M A D - ein Datenmodell für den Kern eines Non-Standard-Datenbanksystems

Bernhard Mitschang
Universität Kaiserslautern

Überblick
Eine zentrale Anforderung an Datenbanksysteme für den Einsatz in den sog. nicht-konventionellen Anwendungen besteht in der anwendungsgerechten Modellierung und Verwaltung der Anwendungsobjekte. Von dem Architekturkonzept der DBS-Kern-Architektur ausgehend, werden zum einen Anforderungen an die Datenmodelle der DBS-Kern-Schnittstelle erarbeitet und zum anderen verschiedene Datenmodelle diesbezüglich analysiert. Dieser Kriterienkatalog und die (teilweise) Unzulänglichkeit der untersuchten Datenmodelle gaben den Anlaß zur Entwicklung des Molekül-Atom-Datenmodells (MAD-Modell). Hier werden nun sowohl die Modellierungs- als auch die Verarbeitungskonzepte des MAD-Modells herausgearbeitet und beispielhaft vorgestellt. Weiterhin werden die Konzepte einer SQL-ähnlichen Sprache angegeben, die das MAD-Modell an der DBS-Kern-Schnittstelle zur Verfügung stellt.

Abstract
A key requirement encountered when using database systems for non-standard applications is precise modeling and efficient management of the application objects. Starting with the concept of the DBS-kernel architecture, we point out the essential requirements for data models at the kernel interface and discuss some data models within this context. This requirement catalogue and the shortcomings of the data models investigated gave rise to the development of the molecule-atom data model (MAD model). Here, we describe and exemplify both modeling and processing concepts of the MAD model. Additionally, we illustrate the basic concepts of an SQL-like language representing the MAD model at the DBS-kernel interface.

1. Einleitung

Die Datenverarbeitung im Bereich der sog. nicht-konventionellen Anwendungen verlangt in stetig wachsendem Maße nach geeigneter Datenbankunterstützung zur Verwaltung der anfallenden Datenmengen. Die Brauchbarkeit herkömmlicher Datenbanksysteme (DBS) für einen solchen Einsatz wird allerdings zunehmend in Frage gestellt, da die Qualität und Quantität der geforderten Datenhaltung um Größenordnungen über den Möglichkeiten der kommerziellen Datenbankverarbeitung liegt. Man denke etwa an den Entwurf eines VLSI-Chips, an die Analyse bewegter Szenen in Bildfolgen oder an die diagnostischen und therapeutischen Fähigkeiten von medizinischen Expertensystemen.
Faßt man diese Mängelberichte aus /HR85,Lo85/ zusammen, so kristallisiert sich als zentrale Problemstellung - und damit auch gleichzeitig als zentrale Anforderung an "bessere" DBS (Non-Standard DBS, abgekürzt NDBS) - die anwendungsgerechte Modellierung und Verwaltung der Anwendungsobjekte heraus.
Vielversprechende Lösungsansätze basieren auf dem völlig neuen Architekturkonzept der DBS-Kern-Architektur /HR85,LD85,Mi84,PSSW84/. Dahinter verbirgt sich die Idee der Zweiteilung der NDBS-Architektur (siehe Abbildung 1.1) in einen anwendungsunabhängigen DBS-Kern (oder Speicherserver) und eine anwendungsbezogene Systemebene, Modellabbildung genannt. Die Vorteile dieses Architekturansatzes liegen vor allem darin, daß zum einen durch die Modellabbildung eine anwendungsbezogene Schnittstelle mit den benötigten Objekten und Operationen bereitgestellt werden kann und sich zum anderen im Speicherserver alle geeigneten, allgemein verwendbaren Darstellungs- und Zugriffstechniken redundanzfrei vereinigen und effizient implementieren lassen. Der Speicherserver realisiert damit ein allgemeines Datenmodell, auf dem die mit noch mehr Semantik ausgestatteten Datenmodelle der verschiedenen Anwendungsklassen aufbauen. Diese werden innerhalb der Modellabbildung durch eine optimale Transformation auf die Schnittstelle des Speicherservers abgebildet. Auf diese Weise kann vom NDBS ein anwendungsbezogenes Modellierungswerkzeug als objektunterstützende Schnittstelle bei gleichzeitiger Optimierung des Leistungsvermögens zur Verfügung gestellt werden.

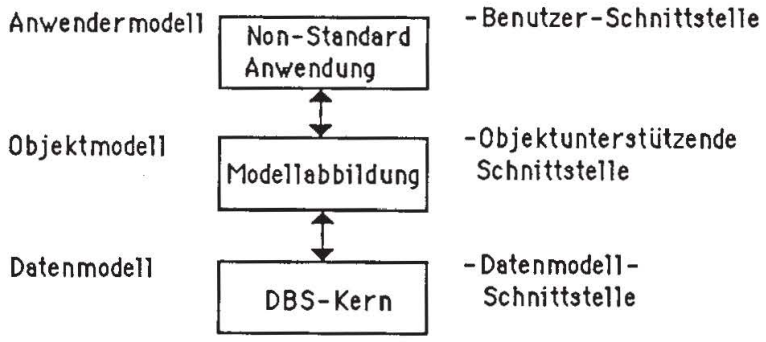


Abbildung 1.1: DBS-Kern-Architektur

Diese mehr statischen Aspekte der Datenabbildung werden in /HR85/ detaillierter behandelt und in /Hä86,HHM86/ weiter ergänzt um Aspekte des dynamischen Ablaufs, d.h. um Konzepte zur effizienten Objektverarbeitung.
Im nächsten Kapitel werden allgemeine Forderungen an das Datenmodell der DBS-Kern-Schnittstelle erarbeitet, sowie verschiedene Datenmodelle diesbezüglich untersucht und miteinander verglichen. Das dritte Kapitel konzentriert sich auf das Molekül-Atom-Datenmodell (MAD-Modell) als vielversprechenden Lösungsansatz. Dort werden die allgemeinen Modellierungs- und Verarbeitungskonzepte des MAD-Modells vorgestellt und anhand einfacher Beispiele veranschaulicht. Ein Resümee schließt die Modellbetrachtungen ab und gibt einen Ausblick auf momentan laufende und geplante notwendige Arbeiten.

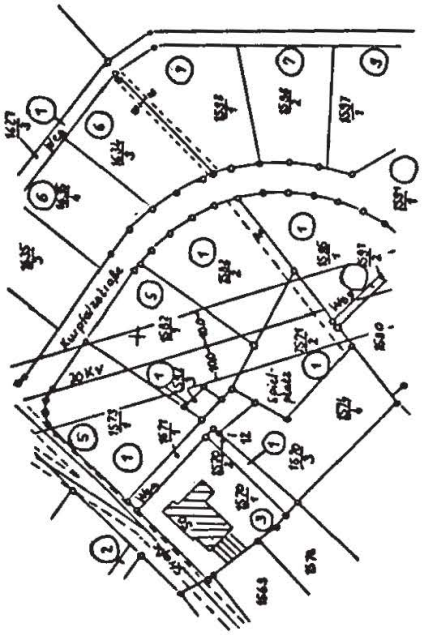
2. Datenmodelle für den DBS-Kern

Die Modellabbildung als oberste Schicht eines NDBS ist jeweils auf eine Anwendungsklasse zugeschnitten. Durch angepaßte Objekte und Operationen stellt sie nützliche Modellierungswerkzeuge für eine Anwendung zur Verfügung. Zusätzlich werden von ihr noch erforderliche Integritätsbedingungen abgewickelt. Beispielsweise könnten in einer geographischen Anwendung das Objekt PARZELLE mit den zugehörigen Operationen TEILE_PARZELLE und VEREINIGE_PARZELLE sowie die Integritätsbedingungen "Parzellen dürfen sich weder teilweise noch vollständig überlappen", "der Kantenzug einer Parzelle muß geschlossen sein" etc. realisiert sein. Zur Spezifikation solcher komplexen Objekte ist ein semantischeres Datenmodell heranzuziehen, das eine entsprechend "objektbezogene" Strukturierung erlaubt.
Genaue Analysen bzgl. der Eigenschaften dieser "Komplexobjekte" finden sich in /BB84/ und /Mi85/. Dort werden die inhärenten Objektstrukturen analysiert und aufgezeigt. In /Mi85/ werden hauptsächlich Objekte aus ingenieurwissenschaftlichen Anwendungen untersucht: Die Objektbeschreibungsdaten werden dort unterteilt in allgemeine Objektstrukturen und in davon unabhängige Entwurfsstrukturen. Letztere werden durch die verschiedenen Repräsentationen, die Versionen und Alternativen sowie die Konfigurationen gebildet und beschreiben den eigentlichen Entwurfsvorgang. Ergänzend dazu zeigen die Objektstrukturen die u.U. auch rekursive Zusammensetzung der "Komplexobjekte" aus ihren Komponenten. Hier unterscheidet man zwischen reinen Strukturierungseigenschaften, die im wesentlichen den Aufbau eines "Komplexobjekts" aufzeigen und Isolationseigenschaften, die die Nicht-Eigenständigkeit (weak relationship) und die Mehrfachbenutzung als Komponentenobjekt (shared object) definieren. Ähnlich zu /Mi85/ werden in /BB84/ die "Komplexobjekte" - dort "molecular objects" genannt - danach unterschieden, ob beim Zusammenfassen von bereits definierten Molekülen zu Komponenten eines neuen Moleküls disjunkte oder nicht-disjunkte Mengen von Komponentenmolekülen auftreten und ob eine rekursive Aufbaubeziehung vorhanden ist oder nicht. Gemäß dieser Einteilung unterscheiden Batory/Buchman vier Fälle, nämlich disjunkt/nicht-rekursiv, disjunkt/rekursiv, nicht-disjunkt/nicht-rekursiv und nicht-disjunkt/rekursiv.
An der unteren Schnittstelle der Modellabbildung sind die im "Objektmodell" dargestellten komplexen Objekte der Anwendung durch das Datenmodell des DBS-Kerns geeignet zu repräsentieren. Sowohl in /Hä86,HHLM86,Sch86/ als auch in /BB84... "support for molecular objects should be an integral part of future DBMSs"/ wird

