des Benutzers eines Landinformationssystems auf seine Daten; hier speziell auf den Objekttyp PARZELLE. Die für den Benutzer relevanten Eigenschaften einer Parzelle, die er modelliert wissen möchte, sind z.B.:

- Eine Parzelle wird in speziellen Registern (Liegenschafts- und Grundbuch) geführt, hat einen Besitzer, Kaufpreis, Fläche, Nutzungswert etc.

- Eine Parzelle besitzt eine identifizierende Nummer.
- Eine Parzelle hat eine bestimmte (Orts-)Lage.
- Parzellen liegen "dicht", d.h., zwischen einer Parzelle und ihren Nachbarn darf es keinen freien Raum geben.
- Parzellen dürfen einander nicht überlagern.
- Eine Parzelle ist ein geschlossenes Polygon mit einer beliebigen Anzahl von Punkten und Kanten.

Ebenfalls von Wichtigkeit für den Benutzer sind die verschiedenen Zugriffsmöglichkeiten auf diesen Objekttyp:

- objektbezogen: Zugriff über die identifizierende Nummer und/oder Zusatzbedingungen bzgl. anderer Attribute
- raumbezogen: Zugriff über eine Gebietsangabe
- nachbarschaftsbezogen: Zugriff mittels topologischer Beziehungen, wie z.B. über eine Nachbarschaftsbeziehung oder Begrenzungsbeziehung.

Versucht man z.B., obigen Objekttyp PARZELLE im Relationenmodell darzustellen, so existieren gleich mehrere Möglichkeiten. Die wünschenswerteste Darstellung, die für eine Parzelle genau ein Tupel einer Relation benutzt, scheitert an der Normalisierungsforderung des Relationenmodells, da eine variabel lange Liste (Wiederholungsgruppe) für die begrenzenden Kanten und zugehörigen Punkte nicht verwendbar ist. Man kann sich allerdings leicht 4 unterschiedliche, im Relationenmodell erlaubte Darstellungen überlegen: Etwa eine Zerteilung in die 3 Relationen PARZELLE, KANTE und PUNKTE oder auch eine Darstellung mittels einer Relation PARZELLE, die pro Tupelausprägung nur genau einen Eckpunkt der Parzelle repräsentiert; eine ganze Parzelle wäre damit in eine Menge von Tupelausprägungen zerlegt. Die restlichen Darstellungsmöglichkeiten unterscheiden sich nur in der Anzahl der verwendeten Relationen sowie in der Anzahl der benötigten Tupelausprägungen zur Beschreibung einer Parzelle. Zusätzliche Schwierigkeiten ergibt die Einbeziehung und Realisierung der restlichen o.g. Aspekte wie z.B. die Dichte-Eigenschaft oder die Zugriffspfade.

Schaut man sich obige Modellierungsergebnisse an, so erkennt man, daß Relationen definiert werden, die anstatt eine Parzelle zu beschreiben, nur Teilaspekte derselben darstellen. Das ursprüngliche Entity "Parzelle" ist verschwunden. Das heißt, der Anwendungsprogrammierer kann nicht mehr in seiner gewohnten Benutzersicht (s.o.) arbeiten. Er muß jetzt alle seine Operationen der von der betreffenden Modellierung bestimmten Objektbeschreibung anpassen. Analog zu dieser Operationsmodifikation muß auch eine Anpassung der auszuführenden Integritätskontrollen erfolgen.

In /SP82/ wird die gleiche Problematik, allerdings bezogen auf das Anwendungsgbiet Textverarbeitung/Information-Retrieval aufgezeigt und anhand eines Beispiels belegt, welches sehr deutlich die Komplexität der Anfrageformulierung und -Abarbeitung bei unangemessener Modellierung hervorhebt.

Beispiel 2: rechnergestütztes Entwerfen und Konstruieren (/NH82/, /EW81/, /Lo81/)

Der VLSI-Chip-Entwerfer sieht einen elektronischen Schaltkreis in verschiedenen Repräsentationen (Mehrfachrepräsentation), z.B. in einer funktionalen Spezi-

fikation, als Schaltkreisdiagramm, als Layout oder auch in einer speziellen Hardware-Beschreibungssprache. Unterschiedliche Technologien, andersstrukturierte Layouts etc. zu verwenden, eröffnen ihm zusätzlich die Möglichkeit, verschiedene Alternativen nebeneinander auszuprobieren und auch Versionen eines Schaltkreises zu entwerfen. Die Modellierungsprobleme liegen hier besonders in der Darstellung der Abhängigkeiten unter den verschiedenen Mehrfachrepräsentationen, Alternativen und Versionen sowie in der Aufrechterhaltung der Konsistenz. Hierzu wird in /Ne83/ ein graphenbasiertes Konzept zur Repräsentation und Verwaltung von Entwurfsinformation in einem herkömmlichen DBS angegeben.

3. Architekturvorschläge für Non-Standard-Datenbanksysteme

In Abbildung 1 sind, wie z.B. in /SL83/ kurz vorgestellt, aber nicht in der hier durchgeführten Exaktheit bewertet, die einzelnen Architekturvorschläge für Non-Standard-DBS abgebildet. Die Reihenfolge der angegebenen Vorschläge spiegelt dabei die Fortentwicklung der Systemstrukturen wider. Damit einhergehend findet auch eine ständige Verbesserung der Eignung dieser Architekturen für den Einsatz in Spezialanwendungsgebieten statt.

