in: SFB-Bericht 21/91, University Kaiserslautern, 1991

Das Molekül-Atom-Datenmodell in der praktischen Anwendung - eine Zwischenbilanz -

Harald Schöning

SFB 124
Forschungsbericht Nr. 21/91
Universität Kaiserslautern
Erwin-Schrödinger-Straße
D-6750 Kaiserslautern

Überblick

Zur Unterstützung von Non-Standard-Anwendungen wie z.B. den VLSI-Entwurf wurde an der Universität Kaiserslautern das Non-Standard-Datenbanksystem PRIMA entwickelt. Es ist nach dem Prinzip der Datenbankkernsystem-Architektur konzipiert und implementiert an seiner Kernschnittstelle das Molekül-Atom-Datenmodell. Dieses Datenmodell erlaubt die Verarbeitung komplexer, dynamisch definierter Objekte und wurde speziell für den Gebrauch in Non-Standard-Anwendungen entworfen. Im Rahmen dieses Aufsatzes sollen nun diskutiert werden, wie gut das Molekül-Atom-Datenmodell die Anwendungen wirklich unterstützt, und wie gut es sich in das Gesamtsystem einpaßt. Dabei wird herausgearbeitet, wie die Konzepte des Molekül-Atom-Datenmodells in den bisher realisierten Anwendungen benutzt werden, ob sie den Anwendungen angemessen sind, und welche kleineren oder größeren Verbesserungen eine noch bessere Unterstützung der Anwendungen ermöglichen können.

1. Einleitung

Herkömmliche Datenbanksysteme sind für den Einsatz in sogenannten Non-Standard-Anwendungen (wie z.B. VLSI-Entwurf, CAD, Wissensbankverwaltung) nicht geeignet. Hauptkritikpunkte sind dabei:

(1) mangelnder Anwendungsbezug bei der Datenmodellierung, insbesondere mangelnde Unterstützung von komplexen Objekten, wie sie für diese Anwendungen als typisch angesehen werden,

(2) fehlende oder mangelhafte Datentypunterstützung,

(3) ein unzureichendes Transaktionskonzept, das “lange” Transaktionen, wie sie als typisch für Entwurfsvorgänge angesehen werden, nicht unterstützt,

(4) die fehlende Möglichkeit, komplexe Integritätsbedingungen zu formulieren und zu prüfen,

(5) ein unbefriedigendes Leistungsverhalten.

Aus diesen Gründen wurden für den Einsatz in Non-Standard-Bereichen neue Datenbanksysteme entwickelt, die durch Behebung der oben genannten Punkte eine bessere Unterstützung der Anwendungen erreichen sollen.

Die dabei verfolgten Ansätze unterscheiden sich drastisch. So wurde bei STARBURST [Schw86] das Hauptgewicht auf eine Erweiterbarkeit im Bereich Datentypen (2) und Komplexobjektverarbeitung (1) gelegt, wobei als Basis das Relationenmodell dient. Punkt (5) wird durch die Möglichkeit von Speicherstrukturerweiterungen berücksichtigt. In dem Datenbanksystem-Generierungsansatz von EXODUS [Ca86] ist das Transaktionsmodell (3) festgelegt (und berücksichtigt keine “langen” Transaktionen), während bzgl. Datentypen und Datenmodell keine Festlegung getroffen ist. Die Leistungsmerkmale (5) eines mittels EXODUS erzeugten Datenbanksystems lassen sich über eine Vielzahl von Speicherungsmethoden beeinflussen. Die strukturell-objektorientierten Datenbanksysteme zielen hauptsächlich auf eine Verbesserung des Punktes (1) ab, indem sie ein Datenmodell anbieten, das den Umgang mit komplexen Objekten erlaubt. Dies ist im allgemeinen mit einer Unterstützung eines reicheren Datentypkonzeptes (2) verbunden. Beispiele für solche Datenmodelle sind NF^2 [SS86] und das Molekül-Atom-Datenmodell [Mi88]. Damit ist nur das Datenmodell des Datenbanksystems festgelegt. Eine Unterstützung zumindest der Punkte (3) und (5) durch andere Aspekte der Datenbankarchitektur ist zusätzlich möglich. Zu nennen ist hier besonders das Konzept der Datenbanksystemkern-Architektur [HR85, Lo85, Pa87], das folgendermaßen charakterisiert werden kann:

Das Non-Standard-Datenbanksystem besteht danach aus zwei Komponenten: dem Datenbanksystemkern, der ein anwendungsunabhängiges Datenmodell unterstützt, und einer anwendungsabhängigen “Modellabbildungsschicht”, die die Operationen und Objekte der Anwendung auf das Datenmodell des Datenbanksystemkerns abbildet. Dabei wird unterstellt, daß der Datenbanksystemkern mehrbenutzerfähig ist und auf einem Server-Rechnersystem (Monoprozessor, Multiprozessor oder lose gekoppeltes Rechnernetz) abläuft, während die Modellabbildung, die nicht notwendigerweise Mehrbenutzerbetrieb unterstützen muß, auf einer Workstation abläuft. Mit diesem Ansatz soll der Entwicklungsaufwand für anwendungsspezifische Datenbanksysteme reduziert werden: es muß nämlich nur noch die anwendungsspezifische Modellabbildung entwickelt werden, während Aufgaben, wie sie in jedem Datenbanksystem vorkommen (Recovery, Transaktionsverwaltung, Synchronisation, ...), durch den Datenbanksystemkern erledigt werden. Zudem kann die Entwicklung der Modellabbildungsschicht wirkungsvoll durch Werkzeuge unterstützt werden, die die Charakteristika des Datenbanksystemkerns berücksichtigen. Ein anderer wichtiger Aspekt dieser Architektur ist natürlich die Tatsache, daß viele Berechnungen lokal auf der Workstation vorgenommen werden können und somit der Server-Rechner entlastet wird. Es wurde

gezeigt, daß das Leistungsverhalten eines solchen Systems sehr stark von dem Grad an Lokalität der Verarbeitung abhängt, die damit erreicht wird. Ist diese hoch, sind also die Interaktionen mit dem Server-Rechner selten, so ist von dieser Architektur ein befriedigendes Leistungsverhalten (5) zu erwarten. Auf der anderen Seite erfordert dies natürlich eine “Objektpufferung” in der Workstation, die durch entsprechende Mechanismen unterstützt werden muß.

Der Kritikpunkt (3) wird im allgemeinen durch ein geschachteltes Transaktionskonzept [Mo81] berücksichtigt, das die Existenz “langer” Transaktionen in Betracht zieht.

Eine Kombination der Konzepte “strukturell-objektorientiertes Datenmodell”, Datenbanksystemkern-Architektur und geschachtelte Transaktionen scheint also ein gangbarer Weg zu sein, um ein Datenbanksystem zu schaffen, das die neuen Anwendungsgebiete zufriedenstellend unterstützt. Eine solche Kombination wurde in dem Non-Standard-Datenbanksystem PRIMA [HMMS87] realisiert. Es ist als Datenbankkernsystem konzipiert und implementiert das Molekül-Atom-Datenmodell (MAD-Modell).

Ziel des vorliegenden Aufsatzes ist es zu untersuchen, wie gut die inzwischen realisierten Anwendungen von PRIMA unterstützt werden. Dabei wird das Hauptaugenmerk auf das MAD-Modell gelegt werden. Folgende Fragen müssen in diesem Zusammenhang beantwortet werden:

(1) Wie ist das MAD-Modell bezüglich allgemeiner Gesichtspunkte zu bewerten (Mächtigkeit im Vergleich zu anderen Datenmodellen, Orthogonalität, Abgeschlossenheit, Benutzerfreundlichkeit, Möglichkeiten zur Formulierung von Integritätsbedingungen)?

(2) Ist das Komplexobjektkonzept, das das MAD-Modell anbietet, für die Anwendungen brauchbar?

(3) Bietet das MAD-Modell eine für die Anwendungen ausreichende Datentypunterstützung?

(4) Bietet das Datenmodell alle von den Anwendungen benötigten Konzepte an?

(5) Bietet das Datenmodell Konzepte an, die von keiner der Anwendungen benötigt werden?

(6) Ist der Einsatz des Konzeptes der geschachtelten Transaktionen mit der Verwendung des MAD-Modells verträglich?

(7) Wie verträgt sich eine Objektpufferung in der Workstation mit einem Datenbanksystemkern, der das MAD-Modell implementiert?

Zur Beantwortung dieser Fragen wird in Kapitel 2 zunächst das MAD-Modell in Grundzügen vorgestellt. Kapitel 3 diskutiert einige von Anwendungen unabhängige Betrachtungen zum Datenmodell selbst (1). Kapitel 4 beschreibt die bisher auf PRIMA entwickelten Anwendungen, so daß wir in Kapitel 5 zeigen können, wie diese die Konzepte des MAD-Modells nutzen (2-5). Um die Wechselwirkungen des MAD-Modells mit dem Objektpufferansatz und der Transaktionsverwaltung in Kapitel 7 untersuchen zu können (6-7), werden diese Komponenten in Kapitel 6 kurz skizziert. Kapitel 8 faßt die Ergebnisse kurz zusammen und verweist auf weitere offene Fragen.

2. Ein Überblick über das Molekül-Atom-Datenmodell

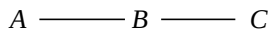
Grundbausteine des MAD-Modells sind die Atome, die mit Tupeln des Relationenmodells vergleichbar sind: sie gehören zu genau einem Typ (Atomtyp, vergleichbar der Relation) und bestehen aus Attributwerten, die jeweils einen bestimmten Typ haben. Dabei sind neben den herkömmlichen Typen INTE-

GER, REAL, BYTE, BOOLEAN, CHARACTER auch komplexere Typen zugelassen: die Zeittypen TIME und DURATION (für Zeitpunkte und Zeitdauern), der Typ HULL, der ein 1-, 2- oder 3-dimensionales Rechteck (also eine Linie, ein Rechteck oder einen Quader) beschreibt, sowie Wiederholungsgruppen (LIST, SET) und Felder (ARRAY) der Basistypen. Ferner gibt es noch einen speziellen Typ IDENTIFIER, der ein systemvergebenes Surrogat zur eindeutigen Kennzeichnung eines Atoms enthält, weshalb jeder Atomtyp genau ein Attribut vom Typ IDENTIFIER enthalten muß. Binäre Beziehungen zwischen Atomtypen werden durch Linktypen beschrieben. Ein Linktyp ist ungerichtet und wird durch ein Paar von Attributen (eines in jedem betroffenen Atomtyp) des Typs REFERENCE dargestellt. Der Wert eines Attributs vom Typ REFERENCE ist eine Wiederholungsgruppe von IDENTIFIER-Werten. In Beispiel 2.1 ist der Link zwischen den Atomtypen A und B durch das REFERENCE-Attributpaar ($A.ab$, $B.ba$) dargestellt. Entsprechend werden Links eines solchen Linktyps durch die Werte der entsprechenden REFERENCE-Attribute modelliert. Ein Link zwischen den Atomen $a1$ vom Typ A und $b1$ vom Typ B bedeutet, daß der Wert von ab in $a1$ den IDENTIFIER-Wert von $b1$ enthält, und entsprechend der Wert von ba in Atom $b1$ den IDENTIFIER-Wert von $a1$. Somit entsteht durch die Links ein ungerichtetes Netz von Atomen, und durch die Linktypen ein ungerichtetes Netz von Atomtypen. Wird ein Atom neu eingefügt, das Referenzen zu anderen Atomen hat, so sorgt das System automatisch dafür, daß die referentielle Integrität gewahrt bleibt, daß also die Links zwischen den Atomen korrekt durch gegenseitige Referenzen realisiert werden.

