Wissenschaftliche Zeitschrift der Technischen Universität „Otto von Guericke" Magdeburg 32 (1988) Heft 5
Sektion Informatik · Direktor: Prof. Dr. rer. nat. F. Stuchlik
DK 518.6:681:3 Eingang: 2. 2. 1988

DB-Schnittstellen für arbeitsplatzorientierte Ingenieuranwendungen

C. Hübel[1]), *B. Mitschang*[1])

Die Schnittstellen gegenwärtig verfügbarer Datenbanksysteme erweisen sich als ungeeignet für den Bereich der Ingenieuranwendungen. Dies gilt im besonderen, da bei Ingenieuranwendungen eine deutliche Arbeitsplatzorientierung vorherrscht und sie damit häufig auf einer speziellen Mehrrechnerarchitektur aufbauen. Ausgehend von einem Schichtenmodell für sog. Non-Standard-Datenbanksysteme (NDBS) schlagen wir daher eine spezielle Systemarchitektur mit einer entsprechend ausgelegten Datenmodell- und Anwendungsmodellschnittstelle für arbeitsplatzorientierte Ingenieuranwendungen vor.

1. Einleitung

Zu den wesentlichen Merkmalen rechnergestützter Ingenieuranwendungen gehören

- die Abwicklung von Entwurfsprozessen,
- die Kooperation von Benutzern bei Teil- bzw. Einzelentwürfen sowie
- die Inanspruchnahme von allgemein zugänglichen Dienstleistungen eines Rechnernetzes.

Hierbei werden häufig Arbeitsplatzrechner (Workstation) eingesetzt. Sie sind auf die speziellen Anwendungs- und Benutzerbedürfnisse zugeschnitten (graphische Ein-/Ausgabe, Window-Technik) und ermöglichen eine weitgehend lokale und deshalb auch effiziente Bearbeitung der Operationen eines einzelnen Benutzers. Extern angeforderte Dienstleistungen beschränken sich im wesentlichen auf Aufgaben der Datenhaltung, Datensicherung, Integritäts- und Zugriffskontrolle (Organisation des Mehrbenutzerbetriebs). Diese Verwaltung des gemeinsam genutzten Datenbestandes wird durch ein Datenbanksystem (DBS) auf dem Wirtsrechner (Host) bewerkstelligt. Der praktische Einsatz eines DB-basierten Systems von Arbeitsplatzrechnern wirft allerdings eine Reihe von Problemstellungen auf. Die konventionell angebotenen DB-Schnittstellen stellen für den Ingenieurbereich nur geringe Anwendungsunterstützung bereit. Viele der zu beobachtenden Mängel haben ihre Wurzel in einer inadäquaten Datenmodellierung und einer sehr ineffizienten, da nicht lokalitätserhaltenden, Datenverarbeitung. Zur Lösung dieser Problematik wird im folgenden eine zweigeteilte Systemarchitektur vorgestellt, die sich dem Workstation-Host-Konzept anpaßt. Weiterhin werden wesentlichen Schnittstellen dieses Architekturvorschlags erläutert und die zugrundeliegenden Realisierungskonzepte aufgezeigt.

2. Architektur DB-basierter Ingenieuranwendungen

Vielerorts [HR85, LD85, PSSWD87] wird die sog. NDBS-Kern-Architektur als aussichtsreiche Systemarchitektur für sog. Non-Standard-DBS, also auch für DBS in Ingenieuranwendungen, angesehen. Hinter diesem Architekturkonzept verbirgt sich die Idee der Zweiteilung der NDBS-Architektur in einen anwendungsunabhängigen NBDS-Kern und in eine anwendungsorientierte Systemebene, Modellabbildung (MA) genannt (Bild1). Die Vorteile dieses Ansatzes liegen vor allem darin, daß einerseits durch die MA eine anwendungsbezogene Schnittstelle (Anwendungsmodellschnittstelle) mit den von der jeweiligen Anwendung benötigten Objekten und Operationen bereitgestellt werden kann und andererseits alle geeigneten, allgemein verwendbaren Darstellungs- und Zugriffstechniken sich im NDBS-Kern redundanzfrei vereinigen und effizient implementieren lassen. Der Kern realisiert damit ein allgemeines Datenmodell, auf dem die mit mehr Semantik ausgestatteten speziellen Modelle der verschiedenen Anwendungsklassen aufbauen. Als ein wesentliches Kriterium für die Tauglichkeit des Datenmodells eines solchen NDBS-Kerns wird oftmals der Grad der durch das Modell erreichten Objektunterstützung genannt. In [BB84, Mi85, HHLM87] sind die folgenden Aspekte, durch detaillierte Analysen untermauert, als zentrale Anforderungen genannt:

- direkte und symmetrische Modellierung und Verarbeitung von rekursiven und netzwerkartigen Strukturen,
- dynamische Definition der gewünschten Verarbeitungsgranulate (meistens strukturierte, heterogene Satzmengen) sowie
- adäquate Objektverarbeitung durch Funktionen zum vertikalen und horizontalen Zugriff (d. h. bezogen auf eine Objektausprägung bzw. eine homogene Objektmenge).

Die Aufgabe der Systemkomponente MA besteht nun darin, ein konkretes Anwendungsmodell auf das Datenmodell des Kerns abzubilden. Eine Erweiterung von bestehenden bzw. ein Übergang zu neuen Anwendungsklassen ist daher durch Erweiterung bzw. Austausch der MA recht einfach zu bewerkstelligen.
In Bild 1 ist das Konzept der NDBS-Kern-Architektur aufgezeigt. Dabei wird die Zuordnung des NDBS-Kerns zur Host-Seite und der MA zusammen mit der darauf aufbauenden Ingenieuranwendung zur Workstation-Seite deutlich. Aufgrund der Homogenität zwischen Hardware- und Software-Architektur sind die durch rechnerübergreifende Kommunikation zu erwartenden „Reibungsverluste" relativ gering. Zudem verspricht die Vermehrung der arbeitsplatzbezogenen Rechnerleistung einen echten Performance-Gewinn gegenüber dem Ein-Rechner-Fall. Die verarbeitungsinhärente Lokalität bzgl. der Datenreferenzen kann auf Workstation-Seite erhalten und ausgenutzt werden. Die verteilte Verarbeitung zwischen Workstation und Host erlaubt darüber hinaus, durch angepaßte Konzepte zur Fehlermaskierung bei System- bzw. Rechnerausfall (Logging/Recovery) sowie durch adäquate Verfahren zur Synchronisation, eine wesentliche Vergrößerung der Anwendungsautonomie. Die hiermit zusammenhängenden Fragestellungen sollen an dieser Stelle allerdings nicht weiter vertieft werden. Das NDBS insgesamt stellt somit ein anwendungsbezogenes Modellierungswerkzeug zur Verfü-

[1]) Universität Kaiserslautern.