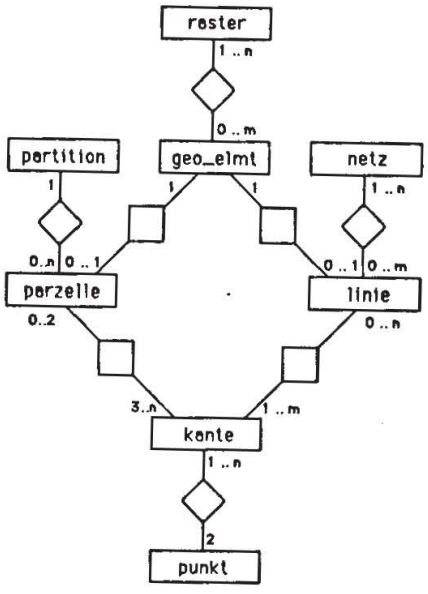
daher eine stärkere "Objektorientierung" des zugrundeliegenden Datenmodells intensiv begründet und gefordert. Der Zugriff (Holen, Kopieren) auf ein komplexes Objekt oder seine Komponenten als extrem häufige Operation verlangt vom DBS-Kern eine schnelle Ableitung und Bereitstellung der zugehörigen heterogenen Sätze oder Tupel, die meistens in netzwerkartiger Weise miteinander in Beziehung stehen. D.h., es sind effiziente Operationen für den satztypübergreifenden Zugriff (wirksame Verknüpfung Fremdschlüssel - Primärschlüssel) in beiden Richtungen erforderlich. Dazu ist eine deskriptive Sprache, die sowohl homogene als auch heterogene Satzmengen auszuwählen erlaubt, vorzusehen. Aus Leistungsgründen ist zudem eine ebenfalls mengenorientierte Bereitstellung in der Modellabbildung notwendig.

Eine geographische Beispielanwendung

Dieses leicht verständliche Beispiel ist aus dem Anwendungsbereich der geographischen Informationssysteme /Fr83,RNLE85/ entliehen. Es dient zur Verdeutlichung sowohl der Modellierungsaspekte als auch der operationalen Aspekte innerhalb der Ebene der Modellabbildung. In Abbildung 2.1 ist die ausgewählte Anwendung einmal als Kartenausschnitt und einmal in Form eines Entity-Relationship-Diagramms aufgezeigt. Der beschriebene Schemaausschnitt stellt einen zentralen und wichtigen Teil eines Landinformationssystems dar. Der Entitytyp 'punkt' trägt die metrische Information und hilft, Kantenobjekte aufzubauen. Kantenobjekte sind aus genau zwei Punktobjekten zusammengesetzt. Die Kantenbeschreibung enthält noch weitere Informationen über die Kantenlänge, Richtung etc. und bildet den Entitytyp 'kante'. Jedes Kantenobjekt dient im weiteren entweder zur Begrenzung von Flächenobjekten und/oder zum Aufbau von Linienzügen. Die somit definierbaren Objekte werden näher beschrieben durch die Entitytypen 'parzelle' und 'linie'. Die Aufteilung der Gesamtfläche in Teilflächen (die Parzellenobjekte) wird im Partitionsobjekt (etwa die Landkreise eines Regierungsbezirks) modelliert. Der Entitytyp 'partition' gibt dazu noch genauere Informationen. Das Netzobjekt, genauer beschrieben durch den Entitytyp 'netz', setzt sich zusammen aus einzelnen Linienobjekten. Mittels dieses Entitytyps werden u.a. die Versorgungs- und Entsorgungssysteme (Leitungsnetze für Gas, Wasser, Strom) sowie Verkehrsnetze (Bahnlinien-, Buslinien-, Straßennetz,...) modelliert. Die Geo-Elemente stellen entsprechende Verallgemeinerungen der zugehörigen Parzellen- bzw. Linienobjekte dar. Der Entitytyp 'geo-elmt' beinhaltet u.a. die Abstraktion von der konkreten Geometrie zu der "umschreibenden Hülle". Alle Geo-Elemente sind gemäß ihrer geometrischen Lage verschiedenen Rasterobjekten (Kartenblätter) zugeordnet, die das gesamte darzustellende Gebiet aufteilen und durch den Entitytyp 'raster' näher beschrieben werden.



a) Kartenausschnitt

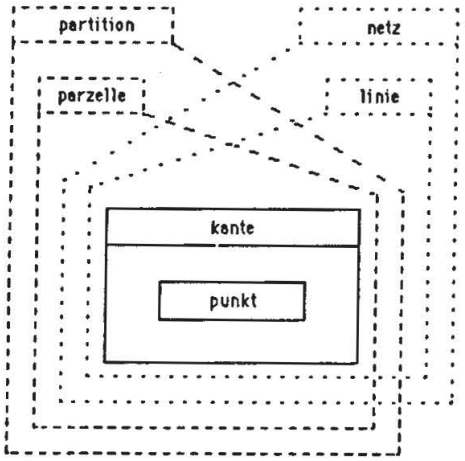


b) Entity-Relationship-Diagramm

Abbildung 2.1: Datenausschnitt eines Landinformationssystems

Objektname	Typ des Objektes	Typeigenschaften
punkt_obj	punkt	disjunkt, ¬ rekursiv
kante_obj	kante-punkt	¬ disjunkt, ¬ rekursiv
parzelle_obj	parzelle-kante_obj oder parzelle-kante-punkt	¬ disjunkt, ¬ rekursiv
linie_obj	linie-kante_obj oder linie-kante-punkt	¬disjunkt, ¬ rekursiv
partition_obj	partition-parzelle_obj oder partition-parzelle-kante-punkt	disjunkt, ¬ rekursiv
netz_obj	netz-linie_obj oder netz-linie-kante-punkt	¬disjunkt, ¬ rekursiv
geo_elmt_obj	geo_elmt-parzelle_obj alternativ zu geo_elmt-linie_obj	disjunkt, ¬ rekursiv disjunkt, ¬ rekursiv
raster_obj	raster-geo_elmt_obj	¬ disjunkt, ¬ rekursiv
parzelle_rec	parzelle-kante-parzelle	¬ disjunkt, rekursiv

a) Struktur und Eigenschaften



b) graphische Darstellung des Aufbaus

Abbildung 2.2: Die Anwendungsobjekte unseres Landinformationssystems

Diese Objektbeschreibungen sind in Abbildung 2.2 nochmals tabellarisch und graphisch zusammengefaßt. Es ist deutlich zu erkennen, daß die Anwendungsobjekte zueinander in "Aufbaubeziehungen" stehen - meistens vom (m:n)-Typ. Weiterhin wird durch die Mehrfachbenutzung etwa der Kantenobjekte die Nachbarschaft bzw. Topologie der Linien- und Parzellenobjekte explizit modelliert. Für die zugehörigen Operationen bedeutet dies, die Beziehungen zu den anderen Objekten jeweils zu berücksichtigen. Beispielsweise darf eine Kante nur dann gelöscht werden, wenn sie von keinem Linien- bzw. Parzellenobjekt mehr benötigt wird. Durch die vielen verschiedenen Objekttypen wird die Dynamik/Flexibilität der Objektbildung eindringlich aufgezeigt. Die angegebenen Objekttypen treten zum einen als eigenständige Objektbeschreibungen auf und zum anderen auch als abhängige Komponentenbeschreibungen von übergeordneten Typen.
Die zugehörigen primitiven Operationen dieser Anwendung umfassen einerseits mehr allgemeinere objektbezogene Operationen wie das Einspeichern, Löschen, Lesen und Modifizieren der Anwendungsobjekte, d.h. Parzellen-, Netzobjekte etc. Andererseits sind auch spezielle, objektspezifische Operationen relevant, wie etwa: Länge-, Umfangberechnungen, Überlappung von Flächen, Nachbarschafts- und Bereichsanfragen, Parzellenvereinigung, Parzellenteilung, Verkürzen bzw. Verlängern von Pipelines, Versorgungs-, Entsorgungsanalysen sowie Verkehrsanalysen.

Die Standard-Datenmodelle
Im folgenden werden die drei Standard-Datenmodelle kurz skizziert und hinsichtlich der oben aufgezeigten Aspekte zur Objektorientierung analysiert:
- Das Netzwerkmodell kennt sowohl Satztypen als auch Beziehungstypen (vom Typ 1:n) und macht die Verknüpfungsstruktur für netzwerkartige Beziehungen im Modell sichtbar. Es erlaubt prinzipiell einen schnellen Satztypwechsel, verlangt jedoch in der Datenmanipulationssprache eine satzweise Navigation entlang den definierten Schemastrukturen.
- Das Hierarchiemodell ist durch starke Beschränkungen der Abbildungsmächtigkeit gekennzeichnet. Die satzweisen Operationen innerhalb der Manipulationssprache sind richtungsbezogen und nur längs des im Schema vordefinierten hierarchischen Pfades durchführbar.
- Das Relationenmodell kennt die Datenbank als eine Kollektion benannter Relationen, die jeweils aus einer Menge von Tupeln bestehen. Ein Tupel ist zusammengesetzt aus einer Anzahl von atomaren Attributen. Die Darstellung von netzwerkartigen Strukturen geschieht über die symmetrischen Primärschlüssel-Fremdschlüssel-Beziehungen. Analog zu den beiden oben genannten Modellen müssen die komplexen (n:m)-Beziehungen indirekt über Hilfsrelationen modelliert werden. Der Verbund (Join) als satztypübergreifende Operation ist in der Regel sehr aufwendig und daher langsam. Eine für das Relationenmodell häufig gewählte Anfrage- und Manipulationssprache ist SQL /Da81/.

Die drei Standard-Datenmodelle beinhalten keinerlei Ansätze zur Objektorientierung. Das DBS kennt ausschließlich Sätze oder Tupel und u.U. auch deren Beziehungen untereinander. D.h., alle Aspekte eines Komplexobjekts (molecular object) müssen oberhalb des DBS realisiert werden und können daher vom DBS nicht entsprechend unterstützt, geschweige denn optimiert werden. Detaillierte Leistungsuntersuchungen an verschiedenen DB-basierten Prototypen /HHLM86,RS86,So85/ haben diese Aussage mittlerweile empirisch bestätigt.