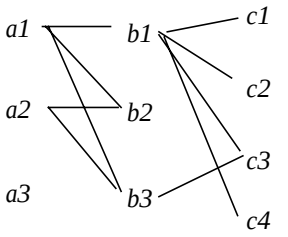
Definition dreier Atomtypen:

```
CREATE ATOM_TYPE A ( ida : IDENTIFIER;
                     groesse : REAL;
                     ab : REFERENCE TO B.ba [0,3]);

CREATE ATOM_TYPE B ( idb : IDENTIFIER;
                     name : LIST OF CHARACTER;
                     ba : REFERENCE TO A.ab;
                     bc : REFERENCE TO C.cb);

CREATE ATOM_TYPE C ( idc : IDENTIFIER;
                     laenge: INTEGER;
                     cb : REFERENCE TO B.bc [1,*]);
```

Atomtypnetz



beispielhaftes Atomnetz

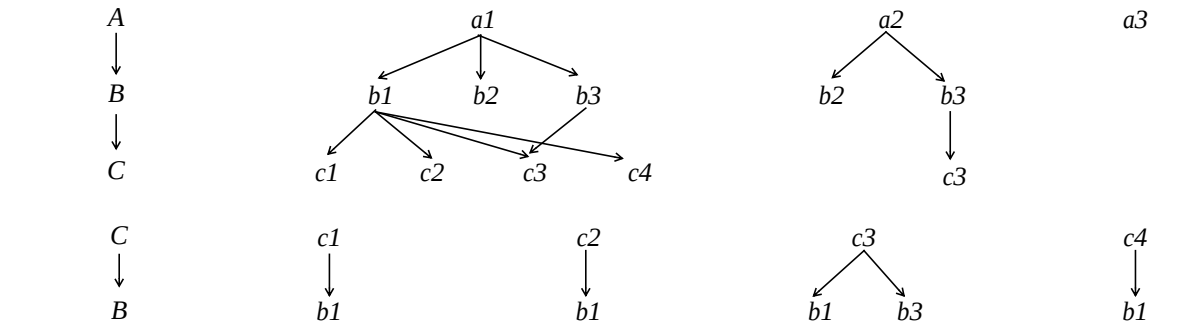


Das MAD-Modell erlaubt die Angabe von Kardinalitätsrestriktionen für Wiederholungsgruppentypen (REFERENCE, LIST, SET), d.h., die Angabe von minimalen und maximalen Elementanzahlen. Von dieser Möglichkeit wurde in diesem Beispiel bei $A.ab$ Gebrauch gemacht (ein A-Atom hat zwischen 0 und 3 Referenzen zu B-Atomen) und bei $C.cb$ (ein C-Atom hat mindestens eine Referenz zu einem B-Atom).

Beispiel 2.1: Beispielhafte Atomtypdefinitionen und dazugehöriges Atomnetz

In Anweisungen der MAD-Modell-Anfragesprache MQL (molecule query language) werden nun über dem Atomtypnetz dynamisch gerichtete Subgraphen definiert, die sogenannten Molekültypen (Molekültypgraphen). Ein Molekültyp ist ein zusammenhängender Graph mit genau einer Wurzel (der sogenannte Wurzelatomtyp). Diese Molekültypen definieren gerichtete Ausschnitte des Atomnetzes, die zugehörigen Moleküle. Dabei entspricht jedem Atom vom Wurzelatomtyp genau ein Molekül. Durch Verfolgen der Links, die durch die zum Molekültyp gehörigen gerichteten Linktypen spezifiziert sind, erhält man alle weiteren Atome, die zum Molekül gehören (Beispiel 2.2). Neben diesen Atomen gehören auch die (gerichteten) Links, die zur Bestimmung der Atome geführt haben, zum Molekül. Beispiel 2.2 illustriert,

daß Moleküle eines Molekültyps unterschiedlich “vollständig” bezüglich der in der Molekültypdefinition auftretenden Atomtypen sein können. So enthält das dritte Molekül zu der oberen Molekültypdefinition weder Atome vom Typ *B* noch vom Typ *C*.



Beispiel 2.2: Zwei Molekültypdefinitionen und die dadurch spezifizierten Molekülmengen

Die zwei verschiedenen Molekültypdefinitionen in Beispiel 2.2 zeigen, daß es möglich ist, in jeder Anfrage einen anderen Ausschnitt des Molekültypnetzes zu wählen und eine andere Richtung zuzuordnen. Dies wird durch die Symmetrie der Links ermöglicht. Ferner zeigt das Beispiel, daß sich die durch einen Molekültyp definierten Moleküle überlappen können, d.h., sie können einige Atome gemeinsam haben. Innerhalb eines Moleküls kann es außerdem mehrere Wege zu demselben Atom geben, wie das zu *a1* gehörige Molekül zeigt. Molekültypdefinitionen bilden den Kern der MQL-Anweisungen, die folgende Gestalt haben:

[**SELECT** | **INSERT** | **UPDATE** | **DELETE**] Projektionsliste
FROM Molekültypdefinition
WHERE Bedingung.

Durch die Bedingung kann eine Untermenge der durch die Molekültypdefiniton spezifizierten Molekülmenge ausgewählt werden, auf die sich die Operation beschränkt (vgl. Beispiel 2.3). Da von jedem Atomtyp außer dem Wurzelatomtyp mehrere Atome zu einem Molekül gehören können, ist eine Quantifizierung der Bedingung unerläßlich. Wenn kein Quantor explizit angegeben ist, geht MQL von einer existentiellen Quantifizierung aus. Die Bedingung von Beispiel 2.3 (Mitte) ist somit äquivalent zu der explizit existentiell quantifizierten Schreibweise **`EXISTS B: B.name="egon".`**

Die Projektionsliste legt die Atome und Attribute der Molekülmenge fest, auf die sich die Operation bezieht. Dabei kann eine Auswahl entweder aufgrund des Typs geschehen, oder wieder durch Bedingungen gesteuert (qualifizierte Projektion) werden (s. Beispiel 2.3).

keine Projektion

```
SELECT ALL
FROM   A-B
WHERE  A.groesse = 7
```

Suche alle Moleküle vom Typ A-B, bei deren A-Atom groesse = 7 gilt und zeige das gesamte Molekül

unqualifizierte Projektion

```
SELECT A
FROM   A-B
WHERE  B.name="egon"
```

Suche alle Moleküle A-B, die ein B-Atom mit name = "egon" enthalten, und zeige davon die A-Atome

qualifizierte Projektion

```
SELECT A, B, (SELECT C
              FROM RESULT
              WHERE C.laenge<A.groesse)
FROM   A-B-C
```

Suche alle Moleküle A-B-C und zeige davon alle A- und B-Atome und die C-Atome, für die laenge < A.groesse gilt.

Beispiel 2.3: Beispielanfragen in MQL

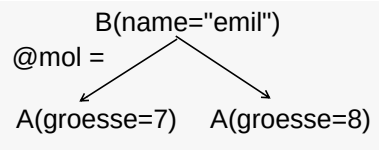
Für Änderungsoperationen gibt es zwei verschiedene Möglichkeiten, die entsprechenden Werte zu spezifizieren (s. Beispiel 2.4): zum einen können diese in der Anfrage selbst aufgeführt sein (als Konstanten oder Ausdrücke, die auch **SELECT**-Anweisungen enthalten dürfen), zum anderen können sie durch eine sogenannte Molekülmengenvariable separat angegeben werden (diese Möglichkeit ist hauptsächlich für die Benutzung von MQL aus Programmen heraus vorgesehen).

explizite Wertangabe

```
INSERT B (name="emil"),
       A(groesse=7),A(groesse=8)
FROM   B-A
```

Wertangabe über Molekülmengenvariable @mol

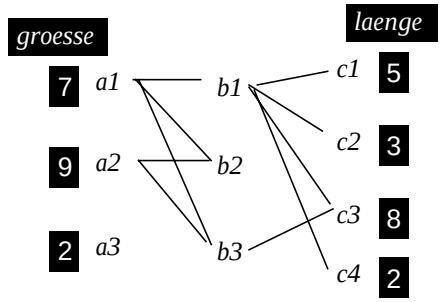
```
INSERT @mol:(B,A)
FROM   B-A
```



Beispiel 2.4: Zwei äquivalente **INSERT**-Anweisungen

SELECT-Anweisungen liefern jeweils eine Molekülmenge, so daß sie auch an Stelle von Molekültypdefinitionen eingesetzt werden können (Abgeschlossenheit des MAD-Modells). Die Molekülmenge wird dabei in einem Datenteil, der die Attributwerte aller zugehörigen Atome enthält, und einem Strukturteil, der die eigentliche Molekülstruktur bestimmt, dargestellt. Diese Darstellung ist nicht-redundant, d.h., Daten zu einem Atom sind nur einmal enthalten, unabhängig davon, in wievielen Molekülen dieses Atom enthalten ist, und wieviele Wege innerhalb eines Moleküls zu diesem Atom führen. Der Umfang des Strukturteiles richtet sich nach der Komplexität der Anfrage: Wenn die Menge der Nachfolger eines Atomes nicht davon abhängig ist, zu welchem Molekül dieses Atom gehört, so ist diese Nachfolgermenge ebenfalls nur einmal in dem Strukturteil enthalten. Betrachtet man jedoch die qualifizierte Projektion aus Beispiel 2.3, so wird klar, daß die Nachfolgermenge durchaus von der Molekülzugehörigkeit abhängen kann. Beispiel 2.5 illustriert dies.