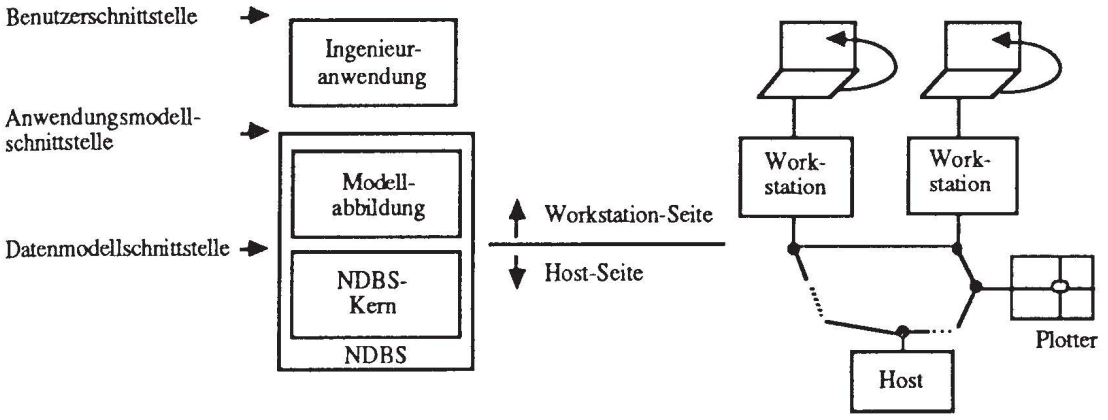


Bild 1
Systemarchitektur NDBS-basierter Ingenieuranwendungen

gung und verspricht gleichzeitig den hohen Leistungsanforderungen, speziell im Ingenieurbereich, gerecht zu werden. Im folgenden Kapitel wird ein Datenmodell aufgezeigt, das den oben genannten Anforderungen genügt.

3. Das Molekül-Atom-Datenmodell

Das Molekül-Atom-Datenmodell (MAD-Modell, [Mi87, Mi88]) erlaubt sowohl eine direkte und symmetrische Modellierung als auch die Verarbeitung von netzwerkartigen und rekursiven Strukturen. Im Gegensatz hierzu verfolgen die von *Schek* [SS86] und *Lorie* [LK84] vorgeschlagenen Modelle nur die Integration von hierarchischen Strukturen. Beim MAD-Modell handelt es sich um eine Erweiterung des Relationsmodells, die es ermöglicht sowohl hierarchische (1:n) als auch komplexe Beziehungen (n:m) auf einfache Datenstrukturen mit Wiederholungsgruppen abzubilden; damit lassen sich auch die rekursiven Beziehungen realisieren. 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. Die dynamische Molekülbildung sowie die einfachen Molekülоperationen stellen die integralen Bestandteile des Modells dar.

3.1. Grundlegende Konzepte

Die Grundelemente von MAD sind die Atomtypen. Sie sind vergleichbar mit den Relationen des Relationenmodells, d. h., sie fassen Attribute verschiedener Typen zusammen. Aus diesen Atomtypen können nun dynamische Molekültypen gebildet werden. Ein Molekültyp ist die Zusammenfassung von Atomtypen, zwischen denen definierte Beziehungstypen existieren. Die Spezifikation dieser Beziehungstypen basiert auf einem vom System unterstützten Primärschlüssel-Fremdschlüssel-Konzept, welches durch speziell dafür eingeführte Attributtypen (IDENTIFIER und REFERENCE) realisiert wird. Jeder Beziehungstyp wird immer durch beide Teilbeziehungen dargestellt, wobei sowohl (1:1)- als auch (1:n)- bzw. (n:m)-Beziehungen direkt und als Teil des Atomtyps beschrieben werden können. Hierzu wird der Wiederholungsgruppentyp SET verwendet. Damit können auf Atomebene die direkten und symmetrischen Netzstrukturen aufgebaut werden. Molekültypen entsprechen folglich speziellen Sichten auf die Menge der im Schema definierten Atom- und Beziehungstypen (also Ausschnitte auf den Netzstrukturen). Sie können selbst wieder zum Aufbau komplexerer Molekültypen benutzt werden, was außerdem auch rekursiv geschehen kann. Jeder Molekültyp legt fest, wie die zugehörigen Moleküle (rekursiv) aus anderen Molekülen bzw. Atomen aufgebaut sind. Der Begriff „Molekül" soll dabei ausdrücken, daß Atome dynamisch verschiedene Bindungen eingehen können. Dieses allgemeine und variable „Bindungskonzept" kann nur einfach zur Erzeugung der von der MA benötigten Objektinformation benutzt werden.

Die im MAD-Modell definierte Sprache MQL (Molecule Query Language) 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 Verbundqualifikation 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 von einem definierten Molekültyp. Damit ist die Abgeschlossenheit des MAD-Modells bzgl. seiner Operationen gewährleistet.