Die vorausgegangene Diskussion legt nun die folgende einfache Architektur eines NDBS nahe: Auf das konventionelle DBS wird eine zusätzliche Abbildungsschicht gesetzt (Abbildung 1a). Diese Zusatzebene unterstützt alle von der Anwendungsseite als relevant erachteten Eigenschaften und Strukturen und bietet dem Benutzer eine einheitliche Schnittstelle zur Verwaltung aller Informationen; d.h. falls erwünscht von konventionellen, satzorientierten Daten über Text- und Bilddaten bis zu Daten im Zusammenhang mit Entwurfs- und Konstruktionsprozessen. Diese Architektur stimmt mit der von Systemen der zweiten Generation aus Kapitel 1 fast überein. Der einzige Unterschied liegt darin, daß hier die Modellierungsmöglichkeiten der Anwendungsprogramme (AP abgekürzt) durch eine mächtigere und angepaßte DBS-Schnittstelle verbessert werden. Eine direkte Leistungssteigerung des Gesamtsystems hat dies allerdings nicht zur Folge. Das Modellierungsproblem, inklusive aller Konsequenzen, wie in Kapitel 2 ausführlich geschildert, besteht weiterhin, nun jedoch in der Zusatzebene des DBS: Das dem Benutzer zur Verfügung gestellte angepaßte Datenmodell muß auf ein vom konventionellen Datenbanksystem bereitgestelltes Datenmodell (mit allen Beschränkungen) abgebildet werden. Es fand also nur eine Problemverlagerung vom Anwendungsprogramm in die Zusatzebene des DBS statt. Daraus folgt, daß sowohl die Vorteile als auch die überwiegenden Nachteile aus Kapitel 2 hier ebenfalls zutreffen und diesen Architekturansatz in Frage stellen. In /Eb83/ und /Lü83/ werden Systeme vorgeschlagen, die die Zusatzebenen-Architektur beinhalten. Die Diskussion in /SP82/ über Integrationsmöglichkeiten zwischen einem Information-Retrieval-System und einem DBMS führt von einem Text-Preprocessor als Zusatzebenen-Architektur über die nachfolgende Kombinations-Architektur zu einem integrierten System gemäß Abb. 1c oder Abb. 1d.

Der nächste Vorschlag (Abbildung 1b) kombiniert ein konventionelles Datenbanksystem zur Verwaltung satzorientierter Daten mit voneinander getrennten Spezialsystemen, die beispielsweise wie Information-Retrieval-Systeme oder Geome-

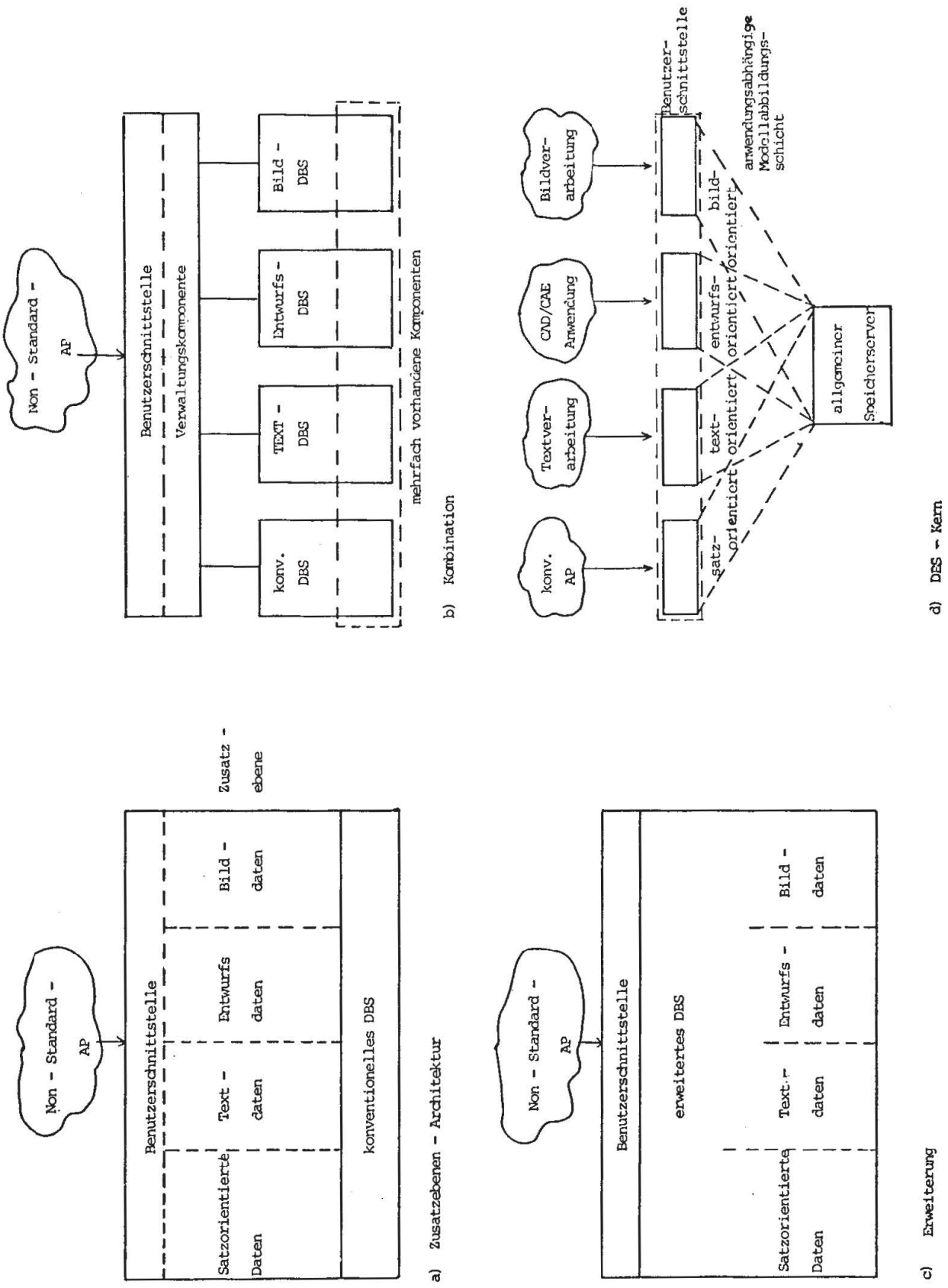

Abbildung 1: Architekturvorschläge für Non-Standard-DBS

trische Modellierungssysteme teilweise schon vorhanden sind. Dadurch werden von vornherein Entwicklungs- und Implementierungskosten eingespart. Das Gesamtsystem besitzt für jede Anwendungsklasse eine eigens dafür vorgesehene Datenhaltung. Das bedeutet, daß gleiche Funktionen wie Speicherverwaltung, Datenverwaltung, Änderungsdienst, Retrieval und Vorkehrungen für die Datensicherheit in allen Subsystemen wiederholt vorhanden sind. Der Austausch von Daten und die Kommunikation der verschiedenen Subsysteme ist umständlich und schwierig, da dies nur über die gemeinsame Benutzerschnittstelle möglich ist. Die Verwaltungskomponente innerhalb der gemeinsamen Schnittstelle wird sehr komplex, da sie die Konsistenz der verteilten Daten zu gewährleisten sowie die Abbildung auf unter Umständen unterschiedliche Subsystemschnittstellen durchzuführen hat. Insgesamt bewertet, erscheint diese Architektur als unnötig komplex und für jeweils nur eine Spezialanwendung zu allgemein und damit ineffizient.