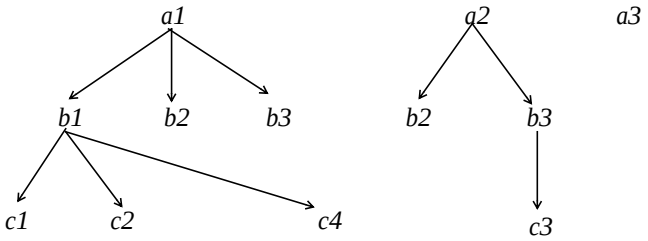
Atomnetz (mit Attributwerten)



Anfrage mit qualifizierter Projektion

```
SELECT A,B,( SELECT C
             FROM   RESULT
             WHERE  C.laenge < A.groesse)
FROM   A-B-C
```

Ergebnismolekülmenge



Beispiel 2.5: Molekülabhängige Nachfolgermengen durch qualifizierte Projektion bei Atom *b3*

Neben den bisher gezeigten baumartigen Typgraphen sind auch netzwerkartige und rekursive Molekültypgraphen erlaubt. In einem netzwerkartigen Molekültypgraphen gehören Atome eines Atomtyps, der mehrere Eingangskanten hat, nur dann zum Molekül, wenn alle eingehenden Linktypen instanziiert werden können. Bei einem rekursiven Graphen werden Komponentenmoleküle definiert, die rekursiv anein-

andergesetzt werden. Dabei gibt es zwei unterstützte Semantiken [Schö89]: die Angabe des Schlüsselwortes **RECURSIVE** spezifiziert die Transitive-Hülle-Semantik. Durch Aggregations- und Konkatenationsoperatoren kann die "generalisierte transitive Hülle" berechnet werden. Eine Einschränkung der transitiven Hülle ist über ein Abbruchkriterium Q möglich (formuliert als **UNTIL Q**), das entweder die Tiefe der Rekursion oder die Eigenschaften der Rekursionsebenen einschränkt. Die Semantik der Rekursion wird durch folgenden Algorithmus bestimmt: Zunächst werden die Startpunkte der Rekursion gefunden (die Wurzelatome mit den dazugehörigen Komponentenmolekülen). Dann werden alle Moleküle der nächsten Rekursionsstufe, die noch nicht im Molekül enthalten sind und die das Abbruchkriterium nicht erfüllen, in das Molekül aufgenommen. Dies wird fortgesetzt, bis es keine solchen Moleküle mehr gibt. Die Terminierung der Rekursion ist garantiert, da die Datenbank endlich ist, und die entstehenden Moleküle schleifenfrei sind. Die zweite mögliche Semantik wird durch das Schlüsselwort **REC_PATH** angezeigt, das bewirkt, daß alle möglichen schleifenfreien Pfade durch das Atomnetz aufgebaut werden. Analog zu **RECURSIVE** kann auch hier ein Abbruchkriterium definiert werden.

In diesem kurzen Überblick wurden nur die für diesen Aufsatz relevanten Konzepte von MQL und des MAD-Modells dargestellt. Eine ausführlichere Beschreibung findet sich in [Mi88].

3. Beobachtungen zum MAD-Modell und zu MQL

In diesem Kapitel werden nun einige Bewertungen des MAD-Modells bezüglich allgemeiner anwendungsunabhängiger Gesichtspunkte versucht, und zwar hinsichtlich der Fragen

- wie steht es um die Unterstützung externer Schemata?
- Wie mächtig ist das Typkonzept des MAD-Modells?
- Wie mächtig sind die Operationen des MAD-Modells im Vergleich zu denen anderer Datenmodelle?
- Wie steht es mit der Abgeschlossenheit des MAD-Modells?
- Ist das MAD-Modell benutzerfreundlich?
- Welche Möglichkeiten bietet das MAD-Modell zur Spezifikation von Integritätsbedingungen?

Unterstützung externer Schemata

Die Definition von Molekültypen über die DDL (**DEFINE MOLECULE_TYPE**) kann man als Mittel zur Sichtenbildung im MAD-Modell ansehen. Es sind jedoch keinerlei Sprachmittel oder Konzepte vorgesehen, die ein Autorierungskonzept auf einer Sichtenbildung unterstützen. Externe Schemata lassen sich daher mit dem MAD-Modell und PRIMA nicht realisieren. Es entsteht jedoch die Frage, inwieweit sich die klassischen Konzepte der Rechtevergabe und Sichtenbildung auf die neuen Anwendungsklassen, für die das Datenmodell entwickelt wurde, übertragen lassen.

Typkonzept

Im Gegensatz zu eNF^2 [Da86] gibt es im MAD-Modell keine beliebige *Schachtelung von Typkonstruktoren* (also z.B. kein LIST OF SET OF REAL). Insbesondere kann der "Tupel-Konstruktor" (die Atomtypdefinition) nicht geschachtelt angewandt werden (was dem Konzept des Datenmodells auch entgegenlaufen würde). Das Konstrukt SET OF TUPLE wird durch den REFERENCE-Typ ersetzt.

Ebensowenig ist es möglich, *benutzerdefinierte Datentypen* oder Operationen in das System einzubringen, wie dies etwa in POSTGRES [St86] oder AIM-P [Li87] möglich ist. Es ist noch nicht klar, ob diese beiden Punkte eine starke Einschränkung für die Modellierungsmächtigkeit des MAD-Modells darstellen. Als Vorgriff auf Kapitel 5 läßt sich allerdings feststellen, daß keine unserer bisher realisierten Anwendungen Anforderungen an das Typkonzept hatte, die über das vom MAD-Modell gebotene hinausgehen.

Die ursprüngliche Unterscheidung von zwei Typen von REFERENCE-Attributen, nämlich REF_TO als Referenzattribut mit genau einem Wert und SET OF REF_TO als mehrwertiges Attribut wurde aufgegeben, da dieselbe Aussagemächtigkeit bereits durch ein anderes Konzept, nämlich die Kardinalitätsrestriktionen, die im Schema angegeben werden können, gegeben ist. Daher entspricht REFERENCE dem ehemaligen SET OF REF_TO und REFERENCE mit den Kardinalitätsrestriktionen [1,1] dem ehemaligen REF_TO.

Obwohl im MAD-Modell vorgesehen, wurden die strukturierten Datentypen SET und RECORD bisher in PRIMA nicht implementiert. Allerdings zeigte sich auch in keiner der im folgenden Kapitel beschriebenen Anwendungen eine Notwendigkeit dafür. Daher stellt sich die Frage, ob das Vorsehen solcher Datentypen wirklich so gewinnbringend ist, wie bei der Definition des Datenmodells postuliert.

Vergleich mit anderen Datenmodellen

Das MAD-Modell ist im Gegensatz zu NF^2 [SS86] in der Lage, nicht-redundante (also überlappende) Komplexobjekte zu repräsentieren. Ebenso können auch netzwerkartige und rekursive Komplexobjekte dargestellt werden. Im Gegensatz zu NF^2 legt die Schemadefinition nicht die Struktur (und implizit die Abspeicherung) der komplexen Objekte fest. Vielmehr wird die Struktur dynamisch in jeder Anfrage neu definiert. Ein ausführlicher Vergleich des MAD-Modells mit anderen “neuen” Datenmodellen ist nicht Thema dieses Aufsatzes. Hier sollen nur noch kurz die operationalen Aspekte des MAD-Modells mit denen “klassischer” Datenmodelle verglichen werden. Insbesondere stellt sich die Frage, ob die operationalen Möglichkeiten des Relationenmodells und der Netzwerkmodelle durch das MAD-Modell abgedeckt werden.

Da das MAD-Modell eine Erweiterung des Relationenmodells darstellt, sind die Basisoperationen des Relationenmodells natürlich auch darin enthalten. Weitergehende Operationen, wie sie SQL z.B. durch die Möglichkeit der Aggregation auf Gruppierungen bietet, fehlen dagegen im MAD-Modell.

Die Netzwerke, mit denen das MAD-Modell umgeht, legen es nahe, Vergleiche mit Netzwerk-Datenmodellen zu ziehen. Dort findet eine durch Anwendungsprogramme gesteuerte navigierende Verarbeitung statt. Im MAD-Modell wird die Navigation durch das Atomtypnetz deskriptiv in einer Anfrage spezifiziert und mengenorientiert ausgewertet. Die Mengenorientierung bewirkt allerdings, daß diese Navigation nur vertikal (also in der Richtung “Owner-Member” oder “Member-Owner” in der Netzwerkmodellterminologie) möglich ist. Auf eine semantiktragende Reihenfolge der gleichberechtigten Atome (der “Member” in einem Set) kann nicht Bezug genommen werden (im Gegensatz zu “FETCH FIRST WITHIN SET” im CODASYL-Vorschlag). Solch eine Navigation ist allerdings im MAD-Modell auch nicht erforderlich. Hinzu kommt, daß hier die Symmetrie von Beziehungen weit besser unterstützt wird als im Netzwerkmodell, und daß alle Arten von Beziehungen (1:1, 1:n, n:m) im MAD-Modell im Gegensatz zum Netzwerkmodell direkt abbildbar sind (von daher kann man bei einer n:m-Beziehung nicht von “Owner” und “Member” sprechen. Folgende Betrachtungen beziehen sich also der Vergleichbarkeit wegen auf 1:n-Beziehungen).

Ein weiterer interessanter Vergleichspunkt ist die SET-MEMBERSHIP-Klausel in CODASYL (MANUAL/ AUTOMATIC und OPTIONAL/MANDATORY/FIXED). Im MAD-Modell kann man ähnliche Semantiken von Linktypen erreichen, vorausgesetzt es gibt zusätzlich zu den Kardinalitätsrestriktionen im Datenbankschema die im Punkt "*Formulierung von Integritätsbedingungen"* angesprochene Möglichkeit, Nullwerte für bestimmte Attribute zu verbieten. Die Semantik des AUTOMATIC, nämlich daß zum Zeitpunkt des Einspeicherns die Links für ein Atom eingerichtet werden (entspricht dem Einhängen in ein Set) erreicht man durch Verbieten von Nullwerten für die entsprechenden REFERENCE-Attribute. MANDATORY kann man über die Kardinalitätsrestriktion [1,*] (d.h., das REFERENCE-Attribut muß mindestens ein Element haben, von dem Atom muß also mindestens ein Link ausgehen) spezifizieren. Bei MANUAL MANDATORY entsteht hier das Problem, daß das entsprechende REFERENCE-Attribut wieder auf einen Nullwert verändert werden darf, was nicht genau die entsprechende Semantik des CODASYL-Vorschlags widerspiegelt. Im MAD-Modell kann man darüber hinaus dann noch Semantiken angeben wie "Dieser *Satz muß mindestens zu zwei Set-Ausprägungen gehören"* oder "Dieser *Satz darf nicht Owner eines leeren Set sein"*, etc., was im Netzwerkmodell nicht möglich ist. Die Semantik von "FIXED" (das Atom ändert seine Links in einem Linktyp nicht) läßt sich jedoch im MAD-Modell nicht formulieren, da keine Übergangsbedingungen formuliert werden können.