Erweiterte Datenmodelle

Eine bessere Anpassung an die gegebenen Anforderungen wird häufig durch Modifikation von existierenden Datenmodellen bzw. durch den Entwurf neuer, stärker auf die Anwendungen bezogener sog. semantischer Modelle zu erzielen versucht. In /DKML84/ und /M185/ finden sich Zusammenstellungen von in der Literatur vorgeschlagenen Modellen. Viele dieser Vorschläge beruhen auf Erweiterungen bzw. Abwandlungen des Relationenmodells. Im folgenden werden zwei weitere Datenmodelle und deren Ansätze zur Objektorientierung vorgestellt, die als aussichtsreiche Kandidaten für das logische Datenmodell des DBS-Kerns in Frage kommen.

Die Relationenmodell-Erweiterung nach Lorie /LK84/ bietet eine rudimentäre Objektorientierung. Basierend auf den neu hinzugefügten Attributtypen IDENTIFIER und COMPONENT_OF können hierarchische Beziehungen zwischen Relationen definiert werden. Die damit festgelegte "komplexe" Struktur macht die sog. "complex objects" sichtbar, die jeweils aus einem Wurzeltupel der ausgezeichneten Wurzelrelation und allen transitiv abhängigen Komponententupeln der zugehörigen Komponentenrelationen bestehen. Ein weiterer neuer Attributtyp REFERENCE erlaubt, den Bezug auf Sätze außerhalb dieser komplexen Struktur als externe Referenz zu beschreiben. Nicht hierarchische Beziehungen können nur unter Einführung von Redundanz modelliert werden. Der Attributtyp IDENTIFIER stellt ein Surrogatkonzept zur eindeutigen Tupelidentifikation zur Verfügung. Alle mittels COMPONENT_OF und REFERENCE spezifizierten Beziehungen sind explizite Formulierungen von Primärschlüssel-Fremdschlüssel-Beziehungen, für die das System die referentielle Integrität garantiert. Da dem System die Objektnotation bekannt ist, können durch zugeschnittene interne Optimierungsmaßnahmen (auf Speicherungsstrukturebene) effiziente Operationen auf den komplexen Objekten verfügbar gemacht werden. Eine Erweiterung von SQL bzgl. der interaktiven Schnittstelle gestattet eine implizite Verbund-Formulierung längs eines vordefinierten hierarchischen Pfades. In der eingebetteten Sprachversion kann ein ganzes Objekt als Einheit selektiert und in den Arbeitsbereich des Anwendungsprogramms gelegt werden, wo es auf "Byteebene" zu manipulieren ist. Das Löschen eines komplexen Objekts geschieht durch ein explizites Löschen des Wurzeltupels und ein anschließendes kaskadiertes und automatisches Löschen aller abhängiger Tupel. Das Einspeichern und Modifizieren eines komplexen Objekts wird allerdings nicht direkt unterstützt. Abgesehen von den bekannten Relationenoperationen gibt es sonst keine weiteren Operationen. Das komplexe Objekt ist statisch festgelegt und kein Objekt der Datenbank, da es nicht mehr im Darstellungsbereich des Datenmodells liegt. D.h., ein Weiterarbeiten mit den komplexen Objekten ist nicht mehr definiert. Verglichen mit den anfangs eingeführten Aspekten von Komplexobjekten kann hier nur der disjunkt/nichtrekursive Fall direkt abgebildet werden.

Gibt man die Beschränkung der ersten Normalform auf und erweitert das Relationenmodell um relationenwertige, also nicht-atomare Attribute, so kommt man zum NF^2-Datenmodell nach Schek /SS83, PA86/. Die Datenbank wird dort angesehen als Kollektion benannter Objekte, die entweder atomar oder zusammengesetzt sind. Zusammengesetzte Objekte sind Mengen, Listen oder Tupeln, die ihrerseits wieder aus atomaren Elementen, Mengen, Listen oder Tupeln bestehen. Eine Relation im Sinne des Codd'schen Relationenmodells ist hier eine ungeordnete Menge von Tupeln. Die integrierte Objektnotation schlägt sich nieder zum einen in der expliziten Darstellung von hierarchisch gegliederten Komplexobjekten und zum anderen in der entsprechend erweiterten (SQL-ähnlichen oder algebrabasierten) Anfrage- und Manipulationssprache. Beliebige nicht hierarchische Beziehungen lassen sich hier jedoch nur mit entsprechender Redundanz modellieren, die dem System nicht bekannt ist und daher auch nicht automatisch aktualisiert werden kann. Die Operationen innerhalb eines hierarchischen Tupels lassen sich durch entsprechende interne Tupeldarstellungen sehr effizient durchführen /DGW85/. Die Sprache auf NF^2-Relationen ist

sehr ausdrucksstark und erlaubt den Zugriff sowohl auf übergeordnete Relationen im Sinne des komplexen Objekts als auch auf relationenwertige Attribute im Sinne von Teilobjekten. Für die SQL-ähnliche Sprachnotation wurde ein Mechanismus zur expliziten Beschreibung der Ergebnisstruktur einer Anfrage hinzugefügt. Im Gegensatz zu /LK84/ kann man diesen Erweiterungsvorschlag als symmetrisch bezeichnen und zwar in dem Sinne, daß die definierten Komplexobjekte - hier NF^2-Tupel genannt - auch als Einheiten manipuliert werden können. Im NF^2-Datenmodell läßt sich nur der disjunkt/nicht-rekursive Fall direkt abbilden. Dazu wird die Möglichkeit der expliziten Darstellung von hierarchisch gegliederten Komplexobjekten und die Mächtigkeit der SQL-Erweiterungen ausgenutzt. Die Rekursion und Nicht-Disjunktheit von Komplexobjekten kann auch hier nicht direkt unterstützt werden. Es gibt allerdings erste Überlegungen und Vorschläge zu deren Integration /Sch86,Li86/.
Ein Vergleich und Resümee der hier beschriebenen Datenmodelle offenbart, daß die anfangs aufgestellten Anforderungen bzgl. der Objektorientierung nur teilweise befriedigt werden. Insbesondere die Forderung nach einer direkten und symmetrischen Modellierung und auch Verarbeitung von Netzstrukturen wird von keinem Modell erfüllt. Von den vier unterschiedenen Aspekten der Objektstrukturierung wird jeweils nur die hierarchische Aggregation direkt unterstützt. Die Nicht-Disjunktheit und die Rekursion sind nur äußerst umständlich abzubilden: Die Modellierung von (m:n)-Beziehungen zwischen Komponentenobjekten bzw. deren Mehrfachbenutzung bedeutet immer die Einführung von Redundanz. Diese aus reinen Modellierungsgründen eingeführte Redundanz ist dem DBS nicht bekannt und kann daher auch nicht vom System automatisch kontrolliert bzw. aktualisiert werden. Das bedeutet einerseits, daß die gesamte Redundanzverwaltung oberhalb des DBS durchgeführt werden muß und daß andererseits keinerlei Systemunterstützung angeboten werden kann. Außerdem führt dies zu höherem Systemoverhead und höherer Speicherplatzbelegung. Im nächsten Kapitel wird nun ein neues Datenmodell vorgestellt, das diese Anforderungen besser erfüllt.

3. Das Molekül-Atom-Datenmodell

Das Molekül-Atom-Datenmodell (MAD-Modell) erlaubt sowohl eine direkte und symmetrische Modellierung als auch Verarbeitung von Netzstrukturen im Gegensatz zu den von Schek und Lorie vorgeschlagenen Modellen, die allesamt nur die Integration von hierarchischen Strukturen verfolgen. Beim MAD-Modell handelt es sich um eine Erweiterung relationaler externer Schemabeschreibungsmittel, die es ermöglicht, sowohl hierarchische als auch komplexe Beziehungen direkt und dynamisch auf einfache Datenstrukturen mit Wiederholungsgruppen abzubilden. Das erklärte Entwurfsziel des MAD-Modells ist die konsistente Erweiterung der Verarbeitung von homogenen zu heterogenen Satzmengen bzw. die Erweiterung von der bisherigen Tupelverarbeitung zur Molekülverarbeitung. Das Konzept der dynamischen Molekülbildung sowie die einfachen Molekülоperationen stellen die integralen Bestandteile des Modells dar.
Das MAD-Modell kennt, von einem abstrakten Standpunkt aus betrachtet, nur Moleküle, wobei jedes Molekül eine Struktur besitzt, die durch den zugehörigen Molekültyp festgelegt ist. Moleküle setzen sich ebenso wie die zugehörigen Typen rekursiv aus anderen Molekülen bzw. Typen zusammen. Die elementaren Bestandteile werden Atome genannt. Jedes Atom ist festgelegt durch seinen Atomtyp und repräsentiert gemäß diesem eine Zusammenfassung von Attributen mit meistens verschiedenen Attributtypen.
Man unterscheidet einfache und strukturierte sowie spezielle Attributtypen. Diese besitzen einen erweiterten Bereich von verwendbaren Datentypen (RECORD, ARRAY, Wiederholungsgruppen etc.). Der Datentyp IDENTIFIER ermöglicht die Integration eines Surrogatkonzepts auf Atom- bzw. Molekülebene. Jeder Atomtyp besitzt genau ein Attribut vom IDENTIFIER-Typ. Dadurch ist jedem Atom eindeutig ein Identifikator zugeordnet. Basierend auf diesem Datentyp erlaubt der REFERENCE-Datentyp den Bezug auf andere Atome bzw. Moleküle im Sinne eines Fremdschlüssels.
Der Molekültyp setzt sich, wie oben schon angedeutet, aus weiteren Molekül- bzw. Atomtypen gemäß einer festgelegten Struktur zusammen. Jeder Molekültyp besitzt genau einen sog. "Anker"-Atomtyp und u.U. mehrere "Komponenten"-Molekül- bzw. -Atomtypen. Dabei dient der Identifikator des Ankeratoms auch gleichzeitig als