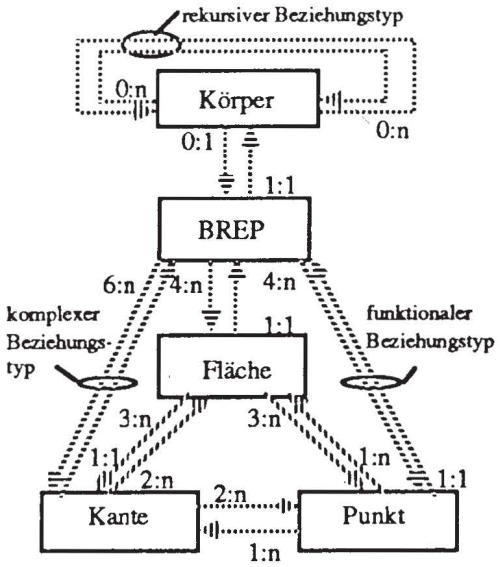
Das MAD-Modell bietet mit dem allgemeinen Konzept der dynamischen Molekülbildung und -verarbeitung die Integration von Netzstrukturen und von rekursiven Strukturen. Die Rekursion verlangt einen reflexiven Beziehungstyp, und die Netzwerkeigenschaft entsteht dadurch, daß sich die betroffenen Atommengen überlappen. Damit können dann alle Aspekte der geforderten Objektunterstützung in einfacher Weise dargestellt und verarbeitet werden. Deren jeweilige semantische Interpretation bleibt natürlich den Ebenen oberhalb des Datenmodells (Modellabbildung) vorbehalten. Von der Datenmodellseite können immer nur allgemeine Abbildungskonzepte (z. B. im NF^2-Modell [SS86] etwa die Relationenschachtelung oder hier die dynamische Molekülbildung) angeboten werden.

Im folgenden werden die wichtigsten Charakteristika des MAD-Modells anhand einer Beispielanwendung aus dem CAD-Bereich vorgestellt. Eine detailliertere Besprechung des MAD-Modells und der Sprache MQL findet sich in [Mi87, Mi88].

3.1. Datendefinition

Das DB-gestützte, volumenorientierte Modellierungssystem KUNICAD [HHLM87] benutzt zur effizienten graphischen Darstellung der konstruierten Werkstücke das Begrenzungsflächenmodell (BREP, boundary representation, [RV82]). Dieses Darstellungskonzept verwendet zur Beschreibung von Objekten deren begrenzende Flächen, welche KUNICAD durch die berandenden Kanten und deren Anfangs- und Endpunkte darstellt. Bild 2 enthält das

a) MAD-Schemadiagramm




```
CREATE ATOM_TYPE Punkt
    ( Punkt_id  : IDENTIFIER,
      Lage      : RECORD
                    x_Koord : REAL,
                    y_Koord : REAL,
                    z_Koord : REAL
                  END
      Linie     : SET_OF (REF_TO (Kante.Begrenzung)) (1,VAR),
      Fläche    : SET_OF (REF_TO (Fläche.Durchdrpunkt)) (1,VAR)
      BREP      : REF_TO (BREP.Punkte)
    )
```

b) Atomtypdefinitionen

```
CREATE ATOM_TYPE Körper
    ( Körper_id      : IDENTIFIER,
      Körper_no      : INTEGER,
      Beschreibung   : CHAR_VAR,
      Sub            : SET_OF (REF_TO (Körper.Super)) (0,VAR),
      Super          : SET_OF (REF_TO (Körper.Sub)) (0,VAR),
      BREP           : REF_TO (BREP.Körper)
    )
      KEYS_ARE (Körper_no)

CREATE ATOM_TYPE BREP
    ( BREP_id        : IDENTIFIER,
      BREP_no        : INTEGER,
      Hülle          : HULL_DIM(3),
      Körper         : REF_TO (Körper.BREP),
      Flächen        : SET_OF (REF_TO (Fläche.BREP)) (4,VAR),
      Kanten         : SET_OF (REF_TO (Kante.BREP)) (6,VAR),
      Punkte         : SET_OF (REF_TO (Punkt.BREP)) (4,VAR)
    )
      KEYS_ARE (BREP_no)

CREATE ATOM_TYPE Fläche
    ( Fläche_id      : IDENTIFIER,
      Fläche         : REAL,
      Grenze         : SET_OF (REF_TO (Kante.face)) (3,VAR),
      Durchdrpunkt   : SET_OF (REF_TO (Punkt.face)) (3,VAR),
      BREP           : REF_TO (BREP.faces)
    )

CREATE ATOM_TYPE Kante
    ( Kante_id       : IDENTIFIER,
      Länge          : REAL,
      Begrenzung     : SET_OF (REF_TO (Punkt.Linie)) (2,VAR),
      Fläche         : SET_OF (REF_TO (Fläche.Grenze)) (2,VAR),
      BREP           : REF_TO (BREP.Kanten)
    )
```