Letzter Vergleichspunkt soll hier die ERASE-Semantik sein. Im Netzwerkmodell sind hier ja sehr mächtige Formulierungen möglich, die sich auf das Löschen abhängiger Sätze beziehen. Nun wurde oben schon festgestellt, daß man im MAD-Modell wegen der Symmetrie der Beziehungen nicht von abhängigen Sätzen sprechen kann. Daher ist es hier erforderlich, entsprechende Molekültypen zu spezifizieren, die gelöscht werden sollen. Eine dem Netzwerkmodell ähnliche Semantik entstehen aus dem Konzept des Umgebungsmolekültyps. Neben der Spezifikation des zu löschenden Typs in der **DELETE**-Klausel wird in der **FROM**-Klausel die Umgebung spezifiziert, aus der er zu löschen ist. Es werden nur solche Moleküle gelöscht, die keine Außenbeziehungen außerhalb der spezifizierten Umgebung haben. Dieser Punkt soll hier aber nicht vertieft werden.

Bei der Verwendung von Quantoren in der **WHERE**-Klausel ist es nicht möglich, zwischen Wertebereich und Variable zu unterscheiden (durch Angabe des Wertebereiches wird implizit eine Variable gleichen Namens definiert). Dadurch können Bedingungen wie "in *dem Molekül gibt es zwei verschiedene Atome des Typs A, die denselben Wert im Attribut a haben"* nicht formuliert werden. Das wäre bei einer expliziten Variablenbenennung durchaus möglich: der logische Sachverhalt $\exists\alpha\in A\ \exists\beta\in A$: $\alpha.a=\beta.a \wedge \alpha\neq\beta$ könnte in MQL vielleicht so formuliert werden:

```
EXISTS AVAR1(A): EXISTS AVAR2(A): AVAR1.a = AVAR2.a AND AVAR1 <> AVAR2.
```

Abgeschlossenheit

Die Abgeschlossenheit des MAD-Modells ist insofern nicht vollständig, als daß Konstrukte der Form

```
INSERT s:= (SELECT ... FROM A-B WHERE ... ) FROM A-B;
```

nicht möglich sind. Mit anderen Worten: aus der Datenbank extrahierte Molekülmengen können so nicht wieder eingefügt werden. Die Ursache liegt hier darin, daß beim Einfügen einer durch **SELECT** gewonnenen Molekülmenge nicht unterschieden werden kann, ob eine Referenz in einem Atom in der Datenmenge auf ein neues Atom (in derselben Datenmenge) oder auf ein Atom der bereits existierenden Datenbank verweisen soll. Um dies zu ermöglichen, wurden temporäre Identifier eingeführt, deren Wertebereich disjunkt zu dem der permanenten Identifier ist. Abhilfe wäre hier möglich durch Einführung einer Sprachform **SELECT ... INTO** <Definition des neu zu erzeugenden Molekültyps>

Benutzerfreundlichkeit

Selbstverständlich bedingt ein mächtiges Modell auch eine mächtige und damit oft nicht leicht handhabbare Anfragesprache. Begnügt man sich mit der Verwendung der Grundkonzepte, so sind MQL-Anfragen jedoch immer noch leicht verständlich und auch leicht erzeugbar. Hier könnten trotzdem noch einige Kleinigkeiten verbessert werden:

- Die Formulierung der Abbruchbedingung der Rekursion durch eine Ausschlußbedingung (**`UNTIL Q`**) ist nicht immer einsichtig. Hier wäre es als Unterstützung für den Modellierer hilfreich, wenn auch eine positive Formulierung gestattet wäre (also z.B.: **`WHILE Q`** mit der Semantik: *setze Rekursion fort, solange Bedingung Q gilt*).
- Momentan muß der Benutzer bei der Schemadefinition Linktypen durch Paare von REFERENCE-Attributen selbst realisieren, wobei das System die Konsistenz dieser Definition prüft. Es würde jedoch dem Konzept des MAD-Modells besser entsprechen, wenn Atomtypen und Linktypen separat definiert werden könnten, d.h. zunächst Atomtypen ohne REFERENCE-Attribute, und danach Linktypen, die vom System zu REFERENCE-Attributpaaren umgesetzt werden. Inkonsistenzen im Schema würden so schon auf operationaler Ebene verhindert. Zur Zeit muß ein Schema noch daraufhin überprüft werden, ob zu jedem REFERENCE-Attribut auch das korrekte Gegen-REFERENCE-Attribut definiert wurde. Ist dies nicht der Fall (was in größeren Schemata leicht auftreten kann), so wird der Benutzer zu einer Korrektur gezwungen, bevor das Schema akzeptiert wird.
- Die Semantik von MQL-Anfragen ist nicht immer so einfach, wie sie auf den ersten Blick scheint. Als Beispiel dafür möge der Abbruch der Rekursion stehen, über den es in Kapitel 2 heißt: "... werden alle Moleküle der nächsten Rekursionsstufe, die ***noch nicht im Molekül enthalten sind*** und die das Abbruchkriterium nicht erfüllen, in das Molekül aufgenommen...". Diese einfache Formulierung hat jedoch bei näherer Betrachtung ihre Tücken, denn was heißt, daß ein Molekül noch nicht im Rekursivmolekül enthalten ist? Es stellt sich also die Frage, wann hier von einer Gleichheit (Identität) der Moleküle auszugehen ist. Diese Frage ist für die Rekursionssemantik entscheidend, denn wenn unendlich viele nicht-identische Moleküle eines Typs aus einer Datenbank abgeleitet werden können, ist die Terminierung der Rekursion nicht mehr garantiert.

  Wann sind zwei Moleküle gleich bzw. identisch? Für Atome ist diese Frage schon nicht einfach zu beantworten. Dies liegt unter anderem daran, daß man Atome sinnvollerweise so betrachtet, wie sie auch außerhalb von PRIMA gesehen werden können, nämlich als Projektionen einer Menge von Attributen der in der Datenbank vorhandenen Atome. Eine vorläufige Definition heißt: Atome, die nicht von derselben Rolle desselben Typs sind, sind weder gleich noch identisch, auch wenn ihre Attribute paarweise gleiche Werte haben. Zwei Atome sind von demselben Typ, wenn sie durch eine identische Projektion (das impliziert dieselbe Rolle) aus demselben im Schema definierten Atomtyp hervorgegangen sind. Eine Projektion ist identisch, wenn sie die gleiche Attributmenge umfaßt (die Attribute werden also hier erst einmal als ungeordnete Menge aufgefaßt; aus Anwendungssicht kann jedoch die Reihenfolge der Attribute verschieden sein). Zwei Atome sind gleich, wenn sie von demselben Typ sind und ihre Attributwerte paarweise gleich sind. Zwei Atome sind identisch, wenn sie gleich sind und aus demselben Atom hergeleitet wurden (diese Definitionen kann man problemlos auf den Fall der durch Anwendung des Operators *kartesisches Produkt* entstandenen Atome erweitern). Nun kann man die Gleichheit (Identität) von Molekülen über die Gleichheit (Identität) der in ihnen enthaltenen Atome und die Gleichheit der Molekülstruktur definieren (im Rahmen dieses Aufsatzes soll auf eine Formalisierung dieses Ansatzes verzichtet werden). Leider ergeben sich aber durch die Mächtigkeit des MAD-Modells Probleme mit diesem Ansatz: In der Projektionsklausel einer

SELECT-Anweisung kann ein Attributwert durch einen Ausdruck definiert werden, in den z.B. auch die Rekursionstiefe mit eingehen kann (vgl. Beispiel 3.1). Wenn eine solche Attributberechnung für die Wurzel des Komponentenmoleküls der Rekursion vorgenommen wird, entsteht eine unendliche Menge von paarweise ungleichen Atomen (und damit ungleichen Komponentenmolekülen) und die Rekursion würde nicht terminieren. Daher muß hier ein spezieller Gleichheitsbegriff eingeführt werden, nämlich der der *Ableitungsgleichheit*. Zwei Moleküle von demselben Typ sind ableitungsgleich, wenn ihre Wurzelatome ableitungsgleich sind. Zwei Atome sind ableitungsgleich, wenn sie nach Weglassen aller durch einen Ausdruck spezifizierten Attribute identisch sind (da im MAD-Modell eine beliebige Anfrageschachtelungstiefe erlaubt, ist diese Definition rekursiv über alle Anfrageschachtelungstiefen anzuwenden).

```
SELECT   ALL
FROM     M:= ( SELECT A(a:=REC_DEPTH)
               FROM   A) - B RECURSIVE B-M;
```

Beispiel 3.1: Eine Anfrage, die eine beliebige Anzahl ungleicher Atome erzeugen würde

Formulierung von Integritätsbedingungen

Neben der Garantie, daß die referentielle Integrität (im Sinne der korrekten Realisierung von Links) überwacht wird, bietet das MAD-Modell nur eine Möglichkeit, Integritätsbedingungen explizit zu beschreiben, nämlich die Spezifikation von Kardinalitätsrestriktionen für Wiederholungsgruppenattribute. Bei der Formulierung der Kardinalitätsrestriktionen kann allerdings nur ein Bereich angegeben werden, aus dem die Elementanzahlen stammen dürfen. Es sind jedoch Anwendungen vorstellbar, wo die erlaubten Anzahlen keinen zusammenhängenden Bereich bilden. Wenn z.B. ein Punkt entweder Eckpunkt in einem geschlossenen Polygonzug oder Mittelpunkt ist, so gehört er zu 2 oder 0 Kanten, nie jedoch zu nur einer. Eine entsprechende Erweiterung des Konzepts der Kardinalitätsrestiktionen wäre also wünschenswert. Ebenso wäre es sinnvoll, definieren zu können, daß ein bestimmtes Attribut nie einen Nullwert annehmen darf. Dies, ebenso wie eine Möglichkeit, Attribute durch Aspekte näher beschrieben zu können, wird auch in [Go91] gefordert.

Übergangsorientierte Integritätsbedingungen sind im MAD-Modell nicht vorgesehen. Bei der Beurteilung der Möglichkeiten zur Formulierung von Integritätsbedingungen, die das MAD-Modell anbietet, muß [De91] berücksichtigt werden. Dort wird dargelegt, daß eine zu große Mächtigkeit zur Formulierung nicht erwünscht ist. Dieser Aufsatz wird in Kapitel 8 nochmal erwähnt.

Nachdem hier nun das MAD-Modell isoliert betrachtet wurde, soll im folgenden untersucht werden, inwieweit sich das MAD-Modell für die bisher realisierten Anwendungen von PRIMA eignet. Dazu werden diese Anwendungen kurz beschrieben.