Eine Fortentwicklung der vorigen Architektur ist in Abbildung 1c zu sehen. Hier wird eine Erweiterung eines konventionellen Datenbanksystems um integrierte Verwaltungskomponenten für Bild-, Text- und Entwurfsdaten vorgeschlagen. Die anwendungsspezifischen Komponenten können sich über alle internen Architekturebenen eines herkömmlichen DBS (siehe /Hä78/) erstrecken, unter Umständen sogar eine unterschiedliche Anzahl von Abbildungsschritten oder, wie oben, schon in anderen Komponenten realisierte Funktionen enthalten. Erschwerend kommt die Komplexität und Größe des Gesamtsystems noch hinzu. Außerdem kann man nicht a priori Verwaltungskomponenten für alle denkbaren Anwendungsgebiete bereitstellen. Einflüsse der Systemarchitektur auf Entwicklungs- und Implementierungskosten solcher Non-Standard-Systeme sollten ebenfalls mit in die Bewertung eingehen, d.h. für diesen Architekturansatz als Negativposten. In /Lo83/ sowie in /Fi83/ und /Lo81/ werden Erweiterungen von herkömmlichen DBS um Non-Standard-Aspekte wie z.B. die Definition von Komplexobjekten, objektbezogene Clusterbildung oder auch die Verwaltung von unformatierten Datenelementen, angegeben.

Nicht zuletzt aufgrund obiger Bewertungspunkte erscheint der in Abbildung 1d gezeigte Architekturvorschlag für den Einsatz als NDBS in einem Ingenieurssystem am geeignetsten. Das Bereitstellen eines allgemeinen NDBS für alle Anwendungsklassen ist wegen der Trägheit, Allgemeingültigkeit und Komplexität dieser Systeme unangebracht; dies war ja auch die Begründung der Inadäquatheit der Systeme der zweiten Generation, die konventionelle Datenbanksysteme benutzten. Man muß also versuchen, ein zugeschnittenes DBS für die Erfordernisse der Objekte und Operatoren einer Anwendungsklasse zur Verfügung zu stellen. Dabei ist es sinnvoll, bevor man für jedes Gebiet ein separates System entwickelt, zu untersuchen, inwieweit man welche Schichten in allen anwendungsbezogenen NDBS übernehmen kann, d.h., man sollte zwischen anwendungsabhängigen und -unabhängigen Systemschichten unterscheiden.

Aufbauend auf einem anwendungsunabhängigen DBS-Kern, im folgenden allgemeiner Speicherserver (SS) genannt, wird eine durch die spezielle Anwendung bestimmte Modellabbildungsschicht realisiert. Sie bietet dem Benutzer ein anwendungsorientiertes Modellierungswerkzeug, z.B. ein erweitertes Datenmodell, semantisches Datenmodell oder Objektmodell an und setzt die so definierte Benutzersicht auf die SS-Schnittstelle um. Diese Modellabbildung beinhaltet ein Optimierungspotential,

welches es auszuschöpfen gilt. Die SS-Schnittstelle bietet nun die Möglichkeit, bestimmte und für die meisten Non-Standard-Anwendungen wichtige Eigenschaften zu unterstützen, was allen unterschiedlichen Abbildungsschichten gleichsam zugute kommt. In /PS84/, /Ka83/ und in /Lu83/ werden Projekte vorgestellt, innerhalb denen NDBS nach diesem Architekturansatz enwickelt und implementiert werden.

Genaueres über die Aufgaben der Modellabbildungsschicht und über die Funktionen des DBS-Kerns werden im nachfolgenden Kapitel erläutert.

4. Die Datenbanksystemkern-Architektur für Ingenieuranwendungen

Die Bewertung der im vorigen Kapitel aufgeführten Architekturvorschläge für NDBS fiel deutlich zugunsten der DBS-Kern-Architektur aus. Jedoch ließ die bisherige Beschreibung dieses Architekturansatzes noch vieles im Unklaren. So sind z.B. die Aufgaben und Schnittstellen der beiden Ebenen noch sehr wenig spezifiziert und auch die geforderte Anpassung an die speziellen Anforderungen des jeweiligen Anwendungsgebietes ist ebenfalls noch unklar. Deshalb wird in diesem Kapitel ein Entwurfskonzept für die Entwicklung von NDBS für den Einsatz in Ingenieuranwendungen vorgestellt.

Wie oben schon erwähnt, bildet der DBS-Kern-Architekturansatz den Rahmen des Gesamtkonzepts. Der eigentliche Kern der Entwurfsmethode liegt darin, eine möglichst optimale Anpassung des NDBS an die Anforderungen der betreffenden Anwendungsgebiete unter Ausnutzung der noch verbleibenden Freiheitsgrade wie Schnittstellen und Ebenenfunktionen zu erreichen. Hierzu dient die Anwendungsanalyse mit ihrem Untersuchungsschema. Das Ziel dieser Analyse ist nun, genügend Information über die Anwendungsgebiete zu bekommen, um dann die erforderliche angepaßte Benutzerschnittstelle, die möglichst optimale Modellabbildung und den unterstützenden DBS-Kern konzipieren zu können.

In Abschnitt 4.1 werden drei anschauliche Beispiele angegeben, die die Struktur und den Umfang der Anwendungsanalyse aufzeigen. Zusätzlich werden dort wichtige Charakteristiken der untersuchten Anwendungsgebiete vorgestellt. Diese Information erlaubt dann in Abschnitt 4.2 eine detaillierte Darstellung des Gesamtkonzeptes.

4.1 Beispiele zur Anwendungsanalyse

Die Anwendungsanalyse benutzt ein Untersuchungsschema, das nach der folgenden Vorgehensweise arbeitet: Zuerst werden die Objekt(typen) inklusive ihrer Beschreibungen und Eigenschaften bestimmt. Dann werden die Beziehungen zwischen den Objekttypen ermittelt sowie deren Funktionalität und Abhängigkeit. Die Funktionalität beschreibt den Typ der Beziehung, d.h. eindeutig, funktional oder komplex. Die Abhängigkeit liefert eine Aussage darüber, ob alle oder nur einige der Objekte eines Typs an der Beziehung teilnehmen. Parallel zu diesen Untersuchungen werden für alle Strukturen Mengengerüste in Form von Größe, Anzahl und Verteilung aufgestellt. Zusätzlich zu den zu analysierenden Operationen werden die notwendigen

Leistungsanforderungen und Integritätsbedingungen erarbeitet.