Bild 2
Eine BREP-Modellierung im MAD-Modell

zugehörige MAD-Schema graphisch als Diagramm und zusätzlich in DDL-Notation.
Jede Atomtypdefinition besteht aus 3 Teilen: Namensvergabe, Attributdefinitionsliste und Schlüsseldefinition. Der Atomtypname benennt den Atomtyp und muß über das gesamte Schema hinweg eindeutig sein. Jede Attributdefinition besitzt einen Namen (identifizierend innerhalb des betreffenden Atomtyps) und einen Datentyp. Aus Gründen der Identifikation und der Verbindung/Beziehungen von Atomen muß jede Atomtypdefinition genau ein IDENTIFIER-Attribut beinhalten. Die Schlüsselspezifikation wird als Liste von Attributnamen ausgedrückt. In Bild 2 wird auch die direkte und symmetrische Modellierung der verschiedenen Beziehungstypen zum einen im Schemadiagramm und zum anderen in den REFERENCE-Attributen der Atomtypdefinitionen veranschaulicht.

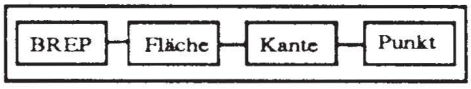
3.3. Datenmanipulation

Die operationalen Aspekte des MAD-Modells werden anhand des Bildes 3 vorgestellt. Alle Beispielanfragen beziehen sich auf das BREP-Schema aus Bild 2. Zusätzlich zur MQL-Anfrage werden jeweils die zugehörigen Molekuldiagramme aufgezeigt, die die Molekülstruktur der Anfrage darstellen. Durch die mehrere Atomtypen einschließende Umrandung wird die zu projizierende Teilstruktur aufgezeigt.
MQL unterscheidet zwischen vertikalem und horizontalem Zugriff. Bild 3a) zeigt ein Beispiel für den vertikalen Zugriff auf ein vernetztes Molekül. In der FROM-Klausel sind alle Komponenten-Atomtypen und die ausgewählten Verbindungen dazwischen angegeben, die den zu bearbeitenden Molekültyp spezifizieren. Zur Identifikation der Verbindungen muß ggf. das REFERENCE-Attribut mit angegeben werden; ansonsten genügt die Angabe der Atomtypen. Für das Anfragebeispiel in Bild 3a) heißt dies, ausgehend von den BREP-Atomen und unter Benutzung der angegebenen Verbindungen, alle erreichbaren Flächen-, Kanten- und Punkt-Atome aufzufinden. Die so aufgebauten (BREP-)Moleküle können durch die WHERE-Klausel qualifiziert werden, d. h. durch Auswertung der angegebenen Qualifikationsterme. Die Ergebnismenge der Anfrage aus Bild 3a) umfaßt daher nur die Moleküle, deren Wurzelatom BREP im Attribut BREP_no den Wert 1713 enthält. Da dieses Attribut hier Schlüsseleigenschaft besitzt, ergibt sich die Kardinalität der Ergebnismenge zu 1.
In Bild 3b) ist eine Rekursiv-Anfrage spezifiziert. Mittels der RECURSIVE-Klausel kann ein rekursiver Molekültyp definiert werden. Damit wird die zugehörige Molekülstruktur zyklisch und die Ableitung der Rekursiv-Moleküle muß schrittweise iterierend durchgeführt werden. Zuerst müssen alle Wurzelatome der Rekursiv-Moleküle bestimmt werden. Dazu wird die zugehörige „Saatinitialisierung" in der WHERE-Klausel ausgewertet. Für alle qualifizierten Wurzelatome wird dann stufenweise zur nächsttieferen Rekursionsebene vorgegangen unter Benutzung der „Körper-.Sub"-Referenzen. Die Mehrfachauswertung des gleichen Rekursionspfades wird dabei ausgeschlossen, wodurch letztendlich die Terminierung gewährleistet ist.
Der horizontale Zugriff wird durch die Beispielanfrage in Bild 3c) vorgestellt. Es werden alle (Körper-)Atome selektiert, deren Teilhierarchie Körper.Sub leer ist. In der zugehörigen SELECT-Klausel wird eine (unqualifizierte) Projektion durchgeführt: alle Atome bzw. nur die angegebenen Attribute werden projiziert.
In Bild 3d) sind gleich mehrere Besonderheiten enthalten:

- Durch die Verwendung von geklammerten Ausdrücken können beliebige Molekülstrukturen innerhalb der FROM-Klausel definiert werden. Hier wird ein baumförmiger Molekültyp spezifiziert.

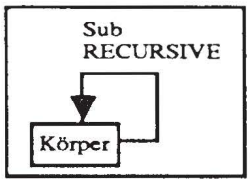
a) Vertikaler Zugriff auf vernetzte Moleküle

```
SELECT  ALL
FROM    BREP-Fläche-Kante-Punkt
WHERE   BREP_no = 1713                    (* Qualifikation *)
```



b) Vertikaler Zugriff auf Rekursiv-Moleküle

```
SELECT  ALL
FROM    Komp_hier (Körper)(RECURSIVE:Körper.Sub-Körper)
WHERE   Komp_hier [0].Körper_no = 4711 .      (* Saatinitialisierung *)
```



c) Horizontaler Zugriff zusammen mit unqualifizierter Projektion

```
SELECT  Körper_no, Beschreibung            (* unqualifizierte Projektion *)
FROM    Körper
WHERE   Sub = EMPTY
```

d) "gemischte" Anfrage

```
SELECT  Kante, (Punkt,
                Fläche := SELECT Fläche_id, Fläche
                          FROM  Fläche          (* qualifizierte Projektion *)
                          WHERE Fläche_dim > 1.9E4)
FROM    BREP-Kante (Fläche, Punkt)
WHERE   BREP_no = 1713
        AND
        EXISTS_AT_LEAST (2) Kante : Kante.Länge > 1.0E2)
                                          (* qualifizierte Restriktion *)
```

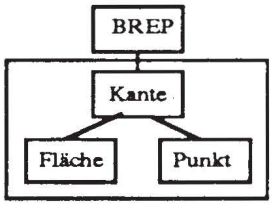


Bild 3
Einige Beispielanfragen

- Quantifizierte Qualifikationsterme (ALL, EXISTS) erhöhen die Ausdruckskraft der WHERE-Klausel entscheidend. Der Existenzquantor testet, daß alle Ergebnismoleküle wenigstens 2 Kanten besitzen, die das Längen-Prädikat erfüllen.
- Die qualifizierte Projektion erlaubt eine verfeinerte Projektion. Durch Verwendung des „SELECT ... FROM ... WHERE"-Konstrukts innerhalb der SELECT-Klausel lassen sich die Komponenten der Ergebnismoleküle abhängig von den Qualifikationstermen der WHERE-Klausel (innerhalb der SELECT-Klausel) bestimmen. Hier werden nur solche Flächen-Atome des „Kanten-(Fläche, Punkt)"-Moleküls projiziert, die die Flächen-Qualifikationsbedingung erfüllen. Mit Hilfe dieses Konstrukts ist es möglich, nur solche Komponenten der „umgebenden" Ergebnismoleküle zu isolieren, die wirklich für die weitere Verarbeitung relevant sind; die Auswahl der Moleküle der Ergebnismenge geschieht (logisch) immer vorher und ist unabhängig von der qualifizierten Projektion.