4. Bisher realisierte Anwendungen von PRIMA

Im folgenden beschreiben wir kurz einige der Anwendungen, die bisher schon auf der Basis von PRIMA entwickelt wurden. Sie werden uns in den folgenden Kapiteln als Referenzen dienen, wenn es gilt, Probleme oder auch besondere Eignung der in PRIMA gewählten Ansätze aufzuzeigen.

Obwohl die ursprüngliche Intention von PRIMA in der Datenbankunterstützung komplexer Anwendungen bestand, also nicht auf einen direkten Umgang mit dem Datenbanksystemkern selbst abzielte, haben wir eine interaktive MQL-Schnittstelle für PRIMA entwickelt, die es erlaubt, einzelne MQL-Anweisungen auszuführen. Die durch **SELECT**-Anfragen gewonnenen Molekülmengen werden am Bildschirm dargestellt und können mit einem Browsing-Mechanismus durchsucht werden. Aus Gründen der Darstellbarkeit werden dabei zu mehreren Molekülen gehörige Atome auch mehrfach dargestellt, es wird also eine Redundanz vorgespiegelt, die in den vom PRIMA-Kern gelieferten Daten nicht vorhanden ist. Änderung, Einfügung und Löschung von Molekülen (Molekülmengen) ist zum einen über MQL-Anweisungen möglich, in denen die jeweiligen einzufügenden oder zu ändernden Werte spezifiziert sind. Zum anderen kann man eine Molekülmenge graphisch eingeben, wenn man den entsprechenden Molekültyp vorher spezifiziert. Eine interaktive Manipulation der durch **SELECT** gewonnenen Daten mit anschließendem Einbringen in die Datenbank ist dagegen nicht immer möglich, weil in Anfragen neue Atomtypen dynamisch definiert werden können (z.B. über ein kartesisches Produkt), die in der Datenbank nicht vorhanden sind. Diese Problematik entspricht der der Änderung von Views in SQL.

Im Rahmen der Forschung über die Integration zeitlicher Aspekte in relationale Datenbanksysteme wurde im SFB 124 an der Universität Kaiserslautern die Anfragesprache TMQL entwickelt, die im Prinzip ein zeitorientiertes SQL-Derivat ist. Zur Implementierung von TMQL bot es sich an, die zeitbezogenen relationalen Anfragen auf nicht-zeitbezogene MQL-Anfragen abzubilden [KRS90]. In den meisten Fällen ist eine 1:1-Abbildung von TMQL-Anfragen auf MQL-Anfragen möglich. Diese wird von einem auf den PRIMA-Kern aufgesetzten System durchgeführt, das die Anfragen dann durch den PRIMA-Kern auswerten läßt und anschließend die Ergebnismolekülmenge in eine Menge von TMQL-Objekten transformiert. Ein zeitbehaftetes Tupel (TMQL-Objekt) wird dabei durch ein (lineares) Rekursivmolekül dargestellt. Beispiel 4.1 zeigt eines solche Transformation.

TMQL-Anfrage:

```
T_SELECT salary ONLY
T_FROM   Employee
T_WHERE  (department='D12') AT 1981/06/01;
```

Ergebnis der Transformation: MQL-Anfrage:

```
SELECT   Employee_tuple(LAST)(salary)
FROM     Employee_tuple
             REC_PATH Employee_tuple.past-Employee_tuple
             UNTIL  Employee_tuple(PREVIOUS).valid <= 1981/06/01
WHERE    (Employee_tuple(FIRST).future=EMPTY) AND
         (Employee_tuple(LAST).valid <= 1981/06/01) AND
         (Employee_tuple(LAST).department = 'D12');
```

Die TMQL-Anfrage liefert das Gehalt der Angestellten der Abteilung D12 mit dem Stand vom 1.6.1981. Bei der Transformation wird aus der zeitbehafteten Relation Employee *ein rekursiver Molekültyp bestehend aus Komponentenmolekülen mit dem einen Atomtyp* Employee*. Dieser enthält neben den in* Employee *enthaltenen Attributen weitere, die den Zeitbezug abbilden. So enthält das Attribut* Employee *die Gültigkeitszeit des Atoms. Nähere Einzelheiten zu dieser Transformation finden sich in [KRS90]*

Beispiel 4.1: Transformation einer TMQL-Anfrage nach MQL

Für den Bereich “VLSI-Entwurf” wurde eine Anwendung PRIMACHIP [HHPZ88] realisiert, die das Chip-Planning (Floorplan-Erstellung, Verdrahtung) PRIMA-gestützt durchführt. Ferner gibt es einen technischen Modellierer TECHMO, der auf PRIMA aufsetzt [HPS90]. Beiden Ansätzen ist gemeinsam, daß sie auf einem Objektpuffer arbeiten, der die von ihnen benötigten Daten lokal auf der Workstation hält, auf der die Anwendungen ablaufen. Die Objektpufferverwaltung [HHMM88, HS89] sorgt für den Datenaustausch mit der Datenbank. Nachdem Untersuchungen gezeigt hatten, daß eine Anwendungslokalität der Daten für die betrachteten Anwendungen unabdingbar ist, wurde eine allgemeine Objektpufferverwaltung implementiert, die allen entsprechenden Anwendungen die eigentliche Kommunikation mit dem Datenbanksystem abnehmen soll. Dazu spezifiziert der Anwendungsprogrammierer in Form von MQL-Anfragen die Daten, auf denen er arbeiten möchte. Diese werden aus der Datenbank extrahiert und in den Objektpuffer eingelagert. In diesem können die Daten mittels hierarchischer Cursor verarbeitet werden. Da bei den zu unterstützenden Anwendungen keine bevorzugte Verarbeitungsrichtung vorliegt [HHMM88, Hä87], können die Daten einmal als Molekülmenge angesehen werden (wie sie durch die MQL-Anfrage geliefert werden), zum anderen aber auch in einer Netzwerksicht ähnlich der des Atomnetzes der Datenbank. Hierbei können mehrere Anfragen ihre Ergebnisse in denselben oder in verschiedenen Objektpuffern ablegen. Am Ende der Verarbeitung sorgt die Objektpufferverwaltung nach einem PROPAGATE-Befehl dafür, daß die durch die Anwendung veränderten Daten in die Datenbank zurückgespeichert werden. Der Objektpufferansatz wird in Kapitel 6 noch weiter vertieft.

Für die Verwaltung versionierter Objekte wurde eine Basisversionsverwaltung [HK89,KS91] implementiert, die für jede Version eines Objektes einen Repräsentanten vorsieht. Die Versionsbeziehungen werden dann durch Links zwischen den Repräsentanten realisiert. Da komplexe Objekte (Moleküle) versioniert werden, bei denen jede Version mit Vorgängern und Nachfolgern in Basisobjekten (Atomen) überlappen kann, ist es wichtig zu wissen, welche Basisobjekte zu einer Version gehören. Daher sind zu allen diesen Basisobjekten Links von dem Repräsentanten aus eingerichtet.

An der Universität Kaiserslautern wird das Wissensbankverwaltungssystem KRISYS [Ma89] entwickelt, das auch auf PRIMA basiert. Der Ansatz zur Abbildung von Frame-Modellen auf das MAD-Modell wurde in [HMM87] näher beschrieben. Hier sei nur soviel dazu gesagt, daß es drei Atomtypen gibt, die zur Modellierung von Frames, Slots und Aspekten dienen.

5. Nutzung des MAD-Modells durch die Anwendungen

In diesem Kapitel werden wir diskutieren, wie die im vorigen Kapitel beschriebenen Anwendungen einzelne Konzepte des MAD-Modells nutzen, und wo die Unterstützung durch das MAD-Modell verbessert werden könnte.

Nutzung der Konzepte

Die vom MAD-Modell unterstützte Behandlung ***nicht-redundanter (überlappender) komplexer Objekte*** ist besonders wichtig, wenn solche Objekte typisch für die Anwendung sind, was zumindest für die Anwendungen PRIMACHIP und TECHMO zutrifft.

Die ***dynamische Definition der Struktur*** der zu verarbeitenden Komplexobjekte ermöglicht es, ein Netz von Basisobjekten unter verschiedenen Gesichtspunkten zu betrachten. Dies wird von den Anwen-

dungen ganz unterschiedlich genutzt. Die Anwendungen PRIMACHIP und TECHMO (und vermutlich alle auf einem Objektpuffer basierenden Anwendungen) halten immer dieselbe “Richtung” bei der Molekültypdefinition ein, so daß eine Flexibilität dieser Sicht hier keine Vorteile bringt. Die TMQL-Anwendung hingegen benutzt diese Eigenschaft des MAD-Modells in hohem Maße, um die Abbildung der zeitbehafteten Operationen auf MQL-Anfragen zu realisieren.

Das Konzept der ***Rekursion*** bei der Molekültypdefinition wird von allen Anwendungen genutzt. Dies zeigt, daß Rekursion auf dieser Ebene tatsächlich ein wichtiges Konzept darstellt.

Das Konzept der ***qualifizierten Projektion***, das aus konzeptioneller Sicht eine notwendige Funktionalität realisiert, wird in keiner der bisher erstellten Anwendungen genutzt. Es existieren jedoch Überlegungen, die qualifizierte Projektion in der Versionsverwaltung einzusetzen [KS91].

Die automatische ***Herstellung der referentiellen Integrität*** wird von allen Anwendungen genutzt. Sie nimmt dem Programmierer viele explizite Änderungsoperationen ab. Überhaupt scheint das Link-Konzept eine der meistbenutzten Eigenschaften des MAD-Modells zu sein. Allerdings wurde von Anwendungsseite oft der Wunsch geäußert, eine gewisse durch die Anwendung vorgegebene Ordnung auf den IDENTIFIER-Werten innerhalb eines REFERENCE-Attributes zu garantieren, um damit bestimmte anwendungsspezifische Zusammenhänge zu modellieren (z.B.: der erste Wert eines bestimmten REFERENCE-Attributes ist die linke obere Ecke eines Rechtecks). Dies widerspricht natürlich einer systemseitigen Garantie der referentiellen Integrität, wenn das System diese Ordnung nicht kennt oder nicht selbst berechnen kann. Eine Abhilfe, die in einigen Anwendungen ergriffen wurde, ist es, die referentielle Integrität doch immer “von Hand” herzustellen, und so eine systemseitige Änderung der REFERENCE-Attributwerte zu verhindern. Dies ist natürlich ein recht unsicherer Weg.

Die ***molekülmengenorientierten Manipulationsoperationen*** werden von den meisten Anwendungen nur für Ein-Atomtyp-Moleküle eingesetzt (vgl. Diskussion zu PROPAGATE in Kapitel 7).