In den nachfolgenden drei Beispielen wird die Sicht des Benutzers auf seine Daten beschrieben. Der Detaillierungsgrad wird auf das notwendige Maß reduziert. Die für die späteren Abschnitte wichtigen Aspekte sind jeweils unterstrichen. Beispiel 1 zeigt die wesentlichen Analyseschritte auf, wohingegen in den restlichen beiden Beispielen nur noch wichtige Zusatzaspekte aus den jeweiligen Anwendungsgebieten angegeben werden.

Beispiel 1: (/HR83/, /Fr83/, /GP83/)

Das hier betrachtete Beispiel ist aus dem Anwendungsgebiet Raumplanung - speziell Stadtplanung - entnommen.

Die Objekttypen, mit denen es der Stadtplaner generell zu tun hat, sind z.B.
- Straßen, Straßenabschnitte, Kreuzungen, Liniensegemente,...
- Gebäude, Parzellen, (Wohn-)Blöcke, Stadtbezirke,...

die entweder punkt-, linien- oder flächenförmig sind.

Diese geometrischen Eigenschaften wirken sich auf die Beziehungen zwischen einzelnen Objekten aus. Es können dabei fünf verschiedene Beziehungstypen unterschieden werden:
- Nachbarschaftsbeziehung: Parzellen sind einander benachbart
- Enthaltenseinsbeziehung: Gebäude liegen in Parzellen
- Hierarchiebeziehung: ein Bezirk setzt sich zusammen aus Blöcken, diese wiederum aus Parzellen, etc. Diese Beziehung liefert die sog. "komplexe" Objektstruktur
- Begrenzungsbeziehung: ein Block ist durch Straßenabschnitte begrenzt
- Art/Gattungsbeziehung: eine Linienüberschneidung ist entweder eine Kreuzung oder ein Parzelleneckpunkt oder eine Geländeecke.

Typische vom Stadtplaner durchzuführende Operationen (siehe Beispiel 1 in Kap. 2) sind z.B.:
- Retrieval von spezifizierten Objekten; räumliche Anfragen, wie: "Bestimme alle Parzellen, die vollständig innerhalb von dem Planquadrat mit der Nummer 999 liegen!" oder auch topologische Anfragen, wie: "Bestimme alle direkten Nachbarn zur Parzelle 777".
- Modifikationsoperationen, wie Speichern/Löschen und Teilen/Vereinigen einer Parzelle oder auch höhere Operationen wie "Blockteilung entlang einer vorgegebenen Trennlinie".

An Integritätsbedingungen werden u.a. die folgenden verlangt:
- Eindeutigkeit der Geometrie, z.B. dürfen sich Straßen nur an Kreuzungen schneiden.
- rechtliche Forderungen, wie z.B. die Bestimmung, daß der Abstand eines Gebäudes von der Parzellengrenze einen gesetzlich vorgeschriebenen Mindestwert nicht unterschreiten darf.

- geographische Gegebenheiten, wie die Gebietsaufteilung, welche eine vollständige und nicht überlappende Gebietsüberdeckung z.B. mittels Parzellen voraussetzt.

Information über bestimmte Sicherheitsbedingungen an die Daten ist ebenfalls verfügbar. Ein Beispiel dafür ist die Bedingung, daß abgespeicherte rechtliche Forderungen und gesetzliche Bestimmungen nicht änderbar sind und deshalb einem besonderen Schutz unterliegen.

Der Stadtplaner hat aber auch bestimmte Vorstellungen von dem Leistungsverhalten des Systems. Er weiß ebenfalls im voraus, welche Operationen mit welchen Aufrufhäufigkeiten und unter welchen Zeitbedingungen auszuführen sind. Die Reihenfolge einzelner Operationen ist häufig auch vorauszusagen. Hier treten der objektbezogene und der raumbezogene Zugriff am häufigsten auf. Durch Zusatzangaben über Objektgröße, Anzahl und Verteilung kann die Leistungscharakteristik vervollständigt werden. Angaben dieses Typs sind z.B. die folgenden:

- Ein Wohnblock setzt sich aus einer Maximalzahl von Parzellen zusammen, von denen jede eine maximale Fläche nicht überschreiten darf.
- Die Dichteverteilung einer Parzelle ändert sich von Stadt- zu Landregionen extrem.

Beispiel 2: (/NH82/, /EW81/, /Lo81/)

Im zweiten Beispiel aus Kapitel 2 wurde der VLSI-Entwurf als Anwendung aus dem Gebiet des rechnergestützten Entwerfens und Konstruierens gewählt. Die dort aufgezeigten Probleme lagen in der Darstellung der Abhängigkeiten unter den verschiedenen Mehrfachrepräsentationen, Alternativen und Versionen sowie in der Aufrechterhaltung der Konsistenz. Gemäß den physikalischen, geometrischen, technologischen und strukturellen Eigenschaften der technischen Objekte kann man die Datenstrukturen dieser Anwendungsklasse in einzelne Teilbereiche aufspalten. So umfaßt der physikalische Bereich etwa die Materialeigenschaften, elektrische Eigenschaften usw., während im geometrischen Bereich die Objektgeometrie und Zeichnungsinformation verwaltet wird. Der technologische Teil enthält besondere Techniken, Arbeitspläne, Werkzeuge etc. Im strukturellen Teilbereich wird die Objektstruktur festgehalten, wie z.B. das Wissen über den Aufbau eines VLSI-Chips: Ein Chip besteht aus einzelnen Funktionseinheiten, die ihrerseits aus Zellen zusammengesetzt sind, welche wiederum aus Gattern und Transistoren bestehen.