Molekülidentifikator. Die o.g. strukturbildenden Beziehungen (auch Assoziationen genannt) werden mit Hilfe des REFERENCE-Attributtyps und der Wiederholungsgruppentypen ausgedrückt. Man unterscheidet dabei die 3 binären Assoziationstypen eindeutig (1:1), funktional (1:n) und komplex (n:m). Für jede Assoziation zwischen zwei Atomtypen A und B werden immer beide Teilbeziehungen modelliert. D.h., in A werden Referenzen auf B gemäß der Teilbeziehung A → B vermerkt und umgekehrt werden in B die Referenzen auf A gemäß der Teilbeziehung B → A gehalten. Damit ist eine symmetrische Modellierung und auch Verarbeitung gewährleistet. Außerdem können alle auftretenden Beziehungen direkt, d.h. ohne Einführung von Redundanz oder Hilfsstrukturen modelliert werden. Durch diese REFERENCE-basierten Attribute werden die Primärschlüssel-Fremdschlüssel-Beziehungen explizit formuliert (der Wert des REFERENCE-Attributes ist der IDENTIFIER-Wert des referenzierten Atoms), für die dann das System die referentielle Integrität garantiert.
Basierend auf dem Konzept der Assoziationen als struktur- bzw. molekülbildende Beziehungen kann nun eine entsprechend dynamische Objektnotation definiert werden. Ausgehend von den Atomtypen und den Assoziationen können einfache Molekültypen aufgebaut werden, die dann, ergänzt durch vorab definierte Molekültypen bzw. durch weitere Atomtypen und Assoziationen, die komplexeren Molekültypen bilden. Die so definierten Moleküle sind gemäß dem zugehörigen Molekültyp strukturiert. Dieses dynamische Molekülkonzept führt dann zu einfachen und effizienten Molekülоperationen, die durch zusätzliche interne Optimierungsmaßnahmen (auf Speicherungsstrukturebene) verbessert werden (die hier vorgesehenen Maßnahmen sind gegenüber den bekannten Standard-Konzepten um einiges anspruchsvoller und erfolgversprechender /Si86/).
Die im MAD-Modell definierte Sprache SQL* erlaubt eine recht komfortable Molekülverarbeitung. Sie ist an SQL angelehnt und besteht daher aus den drei Basiskonstrukten der SELECT-, FROM- und WHERE-Klausel, allerdings mit einer im Vergleich zu SQL erweiterten Syntax und Semantik. Die FROM-Klausel spezifiziert die Molekültypen, die für die konkrete Operation relevant sind. Restriktion und Verbundoperation werden in der WHERE-Klausel angegeben. Die SELECT-Klausel bestimmt dann die zugehörigen Projektionen und erlaubt eine Strukturierung des Anfrageergebnisses. Basierend auf diesen Basiskonzepten werden die Molekülоperationen Lesen, Einspeichern, Löschen und Ändern zur Verfügung gestellt. Das Ergebnis dieser Operationen sind wiederum Moleküle mit definiertem Molekültyp. Damit ist die Abgeschlossenheit des MAD-Modells bzgl. seiner Operationen gewährleistet.
Mit den hier vorgestellten Konzepten zur dynamischen Molekülbildung und -verarbeitung können alle Aspekte der benötigten Objektorientierung einfach und direkt zur Verfügung gestellt werden: Die Rekursion verlangt eine Assoziation, die die rekursive Beziehung ausdrückt, und die Nicht-Disjunktheit entsteht auf natürliche Art und Weise dadurch, daß sich die durch Assoziationen verbundenen Atomtypmengen überlappen. Im folgenden werden die wichtigsten Charakteristika des MAD-Modells anhand der zur Verfügung gestellten Sprache SQL* in relativ einfacher und anschaulicher Weise aufgezeigt. Dazu wird auf die in Kapitel 2 eingeführte Beispielanwendung Bezug genommen. Gemäß der natürlichen Einteilung der Sprachanweisungen in Definitions-, Lastbeschreibungs- und Manipulationskommandos zerfällt die Sprache in die drei Bestandteile
- Datendefinitionssprache (data definition language, DDL),
- Lastdefinitionssprache (load definition language, LDL) und
- Datenmanipulationssprache (data manipulation language, DML).

Eine ausführlichere und detaillierte Sprach- und Syntaxbeschreibung ist in /Mi86/ enthalten.

3.1 Die Datendefinitions- und Lastdefinitionssprache

Zur Definition der Objekte des MAD-Modells stehen die sieben in Tabelle 3.1 aufgeführten DDL-Kommandos zur Verfügung. Es lassen sich davon jeweils zwei Anweisungen derart zusammenfassen, daß beide zueinander invers sind. Die Atomtypdefinition definiert einen neuen Atomtyp, wohingegen die Anweisung 'drop' zum Löschen des angegebenen Atomtyps führt. In gleicher Weise sind die Anweisungen zum Definieren und zum Löschen eines Molekültyps zueinander komplementär. Mit Hilfe der Kommandos 'expand' bzw. 'shrink' können nachträglich noch Attribute zum Atomtyp

hinzu- bzw. von ihm weggenommen werden. Die 'rename'-Anweisung ist isoliert und erlaubt ein einfaches Umbenennen von Attribut-, Atom- und Molekültypnamen.

Name	Beschreibung
CREATE ATOM_TYPE att_name <Attributdefinitionsliste> [<Schlüsseldefinition>]	Definition eines Atomtyps durch Angabe der beschreibenden Attribute und strukturbildenden Beziehungen. Der Atomtyp bekommt einen Namen.
DROP att_name	Löschen der angegebenen Atomtyp-Definition. Zudem müssen die zugehörigen Attributdefinitionen und gegebenenfalls die zugehörigen Gegenreferenzen (s.u. expand) jeweils ausgetragen werden.
DEFINE MOLECULE_TYPE mol_name FROM <Molekülstrukturdefinition> [WHERE <Qualifikationsbedingungen>]	Definition eines Molekültyps, welcher sich aus den angegebenen Molekül- bzw. Atomtypen gemäß den festgelegten Assoziationen und Qualifikationsbedingungen aufbaut. Die Molekültypbeschreibung bekommt einen Namen.
RELEASE MOLECULE_TYPE mol_name	Freigabe des spezifizierten Molekültyps.
EXPAND ATOM_TYPE att_name BY <Attributdefinitionsliste>	Dem durch seinen Namen identifizierten Atomtyp werden die hier spezifizierten Attribute hinzugefügt. Befinden sich hierunter strukturbildende Attributtypen, also vom REFERENCE-Typ, so müssen die zugehörigen entgegengesetzten Teilbeziehungen (Gegenreferenzen) vom Benutzer entsprechend ergänzt werden.
SHRINK ATOM_TYPE att_name BY <Attributnamenliste>	Komplementäre Operation zu EXPAND-ATOM-TYPE. Die spezifizierten Attribute werden aus der angegebenen Atomtyp-Definition entfernt. Falls es sich hierbei um REFERENCE-basierte Atomtypen handelt, so wird die zugehörige Gegenreferenz automatisch vom System gelöscht. Die Liste der zu entfernenden Attribute darf kein Schlüsselattribut enthalten.
RENAME old_name,new_name	Umbenennen aller Namen. Es können sowohl Attribut- als auch Atom- bzw. Molekültypnamen geändert werden.

Tabelle 3.1: DDL-Anweisungen

```
CREATE ATOM_TYPE punkt
   (punkt_id       IDENTIFIER,
    punkt_nr       INTEGER,
    koordinate     RECORD
                      x        INTEGER,
                      y        INTEGER,
                   END,
    kanten         SET_OF ( REF_TO (kante)) (1, VAR)
   )

CREATE ATOM_TYPE raster
   (rast_id        IDENTIFIER
    pq_nr          INTEGER
    location       HULL DIM (2)
    elmts          SET_OF (REF_TO (geo_elmt)) (0, VAR)
   )

CREATE ATOM_TYPE netz
   (netz_id        IDENTIFIER,
    netz_nr        INTEGER,
    netz_typ       CHAR (30),
    beschreibung   CHAR VAR,
    teilnetz       SET_OF ( REF_TO (linie)) (0, VAR)
   )

CREATE ATOM_TYPE geo_elmt
   (geo_id         IDENTIFIER,
    abstr_box      HULL DIM (2)
    lin_obj        REF_TO (linie)
    parz_obj       REF_TO (parzelle)
    surround       SET_OF (REF_TO (raster)) (1, VAR)
   )

CREATE ATOM_TYPE linie
   (lin_id         IDENTIFIER,
    lin_nr         INTEGER,
    laenge         REAL,
    bezeichnung    CHAR (50) ,
    kantenzug      SET_OF ( REF_TO (kante)) (1, VAR),
    netze          SET_OF ( REF_TO (netz)) (1, VAR)
    abstract       REF_TO (geo_elmt)
   )
```

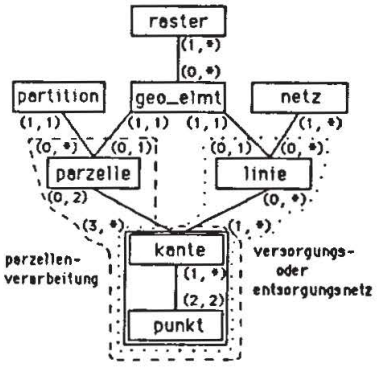
```
DEFINE MOLECULE_TYPE parzellenbegrenzung
FROM    kante-punkt
WHERE   kante.parzellen <> EMPTY

DEFINE MOLECULE_TYPE parzellenverarbeitung
FROM    parzelle-parzellenbegrenzung

DEFINE MOLECULE_TYPE linienobjekt
FROM    linie-kante-punkt

DEFINE MOLECULE_TYPE versorgungsnetz
FROM    netz-linienobjekt
WHERE   netz_typ = 'VERSORGUNG'

DEFINE MOLECULE_TYPE entsorgungsnetz
FROM    netz-linienobjekt
WHERE   netz_typ = 'ENTSORGUNG'
```




```
CREATE ATOM_TYPE kante
   (kanten_id      IDENTIFIER,
    kanten_nr      INTEGER,
    kanten_typ     CHAR (1),
    laenge         REAL,
    parzellen      SET_OF ( REF_TO (parzelle)) (0, 2),
    linien         SET_OF ( REF_TO (linie)) (0, VAR),
    punkte         SET_OF (REF_TO (punkt)) (2, 2)
   )

   KEYS ARE (kanten_nr)

CREATE ATOM_TYPE partition
   (part_id        IDENTIFIER,
    part_nr        INTEGER,
    name           CHAR (100),
    beschreibung   CHAR_VAR,
    parzellen      SET_OF (REF_TO (parzelle)) (0, VAR)
   )
   KEYS ARE (part_nr),
            (name, beschreibung)
```

```
CREATE ATOM_TYPE parzelle
   (par_id         IDENTIFIER,
    par_nr         LONGINT,
    bezeichnung    CHAR (256),
    eigentuemer    SET_OF (RECORD
                      vorname     CHAR (30),
                      nachname    CHAR (30),
                      adresse     RECORD
                                     strasse    CHAR (30),
                                     hausnr     INTEGER,
                                     plz        INTEGER,
                                     wohnort    CHAR (50),
                                  END,
                   END) (1, 5),
    begrenzung     SET_OF ( REF_TO (kante)) (3, VAR),
    partition      REF_TO (partition)
    abstract       REF_TO (geo_elmt)
   )
```