KRISYS nutzt nach der momentan verfolgten Abbildungsstrategie nur Basiskonzepte des MAD-Modells aus, wie in [De91] dargelegt wird.

Bessere Unterstützung

Für einige Anwendungen wäre es wünschenswert, wenn REFERENCE-Attribute nicht immer nur auf genau einen Typ zeigen müßten (polymorphe REFERENCE-Attribute). So soll z.B. in der TMQL-Anwendung das Atom, das den jeweils aktuellen Zustand eines Objektes darstellt, zu einem anderen Atomtyp gehören als die Atome, die die Geschichte des Objekts repräsentieren. Daher muß die logische Beziehung “zeitlicher Nachfolger” durch zwei REFERENCE-Attribute realisiert werden, von den aber immer nur eines einen nicht-leeren Wert hat. (Die Zusicherung “nur eines der beiden REFERENCE-Attribute hat einen nicht-leeren Wert” ist übrigens im MAD-Modell auch nicht formulierbar). Wünschenswert wäre hier, daß die logische Beziehung auch nur durch ein REFERENCE-Attribut realisiert wird.

In der Modellierung des VLSI-Entwurfs als Entity-Relationship-Diagramm in [Si89] tritt ein ähnlicher Fall auf, in dem ein Entity genau zu einem von drei möglichen anderen Entities eine Beziehung hat. Ein damit ganz verwandtes Problemfeld ist die fehlende Möglichkeit, im MAD-Modell Generalisierungshierarchien zu spezifizieren (wie es auch in [Go91] gefordert wird). Allerdings legt [De91] dar, daß KRISYS eine evtl. in das MAD-Modell integrierte Generalisierung nicht nutzen würde. So tritt z.B. beim VLSI-Entwurf die Notwendigkeit auf, ein Objekt in den verschiedenen Entwurfsstufen zu speichern, die sich alle durch einen unterschiedlichen Informationsgehalt auszeichnen. Dies könnte sehr gut durch eine Generalisierungshierarchie modelliert werden. Auch hier müßten Referenzen auf mehr als einen Typ verweisen

können, denn als Bestandteil des Wertes eines REFERENCE-Attributs, das auf einen bestimmten Typ verweist, muß natürlich auch ein IDENTIFIER erlaubt sein, der zu einem Atom einer Spezialisierung dieses Typs gehört.

6. Modellabbildungsschicht und Transaktionskonzept

Wie bereits erwähnt, folgt PRIMA dem Konzept der Datenbankkernsystem-Architektur. Der Kern realisiert dabei an seiner Schnittstelle das MAD-Modell (vgl. Kapitel 2). Die Modellabbildungskomponente basiert auf einem Objektpuffer, in dem die vom Datenbankkernsystem gelieferten Komplexobjekte abgespeichert werden. Die Anwendung lädt diesen Objektpuffer durch MQL-Anfragen (*Checkout*, vgl. [LP83, KDG87]), greift dann auf die im Objektpuffer gespeicherten Objekte zu, manipuliert diese gegebenenfalls, und beauftragt dann die Objektpufferverwaltungskomponente (PROPAGATE), die Änderungen in die Datenbank einzubringen (Checkin). Diese Architektur soll durch Bild 6.1 verdeutlicht werden.

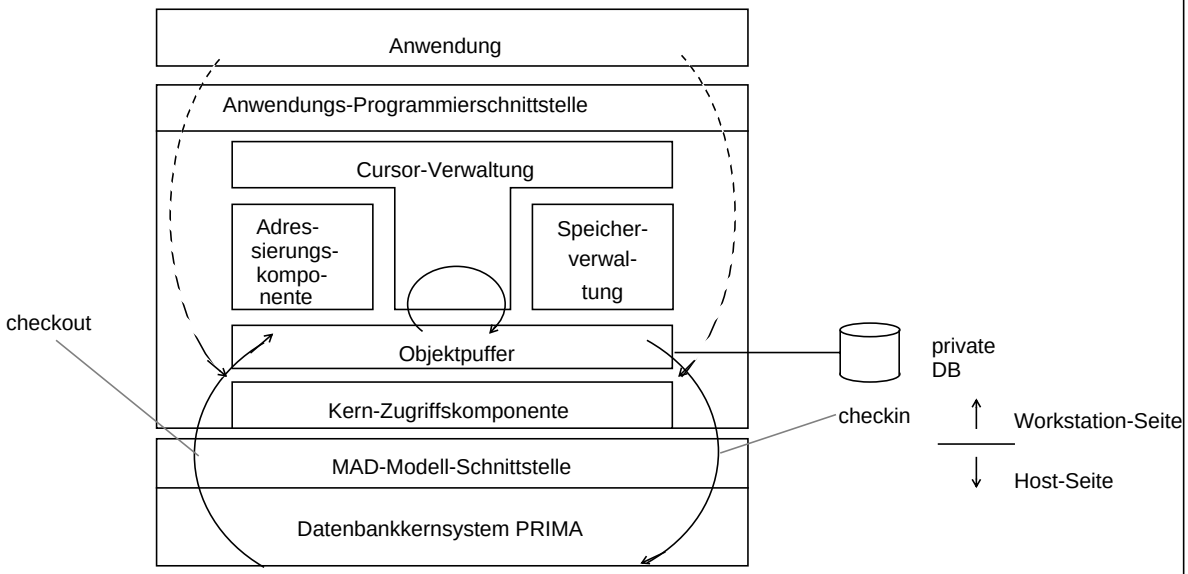


Bild 6.1: Ein Anwendung setzt auf dem Objektpuffer auf, der wiederum mit PRIMA arbeitet [HPS89]

Um die im Objektpuffer gespeicherten Objekte verarbeiten zu können, bietet die Objektpufferverwaltung [HS89] die Möglichkeit, hierarchische Cursor auf den Daten zu definieren. Dadurch können Anwendungsprogramme einen navigierenden Zugriff auf die Objekte durchführen. Dabei können die Daten einmal als Molekülmenge angesehen werden (wie sie durch die MQL-Anfrage geliefert werden), zum anderen aber auch in einer Netzwerksicht ähnlich der des Atomnetzes der Datenbank. Es ist möglich, die Ergebnisse mehrerer Anfragen in einem Objektpuffer zu speichern; die Benutzung mehrerer Objektpuffer nebeneinander ist aber auch erlaubt. Diese ist allerdings insofern problematisch, als die Objektpufferverwaltung sich der Abhängigkeiten zwischen den Daten in den einzelnen Objektpuffern bewußt sein muß. Sobald diese Daten überlappen (was schon der Fall ist, wenn Daten in einem Objektpuffer Referenzen auf Daten in einem anderen enthalten), müssen diese Daten durch die Objektpufferverwaltung

selbst oder durch das Anwendungsprogramm konsistent gehalten werden, was im allgemeinen Fall nicht möglich zu sein scheint.

Das Verarbeitungsmodell, das mit der Einführung eines Objektpuffers unterstellt wird (einige Checkout-Operationen am Beginn der Verarbeitung, lange Dauer der Verarbeitung, ein Checkin zum Abschluß) stellt natürlich auch einige Anforderungen an ein geeignetes Transaktionskonzept [HHMM88]. So müssen Daten, die durch Checkout ausgelagert wurden, vor der Veränderung durch andere Anwendungen geschützt werden, damit die Daten, die im Objektpuffer enthalten sind, nicht inkonsistent werden. Unser Transaktionskonzept sieht die Klammerung eines Verarbeitungsschrittes (Checkout(s) bis Checkin) in einer Transaktion (top-level-Transaktion) vor. Jedes Checkin und Checkout wird in einer Sub-Transaktion dieser Transaktion abgewickelt. Bei Checkout werden Sperren erworben, die nach Beendigung der Checkout-Transaktion an die top-level-Transaktion vererbt werden und somit der Checkin-Transaktion wieder zur Verfügung stehen.

Dieses Konzept muß natürlich auch die Verteilung der Arbeit auf mehrere Rechner (Workstation und Server-Rechner) berücksichtigen. Dies betrifft zum einen Logging und Recovery (auch die lokal auf der Workstation ausgeführte Arbeit muß gesichert werden) - hier wurden entsprechende Check- und Savepoints eingeführt. Zum anderen soll die Arbeit auf der Workstation durch einen Ausfall des Server-Rechners nicht behindert werden (solange kein Datenaustausch mit der Datenbank stattfinden soll). Insbesondere soll natürlich ein Einbringen der lokalen Änderungen auch nach einem Ausfall und Wiederanlaufen des Server-Rechners möglich sein. Dazu müssen die von der Anwendung erworbenen Sperren einen solchen Systemausfall überleben, also persistent sein.

Ganz zwangsläufig mußten also Änderungen und Erweiterungen am klassischen Transaktionskonzept vorgenommen werden, um die Verarbeitungscharakteristika einer auf dem Objektpuffer- und Datenbankkernsystem-Ansatz basierenden Anwendungen geeignet zu unterstützen. Im nächsten Kapitel soll untersucht werden, in wieweit dies auch für das Datenmodell (das MAD-Modell) zutrifft, bzw. wie gut das MAD-Modell diese Verarbeitungscharakteristika unterstützt.

7. Zusammenspiel von MAD-Modell, Objektpufferansatz und geschachtelten Transaktionen

Wir betrachten das Zusammenwirken des MAD-Modells zunächst mit der Objektpufferverwaltung, also die Unterstützung für das Laden und Propagieren des Objektpuffers, die das MAD-Modell bietet. Anschließend diskutieren wir das Zusammenspiel des MAD-Modells mit dem in PRIMA integrierten Transaktionskonzept.

Unterstützung des Objektpufferansatzes durch das MAD-Modell

Der Objektpuffer wird dadurch geladen, daß eine MQL-Anfrage gestellt wird, und die von dieser gelieferten Daten in den Objektpuffer eingelagert werden. Nun können darauf hierarchische Cursor definiert werden. Solange man dabei die Molekülsicht hat (das impliziert, daß der hierarchische Cursor eine Untermenge der gerichteten Links benutzt, die in den Daten enthalten sind), ist das völlig unproblematisch, denn es wird genau auf der Struktur gearbeitet, die vom MAD-Modell und MQL geliefert wird. Nun will man im Objektpuffer aber auch die Netzwerksicht auf die Daten haben. Gewünscht ist es also, beliebig

viele hierarchische Cursor in beliebigen Richtungen über dem Atomnetz definieren zu können, das man durch Ignorieren der Richtung der in der Molekülmenge enthaltenen Links erhält. Die Links in der Ergebnismenge sind aber gerichtet, das bedeutet, daß in den Strukturinformationen zu den Molekülen keine Darstellung der Gegenrichtungen enthalten ist. Um also eine Netzsicht zu erhalten, muß die Objektpufferverwaltung die Werte der REFERENCE-Attribute interpretieren. Diese können jedoch auch Referenzen enthalten, die nicht auf Daten im Objektpuffer verweisen, nämlich dann, wenn sie auf Objekte zeigen, die durch die Anfrage ausgeblendet wurden, mit der der Objektpuffer gefüllt wurde. Die Objektpufferverwaltung muß dies erkennen und entsprechend behandeln.