Das typische Zugriffsgranulat ist hier das technische Objekt, welches gerade modifiziert, weiterentwickelt oder neuentwickelt wird. Die benötigten Zugriffsoperationen sind die gleichen wie in Beispiel 1. Der objektbezogene Zugriff liegt in der Aufrufhäufigkeit jedoch vor allen anderen Operationen. Der Zugriff auf das interessierende komplexe CAD-Objekt wird hier, allerdings im Gegensatz zur objektbezogenen Zugriffsmethode in der geographischen Datenhaltung, häufig ausgelöst durch eine Zeigeoperation (Pick-Funktion, siehe /Fi83/) am Bildschirm. Der damit vom Benutzer am Anfang einer Verarbeitungseinheit festgelegte Entwurfskontext bleibt über die gesamte Verarbeitungszeit relativ konstant (MByte-Bereich) und umfaßt in gängiger Weise nur einige technische Objekte, inklusive deren Beschreibungen. Auf Grund der Natur eines Entwurfsprozesses, der viele

eventuelle Lösungen erwägt und erprobt, die oft unterschiedliche Technologien mit einer jeweils verschiedenen Menge von beschreibenden Attributen einbeziehen, sind hier dynamische Schemata notwendig. D.h., der Benutzer kann parallel zu seinem Entwurfsprozeß neue Objekte und/oder zugehörige Beschreibungen definieren bzw. alte löschen oder ändern. Durch den Entwurfsprozeß kommen Verarbeitungszeiten, d.h. Transaktionsdauern von Tagen oder Wochen zustande. Ein Zurücksetzen der Verarbeitung auf den Stand von vor 4 Wochen aufgrund eines Systemfehlers ist für den Entwerfer nicht mehr tolerierbar. Das bedeutet, daß angepaßte Recovery- und Sperrkonzepte innerhalb des DBS zu entwickeln sind.

Beispiel 3: (/Ch81/, /ID81/, /Ta81/)

Die Bildverarbeitung und Mustererkennung befaßt sich mit der Verwaltung der Bilddaten und zugehörigen Beschreibungsinformationen sowie mit dem Erkennen und Verwalten von Bildinhalten. Zur Darstellung von Bildern muß außer einem evtl. vom System generierten Identifikator noch ein spezieller Datenelementtyp mit variabel langem nicht vordefiniertem Format zur Verfügung stehen. Die Operationen auf Bildern, wie Skalierung, Überlagerung, Ausschnittsbildung etc., sind auf den speziellen Datentyp zu übersetzen. Hier zeigt sich deutlich, daß eine hochparallele Operationsausführung auf den Daten eines Bildes möglich ist und auch zur Leistungssteigerung ausgenutzt werden kann.

Die von der Bildanalyse erkannten und korrekt interpretierten Objekte müssen gespeichert werden, was u.U., analog Beispiel 2, ebenfalls dynamische Schemata erfordert. Eine Zugriffsmöglichkeit zu den Bildern, die die zu suchenden Objekte beinhalten, muß zusätzlich aufgebaut werden - inhaltsbezogener Zugriff. Die zuvor schon genannten Zugriffsarten, wie raum- und nachbarschaftsbezogen sowie objektbezogen sind bei dieser Anwendungsklasse auch anzutreffen. Der objektbezogene Zugriff bezieht sich zum einen auf Bilder und zum anderen auf erkannte Objekte und wird am häufigsten angewandt. Die auszuführenden Transaktionen sind von kurzer Dauer und lokal, d.h., während einer Verarbeitungseinheit wird jeweils nur ein physisches Bild mit den darauf erkannten Objekten angesprochen.

4.2 Einfluß und Auswirkungen der Anwendungsanalyse auf die Systemschnittstellen

Abbildung 2 zeigt eine Verfeinerung von Abbildung 1d mit ersten eingearbeiteten Ergebnissen der Anwendungsanalyse. Die einzelnen Ebenen sind um die von ihnen zur Durchführung ihrer Aufgaben benötigten Strukturen und Funktionen erweitert.

Wie schon aus dem vorherigen Abschnitt ersichtlich ist, umfaßt die Anwendungsanalyse die Untersuchung von Objekt(typ)en, speziellen Beziehungen und (Zugriffs-)Operationen, Integritäts- und Sicherheitsbedingungen und auch von Leistungscharakteristiken.

Die erhobenen anwendungsspezifischen Informationen repräsentieren nun die Anwendung. Sie sind mit Hilfe der von der Modellabbildung zur Verfügung gestellten