Abbildung 3.1: Beispielmodellierung der geographischen Anwendung

Ein einfaches Beispiel dieser Definitionssprache ist in Abbildung 3.1 zu sehen. Diese Abbildung reflektiert die Transformation des Entity-Relationship-Diagramms aus Abbildung 2.1 in das entsprechende MAD-Modell-Schema. Die zugehörige Visualisierung des MAD-Modell-Schemas in Diagrammdarstellung verdeutlicht die direkte Umsetzung des Entity-Relationship-Schemas in das äquivalente MAD-Schema. Jedem Entitytyp wird ein Atomtyp zugeordnet, und jeder Beziehungstyp wird in seine beiden Teilbeziehungen zerlegt und jeweils durch die strukturbildenden Assoziationen, d.h. durch REFERENCE-basierte Attribute, dargestellt. Die im Entity-Relationship-Modell darstellbaren Kardinalitätsrestriktionen der Relationship-Typen können im MAD-Modell ebenfalls angegeben werden. Diese relativ mächtigen Integritätsaussagen können nun vom System automatisch garantiert werden. Die Hinzunahme der "KEYS_ARE"-Klausel bietet eine Schlüsseldefinitionsmöglichkeit an - Attributkombinationen sind zugelassen -, die auch vom System zur Integritätskontrolle ausgenutzt wird. Durch einen Vergleich mit Abbildung 2.2 ist zu erkennen, daß auch die Transformation der Anwendungsobjekte (Objekttypen) auf die Moleküle (bzw. Molekültypen) des MAD-Modells einfach und direkt durchführbar ist. Die in Abbildung 3.2 definierten Molekültypen sind auch im zugehörigen Diagramm eingezeichnet (gepunktete und gestrichelte Linien). Die Spezifikation eines Molekültyps wird einerseits durch die Angabe der betreffenden Molekülstruktur (FROM-Klausel) als Zusammensetzung von Molekül- bzw. Atomtypen durchgeführt. Andererseits ermöglicht die optionale WHERE-Klausel eine wertbezogene Qualifikation der für den Molekültyp vorgesehenen Moleküle.
In diesem Beispiel wird die Verwendung des gegenüber herkömmlichen Datenmodellen erweiterten Angebots an Datentypen besonders deutlich. Der IDENTIFIER-Typ stellt ein Surrogatkonzept zur Identifikation aller Atome zur Verfügung und muß daher in jeder Atomtypdefinition genau einmal vorkommen. Er bietet zusammen mit dem REFERENCE-Typ die Möglichkeit einer typbezogenen "Atomreferenz". Semantisch betrachtet, bedeutet dies eine Art "Platzhalter" für das referenzierte Atom, der bezogen auf das Relationenmodell dem Fremdschlüsselwert entspricht. Mit der weiteren Hilfe des Wiederholungsgruppentypen SET_OF ist die direkte Modellierung einer (1:n)- und (n:m)-Beziehung möglich. Semantisch ist damit eine "Fremdschlüsselwiederholungsgruppe" oder eine mengenorientierte Assoziation verbunden. Durch einen kleinen Abstraktionsschritt kommt man zu der Aussage, daß man es hier anstatt mit expliziten Objekthierarchien (siehe NF^2-Datenmodell) mit implizit definierten Hierarchien zu tun hat. Jeder Atomtyp entspricht dabei genau einer Stufe in der normalerweise mehrstufigen Hierarchie. Dieses Konzept bietet im Vergleich zur expliziten Hierarchie entscheidende Vorteile:

1.) Jede Beziehung wird auf logischer Ebene symmetrisch dargestellt und kann somit auch symmetrisch, d.h. in beiden Richtungen, verarbeitet werden.
2.) (m:n)-Beziehungen können direkt und redundanzfrei modelliert werden.
3.) Die Mehrfachbenutzung von Objekten in verschiedenen Objekthierarchien ist ebenfalls direkt und redundanzfrei modellierbar (Bspl.: Ein Kantenobjekt kann sowohl eine Parzelle begrenzen als auch Teilstück einer Linie sein).

Ebenfalls sehr deutlich ist die Verwendung der restlichen neuen Datentypen zur genaueren attributweisen Modellierung zu erkennen (die variabel lange CHAR-Folge, der RECORD-Typ oder etwa der Hüllen-Datentyp). In Abbildung 3.2 sind alle verwendbaren Datentypen und deren Typbildungsregeln im sogenannten Abstammungsgraphen dargestellt. Die Typen sind dabei die Knoten des Graphen, wohingegen die Kanten die erlaubten Typbildungsmöglichkeiten repräsentieren. Tabelle 3.2 listet die wichtigsten Eigenschaften der verschiedenen Datentypen auf. Die logischen Datenstrukturen, die mit Hilfe der DDL darstellbar sind, können unter Benutzung der Lastdefinitionssprache (LDL) durch entsprechende physische Strukturen unterstützt werden. Sämtliche Anweisungen der LDL sind für den normalen Benutzer weder sichtbar noch zugänglich. Sie bieten Möglichkeiten zur Laufzeitoptimierung an und bewirken i.a. direkte Optimierungsmaßnahmen auf der Speicherungsstrukturebene /Si86/. D.h., die Mächtigkeit der Sprache wird in keinster Weise beeinflußt, allerdings sollte die Effizienz einzelner Operationen entscheidend verbessert werden. Aus diesem Grund bleiben alle LDL-Anweisungen ausschließlich dem Datenbankadministrator vorbehalten. Die Semantik der LDL-Anweisungen wurde so gewählt, daß die jeweiligen DEFINE- und RELEASE-Anweisungen sich in ihrer Wirkung aufheben. In Tabelle 3.3 sind alle LDL-Anweisungen zusammengefaßt und beschrieben.

DATENTYP	CHARAKTERISTIKA
einfache Datentypen	selbsterklärend
strukturierte Datentypen	selbsterklärend
Wiederholungsgruppentypen	
LIST_TYP	ordnungserhaltend, keine Duplikatfreiheit der Elemente
SET_TYP	nicht ordnungserhaltend Duplikatfreiheit der Elemente,
spezielle Datentypen	
CHAR_VAR	variabel lange Zeichenkette
BYTE_VAR	variabel lange Bytekette
TIME	Zeitangabe bestehend aus Jahr, Monat, Tag, Stunde, Min., Sek.
CODE	variabel langer Behälter für ausführbare Operationen
HULL	n-dimensionale Hülle
IDENTIFIER	siehe Text
REFERENCE	siehe Text

Tabelle 3.2: Charakteristika der Datentypen

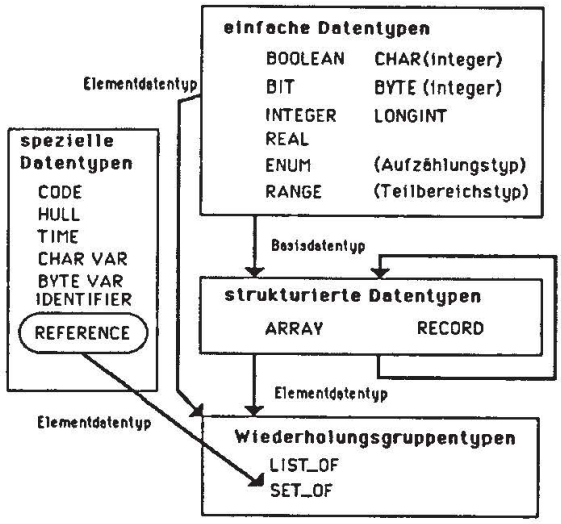


Abbildung 3.2: Abstammungsgraph der Datentypen des MAD-Modells

Name	Beschreibung
DEFINE STATIC_MOLECULE_TYPE mol_name FROM <Molekülstrukturdefinition> [WHERE <Qualifikationsbedingungen>]	Einrichten des spezifizierten Molekültyps als Speicherungsstruktur.
RELEASE STATIC_MOLECULE_TYPE mol_name	Löschen der zugehörigen Speicherungsstruktur.
DEFINE PARTITION ON att_name AS <Attributlisten>	Für den angegebenen Atomtyp wird eine neue Partitionierung definiert und in den Speicherungsstrukturen eingerichtet. Falls vorhanden, wird die bisherige Partition hierdurch überschrieben und ersetzt.
RELEASE PARTITION ON att_name	Kompensiert obige Anweisung durch eine neue Anweisung 'DEFINE-PARTITION', wobei die neue Partition jetzt nur noch aus einem Teil besteht, der alle Attribute des angegebenen Atomtyps umfaßt.
DEFINE ACCESS acc_name ON att_name FOR <Attributliste>	Definiert und benennt einen Zugriffspfad auf den angegebenen Attributen des zugehörigen Atomtyps.
RELEASE ACCESS acc_name ON att_name	Löschen des spezifizierten Zugriffspfades.
DEFINE SEQUENCE seq_name ON att_name USING <Attributliste, Sortierliste>	Definiert und benennt auf den angegebenen Attributen des Atomtyps die spezifizierte Sortierordnung.
RELEASE-SEQUENCE seq_name ON att_name	Löschen der spezifizierten Sortierordnung für den angegebenen Atomtyp.