Ähnlich zu den hier vorgestellten Anfragemöglichkeiten erlauben die Operationen INSERT, DELETE und UPDATE sowohl das Arbeiten mit vollständigen Molekülen als auch mit Komponentenmolekülen. Näheres hierzu findet sich in [Mi88].

4. Die Modellabbildung

Durch die Modellabbildung (MA, auch anwendungsmodell-unterstützende Schicht genannt) soll der neutrale NDBS-Kern um anwendungsbezogene Datenstrukturen mit darauf definierten Operationen erweitert werden. Hierdurch wird die einfache Handhabung eines Anwendungsmodells ermöglicht. Die Objekte der Anwendungsmodellschnittstelle werden demnach durch die MA auf die Objekte der darunterliegenden Datenmodellschnittstelle abgebildet. Diese Transformation ist zu einem durch die Mächtigkeit des Datenmodells (MAD), zum anderen durch die arbeitsplatzrechner-basierte Hardware-Architektur bestimmt.

Vor Nutzung der MAD-Funktionalität durch die Modellabbildung muß zunächst die Frage der Datenmodellanbindung, d. h. die Frage nach einem adäquaten Verarbeitungsmodell geklärt sein. Wie werden die Datenmodellobjekte der MA zur Verfügung gestellt, wie werden sie dort verarbeitet, und wie werden Modifikationen im NDBS-Kern reflektiert?

Das in konventionellen DBS vorherrschende Verarbeitungsmodell ist zugeschnitten auf die Komplexität der mit klassischen Datenmodellen beschreibbaren Objekte. Durch eine satzorientierte Subschema-Verarbeitung ist eine Schnittstelle bestimmt, an der einfach-strukturierte (atomare) Objekte bereitgestellt, durch das Anwendungsprogramm verarbeitet (gelesen, aktualisiert, ergänzt) und anschließend unmittelbar an das DBS zurückgegeben werden können. Der Aufwand für wiederholte Zugriffe kann durch Lokalität im DBS-Puffer zwar reduziert werden; er muß (speziell im Umfeld der Ingenieuranwendungen) allerdings als noch viel zu hoch eingestuft werden.

In NDBS benötigt man demgegenüber Verarbeitungskonzepte, die der größeren Komplexität der zu verwaltenden Objekte Rechnung tragen. Bild 4 skizziert die aufgrund einer MAD-Anfrage zu erwartende Molekülausprägungsmenge. Hierdurch soll die heterogene Struktur der Objekte verdeutlicht werden, deren Bereitstellung, Verarbeitung und Rückgabe es zu unterstützen gilt. Einen Ansatz hierzu sehen wir in dem Einsatz eines anwendungsnahen Objektpuffers als Übergabe- und Verarbeitungsbereich für molekulare Objekte. Die Verarbeitung der in den Puffer eingelagerten Objekte mit einer konventionellen Programmiersprache kann durch ein Cursor-Konzept entsprechend unterstützt werden. Die durch den Puffer eingeführte „Lokalität in Anwendungsnähe" verspricht, vor allem vor dem Hintergrund der Rechnergrenze zwischen Modellabbildung und NDBS-Kern, eine Verbesserung der Leistungsfähigkeit des Gesamtsystems, da eine erhebliche Reduktion der rechnerübergreifenden Zugriffe erreicht wird [HHLM87, Hä87].