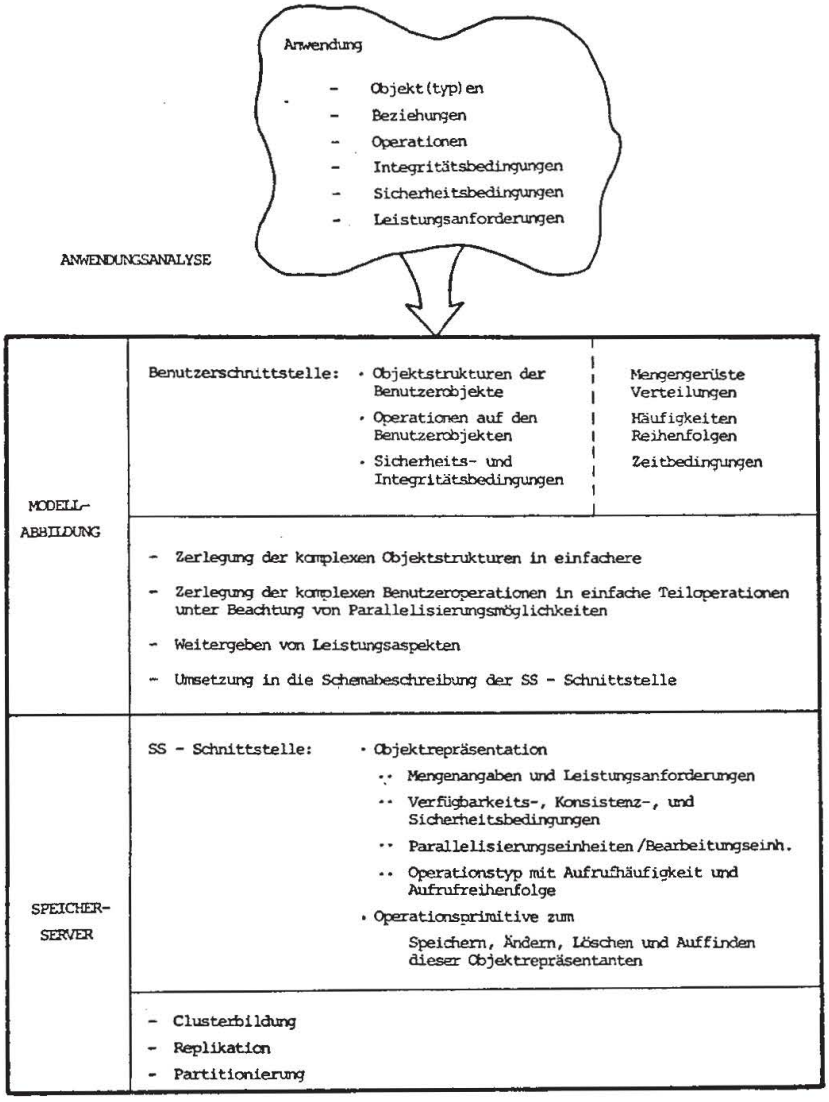

Abbildung 2: Detaillierte Darstellung des DBS-Kern-Architekturansatzes

Benutzerschnittstelle darzustellen. Damit durch diesen Modellierungsprozeß kein Informations- und Bedeutungsverlust (semantische Verlustfreiheit) auftritt, müssen die Darstellungsmöglichkeiten der Benutzerschnittstelle gegenüber der Modellierungsmächtigkeit konventioneller Datenmodelle erweitert und verbessert werden. D.h., die charakteristischen und anwendungsspezifischen Aussagen obiger Analysekriterien schlagen sich auf der Datenmodellebene nieder. Die wichtigsten Bausteine dieser Ebene sind:

- Objektstrukturen der Benutzer-Objekte und Angaben zu deren Mengengerüst, wie Verteilung, Größe und Anzahl
- Operationen auf den Benutzer-Objekten, deren Zeitbedingungen, Aufrufreihenfolgen und Aufrufhäufigkeiten
- Sicherheits- und Integritätsbedingungen inklusive Umfang, Wichtigkeit und Überprüfungshäufigkeit.

Hierdurch sind also, zusätzlich zu konventionellen Modellierungsmöglichkeiten, anwendungsspezifische Eigenschaften darstellbar. Damit ist diese Zusatzinformation für die darunterliegenden Schichten zugreifbar und kann zu Optimierungsmaßnahmen benutzt werden.

Um die Auswirkungen der Anwendungsanalyse auf die Systemschnittstellen besser dokumentieren zu können, betrachten wir, begleitend zur abstrakteren und allgemeineren Beschreibung, das folgende auf unsere Bedürfnisse reduzierte Beispiel: Ein Geometrisches Modellierungssystem (GMS) für 3D-Objekte. Die Benutzersicht umfaßt u.a. die Operationen Vereinigung, Schnitt und Differenz von dreidimensionalen Objekten. Diese zu verwaltenden Objekte untergliedern sich in Standard-Objekte, Teil- und Gesamtobjekte sowie in daraus kombinierte Objekte. Die Modellabbildung muß nun Definitionsmöglichkeiten für die oben beschriebenen 3D-Objekte zur Verfügung stellen. Den Benutzer interessiert dabei beispielsweise die Wahl des systeminternen Darstellungsschemas für die 3D-Objekte (Volumenmodell, Zellzerlegung, Begrenzungsflächenmodell, etc.; siehe /Re80/) nicht. Durch die hohe Operationsschnittstelle, die u.a. auch obige Operationen anbietet, werden die Darstellungsinterna vor dem Benutzer verborgen.

Die Aufgabe der Modellabbildungsebene ist es nun, die oben definierte Information möglichst optimal in ein internes Schema auf der Schnittstelle des Speicherservers umzusetzen. Dazu werden die häufig recht komplexen Objektstrukturen, das sind in unserem Beispiel die gewählten Darstellungsschemata für 3D-Objekte (inklusive Zusatzinformation), in einfachere zerlegt. Für die Operationen muß eine analoge Zerlegung durchgeführt werden.

Für unser Beispiel bedeutet dies folgendes: Die Darstellung des komplexen 3D-Objektes wird in 3 Teile zerlegt. Der Kombinationsteil enthält die Zusammensetzungsinformation des 3D-Objektes und wird als CSG-Baum (Volumenmodell, /Re80/) dargestellt. In der Zusatzbeschreibung werden "Volumen in mm^3", "Anzahl der Teilkörper" und weitere Zusatzinformation verwaltet. Der dritte Teil, hier Konstruktionsteil genannt, enthält die Konstruktionsdaten des 3D-Objektes; hier z.B. die Begrenzungsflächendarstellung. Die Operationen (höhere Mengenoperationen) auf 3D-Objekten verursachen auf dem Kombinationsteil und der Zusatzbeschreibung jeweils nur herkömmliche, vom Relationenmodell her bekannte, Tupeloperationen, auf dem Konstruktionsteil hingegen eine Vielzahl aufwendiger, parallel ablauffähiger Teiloperationen. Die Mengenoperationen bewirken auf der Begrenzungsflächendarstellung des Konstruktionsteils zuerst ein wechselseitiges "Verschneiden" aller Flächen der beteiligten Körper. Jede solche Teiloperation erfordert den Zugriff auf die mit Flächen, Kanten und Punkten assoziierte Information. Die dabei entstehenden neuen Kanten werden, abhängig von der jeweiligen Mengenoperation, dann zum neuen Körper rekombiniert. Sowohl für die "Verschneide"-Operation als auch für die "Rekombiniere"-Operation, die jeweils parallel durchführbar sind, stellt der Konstruktionsteil als vollständige Darstellung eines 3D-Objektes eine Bearbeitungseinheit dar. In diesem Zusammenhang versteht man unter einer Parallelisierungseinheit jeweils die ausführbare Teiloperation (hier: "Verschneide"- und "Rekombiniere"-Operation) mit dem zugehörigen Datenteil (hier: Flächeninformation aus der Begrenzungsflächendarstellung). Die Bearbeitungseinheit beschreibt damit

die der Parallelisierungseinheit übergeordnete Dateneinheit. Deshalb ist der Zugriff auf solch eine Einheit von der darunterliegenden Schicht, dem Speicherserver, aus Effizienzgründen besonders zu unterstützen.

Um solch eine Zerlegung von Benutzer-Objekt und -Operation unabhängig durchführen zu können und nicht von vornherein einzuschränken, muß der Speicherserver "beliebige Objektrepräsentationen" verwalten können. Hier wird explizit der neutrale Begriff "Objektrepräsentation" eingeführt, um zu bekräftigen, daß es sich hier nicht um herkömmliche benutzerdefinierte Objekte handeln muß. Mit Objektrepräsentation können auch relativ selbständige Teilbeschreibungen von benutzerbekannten Objekten gemeint sein. Wichtig für den Speicherserver ist die Möglichkeit diese Information, je nach Wunsch, einmal getrennt zu verwalten und das andere Mal zusammen als Ganzes. Um dieses überhaupt anbieten zu können, muß der Speicherserver über entsprechende Komponenten zur Partitionierung, Replikation und Clusterbildung sowie über geeignete Speicherungsstrukturen verfügen. Die auf den Objektrepräsentationen durchzuführenden Operationen müssen durch entsprechende Zugriffspfade unterstützt und den vorhandenen Darstellungsmöglichkeiten angepaßt sein.