Tabelle 3.3: LDL-Anweisungen

Diese Lastbeschreibung bietet die Möglichkeit, sowohl statische Molekültypen als auch vertikale Partitionen, Zugriffsaspekte und Sortierreihenfolgen auf Atomtypen zu definieren. Zugriffspfaddefinitionen sind sowohl für einzelne Attribute als auch für Attributkombinationen zugelassen. Letztere bieten eine Definitionsmöglichkeit für mehrdimensionale Zugriffspfade an. Die Definition eines statischen Moleküls bewirkt (normalerweise) das Einrichten eines entsprechenden physischen Clusters als Speicherungsstruktur /S186/. Zur Spezifikation eines statischen Molekültyps sind die gleichen FROM- und WHERE-Klauseln verwendbar wie bei der 'DEFINE MOLECULE_TYPE'-Anweisung innerhalb der DDL. Beide Anweisungen dürfen nicht miteinander verwechselt werden, da zwischen ihnen semantisch ein großer Unterschied besteht: Die DDL-Anweisung dient zur Definition von Molekülen auf Benutzerebene. Durch entsprechende LDL-Operationen können diese "logischen Cluster" der Benutzerebene auf entsprechende Speicherungsstrukturen im Sinne von "physischen Clustern" abgebildet werden.

3.2 Die Datenmanipulationssprache

Zur Verarbeitung der Objekte des MAD-Modells stehen die vier in Tabelle 3.4 beschriebenen Anweisungen zum Lesen, Einspeichern, Löschen und Ändern zur Verfügung. Diese Molekülоperationen dienen sowohl zur Verarbeitung eines ganzen Moleküls als auch zur Verarbeitung von Komponenten des betreffenden Moleküls, die entweder wieder Moleküle oder Atome sind. Innerhalb verschiedener DML-Anweisungen können die gleichen Sprachklauseln (meistens FROM- und WHERE-Klauseln) benutzt werden. Die Semantik dieser Klauseln bleibt dabei immer gleich. Die FROM-Klausel spezifiziert die Molekültypen, die für die konkrete Operation relevant sind. Restriktion und Verbundoperation werden in der WHERE-Klausel angegeben. Beide Klauseln dienen beim Einspeichern, Löschen und Ändern zur Spezifikation der aktuellen Umgebung, durch die die eigentlich zu bearbeitenden Moleküle bestimmt werden. Umgekehrt formuliert bedeutet dies ein inhaltsbezogenes deskriptives Ansprechen der Komponenten des (umgebenden) Moleküls, innerhalb dessen gearbeitet werden soll. Falls die Abarbeitung einer DML-Anweisung zu Änderungsoperationen von strukturbildenden, d.h. REFERENCE-basierten Attributen führt, werden die entsprechenden Aktualisierungsoperationen der Gegenreferenzen, d.h. der zugehörigen entgegengesetzten Teilbeziehung, automatisch durchgeführt. Damit wird immer eine konsistente Molekülverarbeitung garantiert.

Name	Beschreibung
MOLECULE-QUERY SELECT <Projektionsargumentlise> FROM <zu selektierende Molekültypen> [WHERE <Qualifikationsbedingungen>]	Das Primitiv jeder Anfrage-Anweisung besteht aus einer SELECT-, FROM- und WHERE-Klausel, wobei letztere auch eingespart werden kann. Die FROM-Klausel spezifiziert die Molekültypen, die für die konkrete Operation relevant sind. Restriktion und Verbundoperation werden in der WHERE-Klausel angegeben. Die SELECT-Klausel bestimmt dann die zugehörigen Projektionen und erlaubt eine Strukturierung des Anfrageergebnisses.
MOLECULE-INSERTION INSERT <Moleküldaten> INTO <einzuspeichernder Molekültyp> [FROM <umgebende Molekültypen> [WHERE <Qualifikationsbedingungen>]]	Fügt die angegebenen Moleküle des in der INTO-Klausel spezifizierten Molekültyps in die DB ein. Dabei wird die Verträglichkeit von Molekül und Molekültyp überprüft. Mit Hilfe der optionalen FROM- und WHERE-Klausel können die Umgebungen (Oberstrukturen oder umgebende Moleküle) spezifiziert werden, in die das einzuspeichernde Molekül (als Komponente) einzutragen ist.
MOLECULE-DELETION DELETE <zu löschender Molekültyp> [FROM <umgebende Molekültypen> [WHERE <Qualifikationsbedingungen>]]	Löscht die durch die WHERE-Klausel spezifizierten Moleküle des angegebenen Molekültyps aus der DB. Mit Hilfe der optionalen FROM- und WHERE-Klausel können, wie oben die Umgebungen spezifiziert werden, aus denen die zu löschenden Moleküle kommen.
MOLECULE-UPDATE UPDATE <Änderungsliste> INTO <zu ändernder Molekültyp> [FROM <umgebende Molekültypen> [WHERE <Qualifikationsbedingungen>]]	Die spezifizierten Moleküle werden gemäß der Änderungsliste aktualisiert. Dabei wird die Verträglichkeit der Änderungsliste mit dem zu ändernden Molekültyp überprüft. Optional können mittels der FROM- und WHERE-Klausel die Umgebungen angegeben werden, aus denen die zu ändernden Moleküle kommen.

Tabelle 3.4: DML-Anweisungen

Um die wichtige Forderung der Abgeschlossenheit des MAD-Modells hinsichtlich seiner DML-Operationen garantieren zu können, wird verlangt, daß das Ergebnis jeder DML-Anweisung, d.h. jeder Molekülоperation, wiederum konsistente Moleküle des zur jeweiligen Operation gehörenden semantisch korrekten Molekültyps darstellt. Jeder korrekt definierte Molekültyp besitzt genau einen Wurzel- oder Ankeratomtyp sowie evtl. mehrere Komponenten-Atomtypen bzw. -Molekültypen, die einen zusammenhängenden Graphen bilden. In diesem Kontext spricht man dann auch von der Zusammenhangsstruktur (der Komponententypen). Die Definition eines korrekten Moleküls ergibt sich dann wie folgt: Ein Molekül ist eine Ausprägung des zugehörigen Molekültyps und besitzt genau ein Wurzel- oder Ankeratom (natürlich vom Ankeratomtyp), das das Molekül eindeutig identifiziert. Weiterhin kann es mehrere Komponentenatome bzw. Komponentenmoleküle geben (ebenfalls vom entsprechenden Typ), die einen wiederum zusammenhängenden Graphen bilden. Dabei müssen nicht von jedem Komponententyp Ausprägungen existieren.

1. Hole Parzellenobjekt 117
```
SELECT  *
FROM    parzelle-kante-punkt
WHERE   par_nr = 117
```

2. Hole Parzellenobjekt 118
```
SELECT  *
FROM    parzellenverarbeitung
WHERE   par_nr = 118
```

3. Hole gemeinsamen Kantenzug
```
SELECT  *
FROM    parzellenbegrenzung
WHERE   kanten_nr ELMT (
        (SELECT kanten_nr
         FROM   parzelle
         WHERE  par_nr=117)
        INTERSECT
        (SELECT kanten_nr
         FROM   parzelle
         WHERE  par_nr=118))
```

4. Prüfe gemeinsamen Kantenzug

5. Lösche gemeinsamen Kantenzug
```
DELETE  parzellenbegrenzung
FROM    parzellenbegrenzung
WHERE   kanten-nr ELMT (3, 8)
```

6. Erzeuge Parzellenobjekt 100

7. Speichere neues Parzellenobjekt 100
```
INSERT  <parzellendaten>
INTO    parzelle-kante-punkt
FROM    parzellenverarbeitung
```

8. Lösche Parzellenobjekt 117
```
DELETE  parzelle-kante-punkt
FROM    parzellenverarbeitung
WHERE   par_nr=117
```

9. Lösche Parzellenobjekt 118
```
DELETE  parzelle-kante-punkt
FROM    parzellenverarbeitung
WHERE   par_nr=118
```

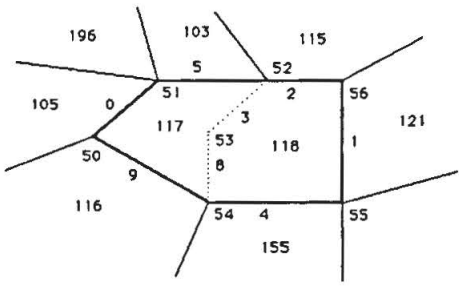


Abbildung 3.3: Teiloperationen zur "Parzellenvereinigung"

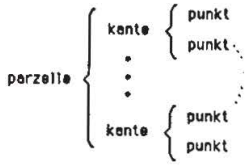
Die einzelnen DML-Anweisungen aus Tabelle 3.4 werden durch ein umfangreiches Verarbeitungsbeispiel in Abbildung 3.3 nochmals verdeutlicht. Zusätzlich wird damit auch noch der Aspekt der anwendungsbezogenen Molekülverarbeitung innerhalb der Modellabbildung der DBS-Kern-Architektur (siehe Kapitel 1 und 2) behandelt. Das gewählte Beispiel ist eine höhere Operation aus dem Bereich Stadtplanung, nämlich "Parzellenvereinigung" und basiert auf dem in Kapitel 2 vorgestellten Schemaausschnitt. Es sollen die Flächen zweier aneinander angrenzender Parzellen zusammengelegt werden, indem die gemeinsamen Kanten, d.h. der Grenzkantenzug, gelöscht und die restlichen Kanten zur neuen Parzelle zusammengefügt werden. Zuletzt ist die neue Parzelle noch einzuspeichern, und die beiden Ausgangsparzellen sind zu löschen. Schaut man sich die einzelnen Operationen aus Abb. 3.3 genauer an, so erkennt man die Adäquatheit der Molekülverarbeitung sehr deutlich: Die o.a. Verarbeitungsstrategie kann mittels der DML-Anweisungen direkt übernommen werden. Im folgenden werden die verschiedenen DML-Anweisungen näher betrachtet, indem die drei wichtigsten Sprachklauseln genauer erläutert werden (hierbei wird ständig auf Tab. 3.4 sowie auf Abb. 3.3 und 3.4 Bezug genommen).

Innerhalb der FROM-Klausel können sowohl vordefinierte Molekültypen (bzw. Atomtypen) durch ihre Namen als auch dynamisch neudefinierte Molekültypen durch ihre Strukturdefinition angegeben werden. Man unterscheidet dabei hierarchische, baumartige, vernetzte und rekursive Molekültypen. In Abbildung 3.4 sind die entsprechenden Beispielanfragen und die Visualisierung der zugehörigen Ergebnismoleküle aufgezeigt (Die gepunkteten Linien zeigen identische Atome an, die hier nur der Visualisierungsstruktur wegen mehrfach vorhanden sind, im konkreten Ergebnismolekül allerdings immer genau einmal vorkommen).