Dieser Ansatz der Interpretation von REFERENCE-Attributen ist nicht mehr praktikabel, wenn durch die Anfrage neue Atomtypen entstehen (z.B. durch Anwendung des Operators *kartesisches Produkt* oder durch Attributberechnung aus einem Ausdruck), weil die dann entstehenden neuen IDENTIFIER-Werte natürlich nicht in den REFERENCE-Attributen enthalten sein können, die in der Datenbank gespeichert waren. Nach der Definition des MAD-Modells "erben" die neuen Atomtypen aber die Links ihrer Ursprungsatomtypen. Auch eine qualifizierte Projektion geht bei diesem Ansatz verloren. So ist in Beispiel 2.5 beim Durchlaufen der Richtung *C-B-A* nicht mehr erkennbar, daß zu *c3* nur in einem der Moleküle ein Link von *b3* aus führt, denn das REFERENCE-Attribut, das in der Datenbank gespeichert war und nun ein Attribut der *C*-Atome ist, enthält die IDENTIFIER-Werte aller ursprünglich (vor der Auswertung der qualifizierten Projektion) von dem *C*-Atom referenzierten *B*-Atome. Molekülspezifische Strukturunterschiede gehen also verloren. Dieselbe Argumentation gilt analog für Rekursivmoleküle. Somit lassen sich zum Füllen eines Objektpuffers mit "Netzsicht" nur Anfragen sinnvoll einsetzen, die auf diese "höheren" Konstrukte im MAD-Modell verzichten und nur Mengen einfacher Moleküle liefern.

Wenn mehrere Anfragen dazu benutzt werden, einen Objektpuffer zu füllen, wird die Lage noch schwieriger, falls sich die Anfragen von den vorkommenden Atomtypen her überlappen, weil sich einzelne Atome dann in der Netzsicht nicht mehr einer bestimmten Anfrage zuordnen lassen (Beispiel 7.1). Auch in der Molekülsicht entstehen Probleme dadurch, daß Atome, die in mehreren Ergebnismengen vorkommen, mehrfach an den Objektpuffer übermittelt werden, und zwar sowohl die Daten, als auch die entsprechenden Verweise in der Struktursicht.

```
SELECT  ALL                 SELECT  ALL
FROM    A-B-C               FROM    B
WHERE   A.groesse=7         WHERE   B.name="Erwin"
```

Anfrage 1 *Anfrage 2*

Wie kann die Objektpufferverwaltung nach dem Laden noch unterschieden, ob ein B-Atom durch Anfrage1, Anfrage 2 oder beide in den Objektpuffer eingelagert wurde?

Beispiel 7.1: Einlagerung überlappender Daten durch mehrere Anfragen in den Objektpuffer

Zum Einbringen der durch das Anwendungsprogramm geänderten Daten in die Datenbank stehen drei MQL-Befehle zur Verfügung: **INSERT**, **DELETE** und **UPDATE**, die alle auf Molekülmengen arbeiten. Nun sind aber typischerweise die Änderungen, die die Anwendung an den Daten vorgenommen hat, nicht auf ein reines Einfügen, Ändern oder Löschen abzubilden, sondern sind eine Kombination dieser drei Operationen. Um also die Daten in die Datenbank einzubringen, muß die Objektpufferverwaltung genau buchhalten, welche Atome geändert, eingefügt, oder gelöscht wurden. Da es vorkommen kann, daß Teile eines Moleküls gelöscht, neue Teile eingefügt, und andere modifiziert wurden, können die Änderungen nicht als homogene Manipulationsoperationen (d.h. nur entweder **INSERT**, **DELETE** oder **UPDATE**)

auf den zum Laden des Objektpuffers verwendeten Molekültypen beschrieben werden. Vielmehr müssen hier Ein-Atomtyp-Molekültypen zur Anwendung kommen. Damit besteht aber ein PROPAGATE nur noch aus einer Menge von atommengenorientierten MQL-Befehlen, so daß die im MAD-Modell angebotene Operationsmächtigkeit nicht genutzt wird.

Bereits in [HS89] wurde darauf hingewiesen, daß die Nutzung der Operationen **INSERT**, **DELETE** und **UPDATE** mit expliziter Wertangabe zusätzlich zu PROPAGATE dem Objektpufferansatz insofern widerspricht, als zum einen das Prinzip des einen Checkin am Ende der top-level-Transaktion verletzt wird und zum anderen der Inhalt des Objektpuffers invalidiert werden kann (weil er nicht mehr mit dem Inhalt der Datenbank übereinstimmt).

Berücksichtigung des Transaktionskonzeptes durch das MAD-Modell

Nach dem Transaktionskonzept werden Sperren lange, d.h., über MQL-Operationen hinweg gehalten. Wenn nun eine Anwendung Daten ändern möchte, ohne Gefahr zu laufen, daß diese Änderung zum Checkin-Zeitpunkt scheitert, muß sie diese Daten mit einer X-Sperre belegen. Da die Sperren im Datenbanksystemkern verwaltet werden (für alle Anwendungen), muß diese X-Sperre beim Datenbanksystemkern angefordert werden. Dazu gibt es nun keine andere Möglichkeit, als beim Lesen der Daten bereits zu spezifizieren, daß diese Daten (möglicherweise) geändert werden sollen: Durch Ergänzung von **SELECT** um **FOR UPDATE** werden für alle Atome, die in der Ergebnismolekülmenge vorhanden sind, X-Sperren angefordert. Hierbei besteht die Gefahr, daß viel zu viele X-Sperren angefordert werden, da die Anwendung ja nicht unbedingt alle gelesenen Moleküle in allen ihren Teilen ändern möchte. Es gibt jedoch innerhalb einer MQL-Anfrage keine Möglichkeit, nur bestimme Teile mit einer X-Sperre zu belegen. Weiterhin gibt es keine Möglichkeit, den Sperrmodus von Daten zu ändern, ohne sie neu zu lesen. Andererseits ist eine Trennung der Sperranforderung von MQL-Anweisungen auch problematisch, denn dann müßte die Anwendung zunächst Sperren anfordern, um dann die Molekülmengen zu lesen. Andererseits weiß sie erst nach dem Lesen der Molekülmenge, welche Daten sie überhaupt berührt.

Weitere Probleme hinsichtlich der Sperren seien nur kurz erwähnt: Es scheint wegen der dynamischen Definition von Molekültypen in Anfragen nicht möglich zu sein, auf Molekülebene zu sperren. Das bedeutet, daß für alle in einem Molekül enthaltenen Atome einzelne Sperren angefordert werden müssen. Die automatische Herstellung der referentiellen Integrität bewirkt, daß neben einem Atom, dessen REFERENCE-Attribut vom Anwender geändert wurde, auch die REFERENCE-Attribute in allen nun zusätzlich oder nicht mehr referenzierten Atomen geändert werden müssen. Wenn das kleinste Sperrgranulat ein Atom ist, dann verhindert eine solche Änderung des REFERENCE-Attributes eine gleichzeitige Änderung eines normalen Attributwertes in diesem Atom, die ansonsten problemlos erlaubt werden könnte. Es sollte daher überlegt werden, ob ein Sperren auf feineren Granulaten (Attributen) oder anderen Granulaten (Links und “Kernatome”) sinnvoll sein kann.

8. Zusammenfassung

Im Rahmen dieses Aufsatzes sollte eine erste Beurteilung des Molekül-Atom-Datenmodells, wie es in PRIMA implementiert ist, versucht werden. Ziel war es dabei, in einer Art Zwischenbilanz festzustellen, welche Konzepte im MAD-Modell sich als besonders hilfreich erwiesen haben, und wo kleinere oder größere Änderungen angebracht wären. Insbesondere sollte eine erste Bewertung der Unterstützung des

Objektpufferansatzes erfolgen, da dieser von einer Vielzahl künftiger Anwendungen adaptiert werden wird. Zusammenfassend läßt sich feststellen, daß mit dem MAD-Modell ein mächtiges und komplexes Datenmodell geschaffen wurde, dessen größte Stärke in der dynamischen Definition und Verarbeitung komplexer Objekte liegt. Am Modell selbst gibt es nur kleinere Verbesserungsvorschläge.

Wendet man den in [HS91] für objektorientierte Datenmodelle aufgestellten Maßstab auf das MAD-Modell und seine Anfragesprache MQL an, so zeigt sich, daß die dort formulierten Anforderungen weitgehend erfüllt sind:

- Deskriptivität: MQL ist eine deskriptive mengenorientierte Anfragesprache.
- Abgeschlossenheit und Adäquatheit: MQL-Anfragen liefern Molekültypen (Mengen von Molekülen desselben Typs), so daß die Abgeschlossenheit gegeben ist. Die Adäquatheit besagt, daß die Ergebnisse von Anfragen die gesamte Modellierungsmächtigkeit des Modells ausnutzen. Auch dies ist in MQL der Fall.
- Orthogonalität ist im wesentlichen gegeben. Insbesondere ist die Anforderung erfüllt, daß überall dort, wo ein Molekültyp stehen darf, auch eine Anfrage erlaubt ist. Andererseits ist es nicht möglich, Anfrageergebnisse direkt in der Datenbank abzulegen. Hier sollte stärker formalisiert werden, unter welchen Voraussetzungen und mit welchen Sprachmitteln Anfrageergebnisse wieder in die Datenbank eingebracht werden können.
- Andere Kriterien aus [HS91] lassen sich teilweise nicht anwenden oder schwer beurteilen (z.B. entfällt die Betrachtung von Vererbungshierarchien, da diese vom MAD-Modell nicht unterstützt werden).

Betrachtet man nun die Nutzung der Möglichkeiten, die das MAD-Modell bietet, durch die inzwischen realisierten Anwendungen, so kann man zwei Klassen unterscheiden:

Diejenigen Anwendungen, die ohne Nutzung eines Objektpuffers direkt auf dem MAD-Modell aufgesetzt sind, nutzen die mächtigen Operationen des Datenmodells ebenso wie die Dynamik der Molekültypdefiniton intensiv.