Der allgemeine Speicherserver stellt daher ein System zum Speichern, Ändern, Löschen und Auffinden von physischen Objektrepräsentationen dar, die die folgenden Eigenschaften besitzen:

- beliebige Objektgrößen,
- beliebige Objektpartitionierung,
- beliebige Replikation von Objekten bzw. Objektteilen und
- Clusterbildung.

Das Attribut "beliebig" aus obiger Aufzählung ist noch zu relativieren: Die Verwaltung beliebiger Objektrepräsentationen ist die eigentliche Wunsch- und Zielvorstellung. Fürs erste genügt allerdings eine gegenüber herkömmlichen Möglichkeiten flexiblere Verwaltung der Information, wie z.B. die Clusterbildung über unterschiedlche Satztypen, eine kontrollierte, von Leistungsaspekten mitbestimmte und für die Modellabbildung transparente, Partitionierung bzw. Replikation von Objekt(teilen) oder auch größere Attributformate und Satzgrößen. Bestehende Systeme bieten maximale Attributgrößen von 255 Bytes an und erlauben nur solche "beschränkten" Objektgrößen, die kleiner als die verwendete Seitengröße - ca. 4 KByte - sind. Im Gegensatz dazu werden hier z.B. Bilder verwaltet, die eine Größe im MByte-Bereich aufweisen.

Zur Durchführung von Mengenoperationen auf 3D-Objekten sind - s.o. - nur die geometrischen Informationen über die Begrenzungsflächen wichtig; die mehr beschreibenden Informationen wie Flächengröße, etc. oder auch topologische Informationen wie Nachbarflächen, usw. dafür unwichtig. In unserem GMS-Beispiel muß dann für die geometrischen Daten eine Clusterbildung durchgeführt werden und die restlichen Daten davon partitioniert werden. Die Operationen auf den Konstruktionsdaten eines 3D-Objektes verwalten jeweils die komplette zugehörige Begrenzungsflächendarstellung, d.h. alle Begrenzungsflächen inklusive deren Kanten- und Punktdaten. Die Zusatzbeschreibung und die Kombinationsdaten werden mit Hilfe der bekannten Tupeloperationen verwaltet.

Nach oben, d.h. zur Modellabbildung hin, bietet der Speicherserver daher eine Definitions-Schnittstelle an, die die folgenden Informationen bzgl. der zu verwaltenden Objektrepräsentationen anzugeben erlaubt:

- Leistungsanforderungen und Mengengerüst,
- Parallelisierungseinheiten/Bearbeitungseinheiten,
- Verfügbarkeits-, Konsistenz- und Sicherheitsanforderungen und
- Operationstyp mit Aufrufhäufigkeit und Aufrufreihenfolge.

Vergleicht man nun die Aussagen aus Abschnitt 4.1 mit den hier spezifizierten Schnittstellen, so erkennt man die Auswirkungen der Anwendungsanalyse recht deutlich:

- Der Speicherserver stellt zum einen Primitive zum Verwalten von Objektrepräsentationen zu Verfügung. Zum anderen werden zusätzliche Datentypen und zugehörige Operationen unterstützt sowie Leistungsanforderungen berücksichtigt, die ausschließlich in Non-Standard-Anwendungen anzutreffen sind.
- Die Modellabbildungsschicht bietet dem Benutzer ein zugeschnittenes Datenmodell an. Die Kriterien, nach denen dieses Datenmodell zu bewerten ist, sind: Natürlichkeit und Einfachheit der Darstellung, Formulierbarkeit aller Integritätsbedingungen und Bereitstellen von höheren Operationen, wie z.B. Vereinigen/Schneiden von dreidimensionalen Objekten, etc. Durch die Zerlegung in einfachere Objekte und Operatoren sowie durch die Einbeziehung von Leistungsanforderungen und das Ausnutzen von Parallelisierungsmöglichkeiten kann die Umsetzung von der Benutzerschnittstelle in die SS-Schnittstelle optimiert werden.

5. Zusammenfassung

Die Motivation zur Behandlung dieser Thematik lag in der Unzufriedenheit der Anwender mit den Dienstleistungen ihrer datenbankgestützten Non-Standard-Systeme. Als typische Benutzer sind zu nennen: Stadt- und Raumplaner, Bauingenieur, Geodät, Geologe, Meteorologe, Konstrukteur im Maschinenbau, Chip-Designer, die Verantwortlichen in der Prozeßleitstelle, u.v.a.m. Die Mängel dieser ingenieurwissenschaftlichen Anwendungssysteme lagen in der schlechten Handhabbarkeit/Benutzerschnittstelle und in dem Datenmodellierungsproblem, wie in Kapitel 2 ausführlich aufgezeigt. Außer einem schlechten Leistungsverhalten ließ die Anpaßbarkeit der Systeme gegenüber Änderungen in der Anwendung stark zu wünschen übrig. Auch die Systementwicklungszeit, die Entwicklungskosten und die Implementierungskosten waren extrem hoch.

Hier wird nun eine Möglichkeit vorgeschlagen, die oben genannten Problempunkte bezüglich des zugrundeliegenden Datenbanksystems zu entschärfen bzw. zu lösen. Dazu müssen sowohl die Schnittstelle zu den Anwendungsmoduln als auch die Repräsentations- und Zugriffsmöglichkeiten des DBS verbessert bzw. die Systemarchitektur selbst überprüft werden. Zur Problemlösung wird in dieser Arbeit ein Entwurfskonzept für die Entwicklung von NDBS für Ingenieuranwendungen vorgestellt: Aufbauend auf einem hier beschriebenen und auch begründeten zweigeteilten DBS-

Architekturansatz, wird unter Einbeziehung der Ergebnisse einer ebenfalls erklärten Anwendungsanalyse versucht, eine für das spezielle Anwendungsgebiet angepaßte, möglichst optimale Systemstruktur zu gewinnen. Außer zur gezielten Informationsbeschaffung für die DBS-Architektur und deren Schnittstellen kann die Analyse auch zur erschöpfenden Untersuchung der später auf dem System zu modellierenden Anwendung benutzt werden. Würde man versuchen, ohne die Anwendungsanalyse auszukommen, dann hätte man keine angepaßte Benutzerschnittstelle mehr, dafür aber größere Modellierungsprobleme und damit einen erheblichen Verlust an semantischer Information. Man müßte mit Hilfskonstruktionen auskommen, hätte dadurch aber gleichzeitig Probleme bei der Operationsausführung und auch mit dem Leistungsverhalten des Systems. Somit bilden die Anwendungsanalyse und der hier vorgestellte DBS-Architekturansatz zusammen ein mächtiges Konzept für datenbankgestützte Non-Standard-Anwendungen.