Die WHERE-Klausel spezifiziert Auswahlbedingungen (Restriktionen) auf Attributebene. D.h., die Terme der WHERE-Klausel stellen die Qualifiaktionsbedingungen für die Moleküle bzw. Atome der in der FROM-Klausel angegebenen Molekül- bzw. Atomtypen dar. Jeder Qualifikationsterm dient ausschließlich zur Qualifikation der zugehörigen Moleküle. Terme, die zwei Molekültypen zuzuordnen sind, definieren eine Verbundoperation zwischen den beteiligten Molekülen. Dieser Molekültyp-Verbund erfordert in Analogie zum Relationenverbund im Relationenmodell die Bestimmung des resultierenden Molekültyps. Die sich aus der Verbundoperation ergebende Molekülstruktur ist die Zusammenfassung der beiden beteiligten Molekülstrukturen an der Stelle der Verbundoperation. Analog zur Verbundoperation im Relationenmodell, die dort zur Zusammenlegung der beiden beteiligten Relationen führt, werden hier die beiden beteiligten Atomtypen (bzw. die Atome) zu dem resultierenden Atomtyp (bzw. Atom) zusammengefaßt. Damit ist gewährleistet, daß das Ergebnis einer Verbundoperation wiederum aus Molekülen des resultierenden Molekültyps besteht. Gleichzeitig bedeutet dies, daß das vorläufige Anfrageergebnis, auf das sich etwaige nachfolgende Projektionen beziehen, immer durch einen korrekten Molekültyp festgelegt ist.

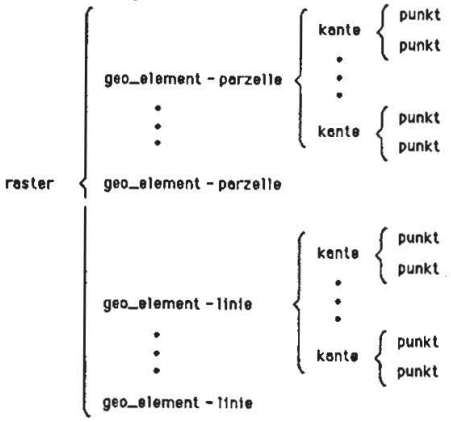
In der SELECT-Klausel können zusätzlich zur Strukturbildung innerhalb der FROM- und WHERE-Klausel noch explizit Strukturierungsmaßnahmen angegeben werden. Die unqualifizierte Projektion bestimmt, welche Komponenten bzw. Komponentenattribute das

1. hierarchische Ergebnismoleküle
"Selektion des Parzellenobjektes mit der Nummer '999'!"



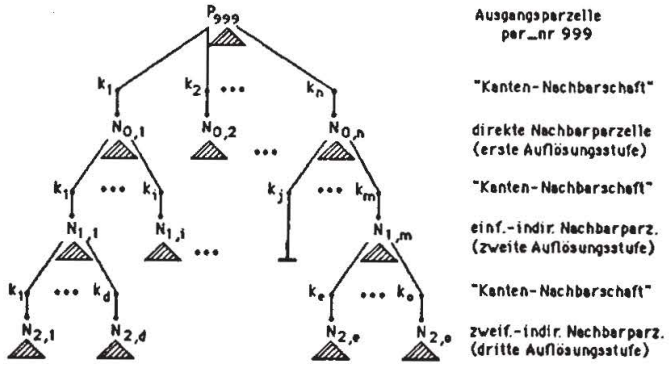

```
SELECT *
FROM   p_obj (parzelle-kante-punkt)
WHERE par_nr = 999
```

2. vernetzte Ergebnismoleküle
"Selektion des Rasterobjektes mit der Planquadratnummer '113'. Dabei sollen nur die Geo-Elemente ausgegeben werden, deren umschreibende Hülle völlig innerhalb des Planquadrates liegt!"




```
SELECT raster_obj
       (raster,
       geo_elmt
       (SELECT *
        FROM geo_obj
        WHERE surround INSD location)
       )
       (* qualifiziere Projektion *)
FROM   raster_obj  (raster-geo_obj
                        (geo_elmt-
                           (parzelle, linie)-
                           kante-punkt
                        )
                   )
WHERE  pq_nr = '113'
```

3. rekursives Ergebnismolekül
"Selektion der "Nachbarschaftsstruktur" und aller direkten sowie ein- und zweifach indirekten Nachbarparzellen
zur Parzelle mit der Nummer '999', wobei jede Nachbarparzelle nur genau einen Eigentümer haben darf!"



(das Zeichen △ steht stellvertretend für ein ganzes Parzellenmolekül P_OBJ)
(das Zeichen ⊥ zeigt einen Abbruch der Rekursionsauflösung aufgrund der until-Klausel)

```
SELECT *
FROM   P_OBJ  (parzelle-kante-punkt),
       P_REK(P1  (parzelle)-kante-P2(parzelle))
             (recursive,
             until (#REC = 3 OR
                    NUM_ELMT (P2.eigentümer ≠ 1))
             )
WHERE  P_OBJ.par_id = P_REK.P1.par_id
                      (* Molekül-Join *)
       AND
       SEED (P_REK).P1.par_nr = '999'
                    (* Saatinitialisierung *)
```

Abbildung 3.4: Visualisierung spezieller Ergebnismoleküle

Ergebnis definieren. Will man genau diese Entscheidung von bestimmten Auswahlbedingungen abhängig machen, so bedient man sich der qualifizierten Projektion. Hierzu ist die Verwendung des Anfrageprimitives bestehend aus SELECT-, FROM- und WHERE-Klausel innerhalb der SELECT-Klausel erlaubt. Sowohl die qualifizierte als auch die unqualifizierte Projektion müssen dabei die Integritätsbedingungen bzgl. Molekül und zugehörigem Molekültyp erfüllen, so daß gilt: Jede Anfrage definiert genau einen korrekten (Ergebnis-)Molekültyp und liefert als Ergebnis eine Menge von Molekülen des entsprechenden Typs. Die Ergebnismenge kann dabei leer, ein- oder mehrelementig sein.
Aus Platzgründen wird auf eine detaillierte und nach den verschiedenen DML-Anweisungen getrennte Diskussion verzichtet (sämtliche Details können in /Mi86/ nachgelesen werden). Die Charakteristika dieser Anweisungen und auch der gesamten Molekülverarbeitung sind schon durch Tabelle 3.4 sowie durch die Abbildungen 3.3 und 3.4 herausgearbeitet. Lediglich der Aspekt der rekursiven Anfragen (Beispiel 3 aus Abb. 3.4) wird im weiteren noch vertieft: Der rekursive Molekültyp wird durch die 'recursive'-Klausel definiert. Jedem rekursiven Molekültyp (P_REK in Abb. 3.4)

liegt ein Komponenten-Molekültyp (parzelle-kante) zugrunde, von dessen Typ die Komponentenmoleküle sind, die das rekursive Molekül aufbauen. Zusätzlich gibt es noch eine ausgezeichnete Assoziation, die typmäßig einen Zykel (Kanten-Nachbarschaft) herbeiführt und damit die Rekursion überhaupt erst ermöglicht. Basierend auf dem Komponenten-Molekültyp, der zugehörigen zykelbildenden Assoziation und der Spezifikationen innerhalb der 'recursive'-Klausel, ist eine Abarbeitungsvorschrift (siehe /Mi86/) zur Bestimmung der einzelnen Komponentenmoleküle festgelegt, wodurch indirekt auch eine Spezifikation der rekursiven Moleküle erfolgt. Die 'until'-Klausel definiert Disqualifikationsbedingungen für die Komponentenmoleküle und realisiert daher das Abbruchkriterium für die rekursive Auswertung. Das '# REC'-Prädikat kann zur Beschränkung der Anzahl der Rekursionsschritte verwendet werden (im Beispiel werden nur 3 Rekursionsstufen zugelassen). Unabhängig von den konkreten Parameterwerten der 'recursive'-Klausel wird immer eine Vermeidung von Doppel- und Mehrfachauswertungen des gleichen Rekursionspfades gewährleistet. Daher und auch wegen der endlichen Anzahl von Atomen innerhalb der Datenbank ist die Terminierung jeder Rekursionsauswertung garantiert. Durch ein ausgezeichnetes Prädikat innerhalb der WHERE-Klausel kann eine Saat-Initialisierung angegeben werden: Die Qualifikationsterme innerhalb der SEED-Klausel qualifizieren ausschließlich die Wurzelkomponenten der rekursiven Moleküle (im Beispiel ist nur das Molekül mit dem Wurzelatom mit der Nummer '999' zugelassen).

4. Zusammenfassung und Ausblick

In dieser Arbeit wurde das Molekül-Atom-Datenmodell zur Verwaltung von komplexen Objekten vorgestellt und dessen Existenzberechtigung eingehend begründet. Das größere Umfeld der Betrachtungen ist definiert durch den Bereich der sog. Non-Standard-DBS-Anwendungen und gekennzeichnet durch die zentrale Problemstellung der anwendungsgerechten Modellierung und Verwaltung der Anwendungsobjekte. Ausgehend von dem vielversprechenden Lösungsansatz der DBS-Kern-Architektur, d.h. der Zweiteilung der DBS-Architektur in einen anwendungsunabhängigen DBS-Kern (Speicherserver) und eine anwendungsbezogene Ebene (Modellabbildung), werden innerhalb der Modellabbildung die komplexen Objekte der Anwendung mit Hilfe des Datenmodells an der Kern-Schnittstelle (hier MAD-Modell) realisiert. Dabei ist eine entsprechende Objektorientierung - im Sinne einer objektbezogenen Modellierung und Verarbeitung - des zugrundeliegenden Datenmodells erforderlich. Ein Objekt besteht hierbei aus u.U. heterogenen Komponenten, die meistens in netzwerkartiger Weise miteinander verbunden sind.