Die Anwendungen jedoch, die auf einem Objektpuffer basieren, nutzen nur einfache Grundkonzepte des MAD-Modells, insbesondere das Link-Konzept. Auf die Dynamik in der Objekttypdefinition wird weitgehend verzichtet. Aus der Sicht dieser Anwendungen werden “komplexe” Operationen (wie z.B. *qualifizierte Projektion*) im Datenmodell nicht benötigt. Bei der Untersuchung der Möglichkeit Integritätsbedingungen zu formulieren, kommt [De91] zu einem analogen Schluß.

Ein großes Problem liegt darin, daß im Objektpuffer wieder eine netzwerkartige Sicht auf die Daten gewünscht wird, wie sie in der Datenbank vorliegt, in den Ergebnissen von MQL-Anfragen jedoch nicht. Man kann durchaus die Auffassung vertreten, daß die Objektpufferverwaltung mit ihrer Schnittstelle selbst wieder eine (von MQL verschiedene) Realisierung des MAD-Modells ist, zumindest was die Dynamik der Objektdefinition und den Aufbau auf Netzwerke angeht. Es stellt sich also die Frage, ob das MAD-Modell die richtige Schnittstelle für das Datenbankkernsystem ist, wenn in der Workstation ein Objektpuffer zum Einsatz kommt. Insbesondere wäre es wünschenswert, Atomnetze als solche zwischen Datenbank und Objektpuffer austauschen zu können. Die Auswahl über einen Molekülmechanismus scheint jedoch sinnvoll zu sein. Auswahl- und Verarbeitungseinheit sollten also unterschieden werden. Insbesondere muß untersucht werden, ob nicht folgende Erweiterungen zu einer adäquateren Unterstützung des Objektpufferansatzes führen:

- Das Zulassen mehrerer Wurzeln bei der Molekültypdefinition. Die Auswahl geschieht also weiterhin durch Navigation über einem gerichteten Graphen. Diese Navigation startet jedoch bei jeder Wurzel

des Graphen. Von mehreren Wurzeln erreichbare Molekülteile werden nicht-redundant in die Ergebnismenge aufgenommen und die so entstehende Netzstruktur wird repräsentiert.

- Das Erlauben von Nicht-Kohärenz. Als Erweiterung der ersten Punktes kann erlaubt werden, daß der durch die verschiedenen Wurzel der Molekültypdefinition beschriebene Ausschnitt des Datenbankschemas nicht zusammenhängend sein muß. Ferner könnte auf die Forderung verzichtet werden, daß das Ergebnis einer Anfrage eine Menge *in sich zusammenhängender* Moleküle sein muß.
- Als Konsequenz aus dem vorigen Punkt muß dann der Informationsgehalt einer Anfrageantwort neu definiert werden. Die Antwort kann vermutlich keine Informationen mehr über die zur Auswertung der Anfrage verwendeten Moleküle liefern. Andererseits können die Atome der Ergebnismenge dann auch mit ungerichteten Links (statt wie bisher mit gerichteten analog zur Molekültypdefinition) vernetzt sein.

In diesem Zusammenhang muß auch darüber nachgedacht werden, wie komplex die Operationen in der Workstation bei Existenz eines Objektpuffers im Vergleich zu denen auf der Workstation noch sein können. Die Frage ist also, ob in einer Workstation/Server-Architektur nicht die “komplexeren” Operationen auf der Workstation ausgeführt werden (vgl. [De91]).

Es zeigt sich weiterhin, daß einigen Anwendungen die Modellierungsmächtigkeit im MAD-Modell nicht genügt. So werden flexiblere (polymorphe) Referenzen und die Möglichkeit zum Aufbau einer Generalisierungshierarchie über Atomtypen und Linktypen gefordert.

Schließlich bestehen noch größere Anbindungsprobleme des MAD-Modells an das zur Anwendung kommende Transaktionskonzept. Diese Probleme entstehen anscheinend aus einem nicht gut angepaßten Sperrkonzept. Andererseits ist es schwierig, Anwendungen auf “höheren” Objekten zu synchronisieren, solange Objekte in jeder Anfrage dynamisch neu definiert werden können und außerdem überlappend sein können.

Eine Reihe anderer Systemaspekte, die sich aus dem Einsatz eines Objektpuffers ergeben, wurden bisher nicht erwähnt, so z.B. die Frage, ob in diesem Falle noch ein Systempuffer benötigt wird. Diese Fragen sprengen jedoch den Umfang dieses Beitrags.

Danksagung

Ich danke meinen Kollegen Christoph Hübel, Wolfgang Käfer und Dr. Bernhard Mitschang sowie Prof. Dr. Härder, die mir wertvolle Hinweise zur Verbesserung früherer Versionen dieses Aufsatzes gegeben haben.

Referenzen

Ca86 Carey, M.J., et al: The Architecture of the EXODUS Extensible DBMS, in: Proc. of the Int. Workshop on Object-Oriented Database Systems, Asilomar, 1986, S. 52-65.

Da86 Dadam, P., et al: A DBMS Prototype to Support Extended NF^2 Relations: An Integrated View on Flat Tables and Hierarchies, in: Proc. of the ACM SIGMOD Conf., Washington, 1986, S. 356-367.

De91 Deßloch, St.: Handling Integrity in a KBMS Architecture for Workstation/Server Environments, in: Proc. GI-Fachtagung “Datenbanksysteme in Büro, Technik und Wissenschaft”, IFB 270, Kaiserslautern, 1991, S. 89-108.

Go91 Gorchs, Th.: Adäquatheit im FORMAD-Modell als eine Voraussetzung zur Datenbankunterstützung für wissensbasierte Systeme, in: Proc. GI-Fachtagung “Datenbanksysteme in Büro, Technik und Wissenschaft”, IFB 270, Kaiserslautern, 1991, S. 126-139.

Hä87 Härder, T.: Database Support for Engineering Applications, in: Proc. Int. Workshop on Information in Manufacturing Automation, Dresden, 1987.

HHMM88 Härder, T., Hübel, C., Meyer-Wegener, K., Mitschang, B.: Processing and Transaction Concepts for Cooperation of Engineering Workstations and a Database Server, in: Data and Knowledge Engineering 3 (1988), S. 87-107, 1988.

HHPZ88 Härder, T., Hübel, C., Pahle, H., Zimmermann, G.: Ansätze zur DB-Unterstützung für den VLSI-Entwurf, Sonderforschungsbereich 124, Bericht 31/88, Universität Kaiserslautern, 1988.

HK89 Hübel, C., Käfer, W.: Modellierung und Handhabung versionierter Objekte, Sonderforschungsbereich 124, Bericht 26/89, Universität Kaiserslautern, 1989

HMM87 Härder, T., Mattos, N., Mitschang, B.: Abbildung von Frames auf neuere Datenmodelle, in: Proc. 11th German Workshop on Artifical Intelligence, Geseke, IFB 152, 1987.

HMMS87 Härder, T., Meyer-Wegener, K., Mitschang, B., Sikeler, A.: PRIMA - A DBMS Prototype Supporting Engineering Applications, in: Proc. of the 13th VLDB, Brighton, 1987, S. 433-442.

HPS89 Hübel, C., Paul, R., Sutter, B.: Technische Modellierung und DB-gestützte Datenhaltung - ein Ansatz für ein durchgängiges, integriertes Produktmodell, Zentrum rechnergestützter Ingenieursysteme, Bericht 6/89, Universität Kaiserslautern, 1989.

HR85 Härder, T., Reuter, A.: Architektur von Datenbanksystemen für Non-Standard-Anwendungen, in: Proc. GI-Fachtagung “Datenbanksysteme in Büro, Technik und Wissenschaft”, IFB94, Springer Verlag, 1985, S. 253-286.

HS89 Hübel, C., Sutter, B.: Verarbeitung komplexer Objekte in Ingenieuranwendungen, Zentrum rechnergestützter Ingenieursysteme, Bericht 5/89, Universität Kaiserslautern, 1989.

HS91 Heuer, A., Scholl, M.H.: Principles of Object-Oriented Query Languages, in: Proc. GI-Fachtagung “Datenbanksysteme in Büro, Technik und Wissenschaft”, IFB 270, Kaiserslautern, 1991, S. 178-197.

KDG87 Küspert, K., Dadam, P., Günauer, J.: Cooperative Object Buffer Management in the Advanced Information Management Prototype, in: Proc. of the 13th VLDB, Brighton, 1987, S. 483-492.

KRS90 Käfer, W., Ritter, N., Schöning, H.: Support for Temporal Data by Complex Objects, Proc. Int. Conf. on Very Large Data Bases VLDB 90, Brisbane, Australia, 1990, S. 24-35.

KS91 Käfer, W., Schöning, H.: Mapping a Version Model to a Complex-Object Data Model, submitted for publication.

Li87 Linnemann, V. et al.: Design and Implementation of an Extensible Database Management System Supporting User Defined Types and Functions, IBM Germany, Heidelberg Scientific Center, Report No. TR 87.12.011, 1987.

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

LP83 Lorie, R.A., Plouffe, W.: Complex Objects and Their Use in Design Transactions, in: Proc. of the Annual Meeting Database Week “Engineering Design Applications”, San Jose, 1983, S. 115-121.

Ma89 Mattos, N.: An Approach to Knowledge Base Management - Requirements, Knowledge Representation, and Design Issues, Dissertation, Universität Kaiserslautern, 1989.

Mi88 Mitschang, B.: Ein Molekül-Atom-Datenmodell für Non-Standard-Anwendungen, IFB 185, Springer Verlag, 1988.

Mo81 Moss, J.E.B.: Nested Transactions - An Approach to Reliable Distributed Computing, Ph.D. Thesis, Dep. of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 1981.

Pa87 Paul, H.B., et al: Architecture and Implementation of the Darmstadt Database Kernel System, in: Proc. ACM SIGMOD Conf., San Francisco, 1987, S. 196-207.

Schö89 Schöning, H.: Integrating Complex Objects and Recursion, in: Proc. 1st Int. Conf. on Deductive and Object-Oriented Databases DOOD 89, Kyoto, Japan, 1989, S. 535-554.

Schw86 Schwarz, P., et al: Extensibility in the STARBURST Database System, in: Proc. of the Int. Workshop on Object-Oriented Database Systems, Asilomar, 1986, S. 85-92.

Si89 Siepmann, E.: Eine objektorientierte Datenbankmodellierung für den VLSI-Entwurf, in: Proc. GI-Fachtagung Datenbanksysteme in Büro, Technik und Wissenschaft, BTW 89, Zürich, Schweiz, 1989, S. 289-294.

SS86 Schek, H.J., Scholl, M.H.: The Relational Model with Relation-Valued Attributes, in: Information Systems, Vol. 2, No. 2, 1986, S. 137-147.

St86 Stonebraker, M.: Inclusion of New Types in Relational Data Base Systems, in: Proc. 2. Int. Conference on Data Engineering, Los Angeles, Ca., 1986, pp. 262-269.