Ich danke Herrn Professor Dr. T. Härder für die Anregung, mich mit diesem Thema zu befassen sowie für seine hilfreichen Anmerkungen während der Entstehungsphase dieser Arbeit. Bei meiner Kollegin Frau Andrea Sikeler und bei meinem Kollegen Herrn Klaus Küspert und Herrn Klaus Meyer-Wegener möchte ich mich für das sorgfältige Korrekturlesen des Manuskripts bedanken sowie bei den Referees für die hilfreichen Anmerkungen und Vorschläge.

Diese Arbeit entstand im Rahmen eines Projektes innerhalb des von der Deutschen Forschungsgemeinschaft geförderten Sonderforschungsbereiches 124.

Literaturverzeichnis

AF83 Adams, M., Ferkinghoff, B., Bender, K., Drobnik, O., Lockemann, P.C.: Datenhaltungssysteme in der Prozeßdatenverarbeitung: Ein Anforderungsprofil, Interner Bericht Nr. 16/83, April 1983

Ch81 Chang, N.S.: Image Analysis and Image Database Management, UMI Research Press, Ann Arbor, Michigan, 1981

Da81 Date, C.J.: An Introduction to Database Systems, 3rd ed. Addison-Wesley Publ. Co., 1981

Eb83 Eberlein, W.: Architektur technischer Datenbanken für Integrierte Ingenieursysteme, Dissertation, Technische Fakultät der Universität Erlangen-Nürnberg, 1983

EW81 Eberlein, W., Wedekind, H.: A Methodology for Embedding Design Databases into Integrated Engineering Systems, in: Proc. of the IFIP Conf. on CAD Data Bases, North-Holland Publ. Co., 1981

Fi83 Fischer, W. E.: Datenbanken für CAD-Arbeitsplätze, Informatik-Fachberichte Nr. 70, Springer-Verlag, 1983

Fr83 Frank, A.: Datenstrukturen für Landinformationssysteme - Semantische, topologische und räumliche Beziehungen in Daten der Geo-Wissenschaften, Dissertation, ETH Zürich, 1983

GP83 Gründig, L., Pistor, P.: Land-Informations-Systeme und ihre Anforderungen an Datenbank-Schnittstellen, in: Informatik Fachberichte 72, Sprachen für Datenbanken, Fachgespräch auf der 13. GI-Jahrestagung in Hamburg, 1983

GS82 Guttman, A., Stonebraker, M.: Using a Relational Database Management System for Computer Aided Design Data, University of California, Berkeley, College of Engineering, Electronics Research Laboratory, Memorandum No. UCB/ERL M82/37, 1982

Hä78 Härder, T.: Implementierung von Datenbanksystemen, Hanser Verlag, München, Wien, 1978

HR83 Härder, T., Reuter, A.: Database Systems for Non-Standard Applications, in: Tagungsband der ICS, Nürnberg, 1983

ID81 Proc. of the IEEE Comp. Soc. Workshop on Computer Architecture for Pattern Analysis and Image Database Management, 1981

Ka83 Katz, R.: Managing the Chip Design Database, Computer Sciences Technical Report No. 506, University of Wisconsin-Madison; auch in IEEE Computer, December 1983

Lo81 Lorie, R.A.: Issues in Databases for Design Applications, in: Proc. of the IFIP Conf. on CAD Data Bases, North-Holland Publ. Co., 1981

Lo83 Lohman, G.: Remotely-sensed Geophysical-Databases: Experience and Implications for Generalized DBMS, IBM Res. Rep., No. RJ3794, 1983

Lü83 Lüke, B.: DANTE - Ein semantisches Datenmodell für Anwendungen aus dem Konstruktionsbereich, Interner Bericht, Universität Karlsruhe, 1983

Lu83 Lum, V.: Advanced Information Management (AIM) Projektüberblick, Vortrag beim "Non-Standard DB Workshop", Heidelberg, 1983

Mi84 Mitschang, B.: Datenbankgestützte Informationssysteme für Non-Standard-Anwendungen - ein Modellierungs- und Entwurfskonzept -, Interner Bericht, Sonderforschungsbereich 124, Universität Kaiserslautern, 1984

Ne83 Neumann, Th.: On representing the design information in a common database, in Proc. of the Engineering Design Applications at the Data Base Week, 1983

NH82 Neumann, T., Hornung, C.: Consistency and Transactions in a CAD Database, in: Proc. of the 8th VLDB-Conf., 1982

PS84 Paul, H.-B., Schek, H.-J., Scholl, M., Weikum, G.: Überlegungen zur Architektur eines "Non-Standard"-Datenbankkernsystems, Arbeitsbericht DVSI1984-A2, TH Darmstadt

Re80 Requicha, A.: Representation of Rigid Solid Objects, in Computer Aided Design Modelling, System Engineering, CAD-Systems, edited by J. Encarnacao, Springer-Verlag, 1980

SL83 Scheck, H.J., Lum, V.: Position Paper and Panel on "Complex Data Objects" at 9th VLDB, Florence, Italy, October, 31 - November 2, 1983

SP82 Schek, H.-J., Pistor, P.: Data Structures for an Integrated Data Base Management and Information Retrieval System, in: Proceedings of the 8th VLDB-Conf., 1982

SW76 Stonebraker, M., Wong, E., Kreps, P., Held, G.: The design and implementation of INGRES, in ACM TODS, Vol. 1 No. 1, March 1976, p. 189

Ta81 Tang, G.Y.: A Management System for an Integrated Database of Pictures and Alphanumerical Data, in: Computer Graphics and Image Processing, Vol. 15, p. 270-286, 1981.