Eine Analyse verschiedener ausgewählter Datenmodelle bzgl. des Aspekts der Objektorientierung ergab wichtige Hinweise zur Konzeption des MAD-Modells:

- Die Standard-Datenmodelle (Netzwerk-, Hierarchie- und Relationenmodell) stellen keinerlei Ansätze zur Objektorientierung zur Verfügung.
- Die Relationenmodell-Erweiterung nach Lorie bietet eine rudimentäre Objektorientierung. Nur die Modellierung von Objekten, deren Komponenten jeweils hierarchisch angeordnet sind, ist möglich. Die Manipulation dieser sog. 'complex objects' ist allerdings nur in sehr eingeschränktem Maße erlaubt.
- Eine mehr symmetrische Objektnotation wird im NF^2-Datenmodell angeboten. Hier können die komplexen Objekte modelliert und auch später als Einheit manipuliert werden. Die Klasse der so unterstützten Objekte ist allerdings wieder auf den hierarchischen Fall beschränkt.

Aus diesen Untersuchungen heraus ergaben sich die wichtigsten Charakteristika des MAD-Modells:

- direkte und symmetrische Modellierung und Verarbeitung von Netzstrukturen
- konsistente Erweiterung von der herkömmlichen Tupelverarbeitung zur Molekülverarbeitung, wobei Moleküle über (u.U. heterogenen) Komponenten, die auch in netzwerkartiger Weise zueinander in Beziehung stehen können, definiert sind
- Konzept der dynamischen Molekülbildung
- Konzeption einfacher (aber mächtiger) Molekülоperationen.

Die für das MAD-Modell definierte Sprache SQL* gliedert sich in Anweisungen zur Definition, Manipulation und Lastbeschreibung. Diese mengenorientierte und deskriptive Sprache erlaubt die Verarbeitung sowohl von aus (heterogenen) Komponenten zusammengesetzten Molekülen als auch von rekursiv definierten Molekülen. Sie bietet die Möglichkeit zur dynamischen Molekülbildung und stellt einfache

Operationen zur Molekül- bzw. zur Komponentenverarbeitung bereit (SELECT, INSERT, DELETE und UPDATE von Molekülen bzw. von Molekülkomponenten). Unterstützt durch ein reichhaltiges Angebot an verwendbaren Datentypen erlaubt der Datendefinitionsteil eine genaue Modellierung der relevanten Anwendungsobjekte. Die verfügbaren Anweisungen zur Lastbeschreibung stellen gegenüber herkömmlichen Sprachen anspruchsvolle Erweiterungen dar, die zur Optimierung der physischen Speicherungsstrukturen verwendet werden und damit ausschließlich zur Verbesserung des DBS-Leistungsverhaltens dienen.
Alle aufgeführten Aspekte des MAD-Modells sollen im Rahmen der NDBS-Entwicklung PRIMA (PRototyp-Implementierung des MAD-Modells) realisiert werden. Diesbezüglich müssen allerdings noch einige weiterführende Untersuchungen vorab durchgeführt werden, die die Gebiete der Anfrageübersetzung, der optimalen Zugriffspfadauswahl und der Cursorverwaltung, d.h. der Abbildung bzw. Abarbeitung der MAD-Operationen auf der internen Systemschnittstelle der Satz- und Zugriffspfadverwaltung, betreffen. Ebenso wichtig erscheinen die Aspekte der Molekülbereitstellung und -verarbeitung in der Modellabbildung. Speziell hierzu wurden einerseits verschiedene Prototypentwicklungen von DB-basierten VLSI-, GEO- und CAD-Anwendungssystemen durchgeführt. Andererseits wurde eine vereinfachte MAD-Schnittstelle oberhalb eines existierenden netzwerkartigen DBS realisiert (siehe System PROTON in /KMP86/). Die mit Hilfe dieser "Lernsysteme" gewonnenen Erkenntnisse müssen teilweise noch auf die DBS-Kern-Architektur und das MAD-Modell übertragen werden. Aktuelle Überlegungen, die ebenfalls kurzfristig zu Prototypentwicklungen /Ma86/ führen sollen, betreffen den Bereich der Künstlichen Intelligenz. In gleichem Maße muß auch die Verwaltung der Entwurfsstrukturen (Repräsentation, Version, Alternative und Konfiguration) analysiert und integriert werden.

Danksagung
Ich danke Herrn Prof. Dr. T. Härder und den Mitarbeitern der AG Datenverwaltungssysteme für die hilfreichen Anmerkungen während der Entstehungsphase dieser Arbeit. Bei meiner Kollegin Frau Andrea Sikeler und bei meinen Kollegen Herrn Dr. Klaus Meyer-Wegener und Herrn Christoph Hübel möchte ich mich zusätzlich für das sorgfältige Korrekturlesen des Manuskripts bedanken.

5. Literaturverzeichnis

BB84 Batory, D.S., Buchmann, A.P.: Molecular Objects, Abstract Data Types and Data Models: A Framework, in: Proc. of 10th VLDB Conf., Singapore, 1984, pp. 172-184.

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

DGW85 Deppisch, U., Guenauer, J., Walch, G.: Speicherungsstrukturen und Adressierungstechniken für komplexe Objekte des NF^2-Relationenmodells, in: Proc. GI-Fachtagung "Datenbanksysteme in Büro, Technik und Wissenschaft", IFB94, Springer-Verlag, Karlsruhe, 1985, S. 441-459.

DKML84 Dittrich, K.R, Kotz, A.M., Mülle, J.A., Lockemann, P.C.: Datenbank-konzepte für Ingenieuranwendungen: eine Übersicht über den Stand der Entwicklung, in: Proc. GI-14. Jahrestagung, IFB88, Springer Verlag, Braunschweig, 1984, S. 175-192.

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

Hä86 Härder, T.: New Approaches to Object Processing in Engineering Databases, International Workshop on Object-Oriented Database Systems (OODBS), Pacific Grove, California, September 1986.

HHLM86 Härder, T., Hübel, Ch., Langenfeld, S., Mitschang, B.: KUNICAD - ein datenbankgestütztes geometrisches Modellierungssystem für Werkstücke, Forschungsbericht des SFB 124, Nr. 22/86, Universität Kaiserslautern, in: Informatik - Forschung und Entwicklung, 1986.

HHM86 Härder, T., Hübel, Ch., Mitschang, B.: Use of Inherent Parallelism in Database Operations, in: Proc. of Conference on Algorithms and Hardware for Parallel Processing CONPAR 86, Lecture Notes in Computer Sciences, Springer Verlag, Aachen, 1986.

HR85 Härder, T., Reuter, A.: Architektur von Datenbanksystemen für Non-Standard-Anwendungen, in: Proc. GI-Fachtagung "Datenbanksysteme in Büro, Technik und Wissenschaft", IFB94, Karlsruhe, 1985, S. 253-286 (einge-ladener Vortrag).

KMP86 Käfer, W., Mitschang, B., Profit, M.: PROTON - ein Prototyp zur Verwaltung von komplexen Objekten, Forschungsbericht des SFB 124, Nr. 30/86, Universität Kaiserslautern.

LD85 Lum, V., Dadam, P., et al.: Design of an Integrated DBMS to Support Advanced Applications, in: Proc. of 2nd Int. Conf. on Foundations of Data Organization, Kyoto, 1985, pp. 21-31 (invited talk).

Li86 Linnemann, V.: Non First Normal Form Relations and Recursive Queries: An SQL-Based Approach, IBM Scientific Center Heidelberg, West-Germany, 1986 (zur Veröffentlichung eingereicht).

LK84 Lorie, R., Kim, W., et al.: Supporting Complex Objects in a Relational System for Engineering Databases, IBM Research Report, IBM Research Laboratory, San Jose, 1984.

Lo85 Lockemann, P.C., et al.: Anforderungen technischer Anwendungen an Datenbanksysteme, in: Proc. GI-Fachtagung "Datenbanksysteme in Büro, Technik und Wissenschaft", IFB94, Karlsruhe, 1985, S. 1-26.

Ma86 Mattos, N.: Modellierung von FRAME-Konzepten mit dem MAD-Modell, Interner Bericht, Nr. 164/86, Universität Kaiserslautern.

Mi84 Mitschang, B.: Überlegungen zur Architektur von Datenbanksystemen für Ingenieuranwendungen, in: Proc. GI-14. Jahrestagung, IFB88, Springer-Verlag, Braunschweig, 1984, S. 318-334.

Mi85 Mitschang, B.: Charakteristiken des Komplex-Objekt-Begriffs und Ansätze zu dessen Realisierung, in: Proc. GI-Fachtagung "Datenbanksysteme in Büro, Technik und Wissenschaft", IFB94, Karlsruhe, 1985, S. 382-400.

Mi86 Mitschang, B.: MAD - ein Datenmodell zur Verwaltung von komplexen Objekten, Forschungsbericht des SFB 124, Nr. 20/85, Universität Kaiserslautern (überarbeitet im Sommer 1986).

PA86 Pistor, P., Anderson, F.: Designing a Generalized NF^2 Data Model with a SQL-Type Language Interface, Proc. of 12th VLDB Conf., Kyoto, August 1986.

PSSW84 Paul, H.-B., Schek, H.-J., Scholl, M., Weikum, G.: Überlegungen zur Architektur eines "Non-Standard"-Datenbanksystems, Arbeitsbericht Nr. DVSI-1984-A2, TH Darmstadt, 1984.

RNLE85 Ramm, I., Neumann, K., Lipeck, U.W., Ehrich, H.-D.: Eine Benutzerschnittstelle für geowissenschaftliche Datenbanken, Informatik-Bericht 85-08, TU Braunschweig, 1985.

RS86 Reuter, J., Schulz, M.: Datenstrukturen geographischer Informationssysteme, Diplomarbeit, Universität Kaiserslautern, 1986.

Sch86 Schek, H.-J.: Komplexe und molekulare Objekte, Frames und KL-ONE-Concepts, NF^2-Relationen: Eine Gegenüberstellung, Manuskript des Vortrages auf dem Workshop "Datenbanken und Expertensysteme" in Dortmund, Mai 1986.

Si86 Sikeler, A.: Die Schnittstelle der Satz- und Zugriffspfadverwaltung des NDBS PRIMA, (Arbeitstitel), Forschungsbericht des SFB 124, Universität Kaiserslautern (in Vorbereitung).

So85 Sottong, W.: Datenstrukturen im VLSI-Schaltungsentwurf, Diplomarbeit, Universität Kaiserslautern, 1985.

SS83 Schek, H.-J., Scholl, M.: Die NF^2-Relationenalgebra zur einheitlichen Manipulation externer konzeptueller und interner Datenstrukturen, in: "Sprachen für Datenbanken", IFB72, Springer-Verlag, Hamburg, 1983.

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