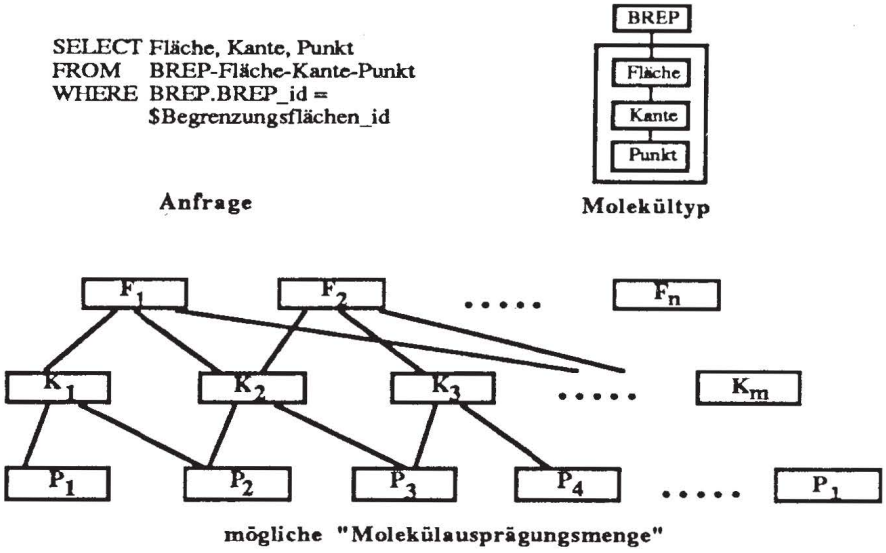


Bild 4
Zusammenhang zwischen MAD-Anfrage, Molekültyp und Molekülausprägung

4.1. Strukturierung der Modellabbildung

Bild 5 zeigt den prinzipiellen Aufbau der Modellabbildung unter besonderer Berücksichtigung der Komponenten zur Datenmodellanbindung, die im wesentlichen die Verwaltung der Objektpuffer-Datenstrukturen bewirken. Der Objektpuffer dient zur Aufnahme der selektierten, molekularen Objektmenge. Alle Atome, als die elementaren Bausteine der sich bzgl. einer Anfrage qualifizierender Moleküle, werden im Objektpuffer als eine Ergebnismenge repräsentiert. Die Molekülstruktur wird durch spezielle Atomattribute, die Referenzattribute, gebildet und ist daher „automatisch" in der Atommenge repräsentiert. Innerhalb dieser Menge sind Atome netzartig miteinander verbunden; sie treten nicht mehrfach auf, auch wenn sie zur Bildung verschiedener Moleküle beitragen. In Bild 4 trägt das Kanten-Atom K2 zur Bildung der durch F1 und F2 bestimmten „Fläche-Kanten-Punkt"-Moleküle bei; es ist in der Ergebnismenge jedoch nur einmal repräsentiert. Dagegen kann es i. a. zwischen mehreren Ergebnismengen zu Redundanzen kommen.
Eine Ursache solcher Redundanzen liegt darin begründet, daß die Ergebnismenge als eine anfragebezogene Kopie der DB-Daten anzusehen ist. Sie stellt damit eine Art strukturierte Programmvariable dar, der durch eine Selektionsanweisung ein bestimmter, komplexer Wert zugewiesen wird. Mehrfachrepräsentationen der Daten eines Atoms treten dann auf, wenn verschiedene Programmvariable teilweise gleiche Komponentenwerte aufweisen. Dies ist allerdings den anwendungsbezogenen Programmen bekannt und kann von ihnen gehandhabt werden. Andere Interpretationen einer Ergebnismenge sind denkbar und in manchen Fällen auch wünschenswert. Das Spektrum reicht hier von molekülbezogenen bis hin zu programm- oder subschemabezogenen Ergebnismengen. Ein Molekülbezug ist dann gegeben, wenn sich eine Ergebnismenge als ein Zusammenschluß disjunkter, molekularer Programmvariablen darstellt. Bei einer solchen Sichtweise können Atome bereits dann mehrfach auftreten, wenn sie Bestandteil verschiedener Moleküle sind. In einer programmbezogenen Ergebnisdarstellung kommt es dagegen zu keinerlei Mehrfachrepräsentationen. Die Resultate verschiedener Anfragen eines Programms werden in einer einzigen Darstellung quasi ineinander „gemischt". Die Kontrolle hierüber liegt im Verantwortungsbereich der verarbeitenden Programme. Ein Konzept zur flexiblen Datenmodellanbindung

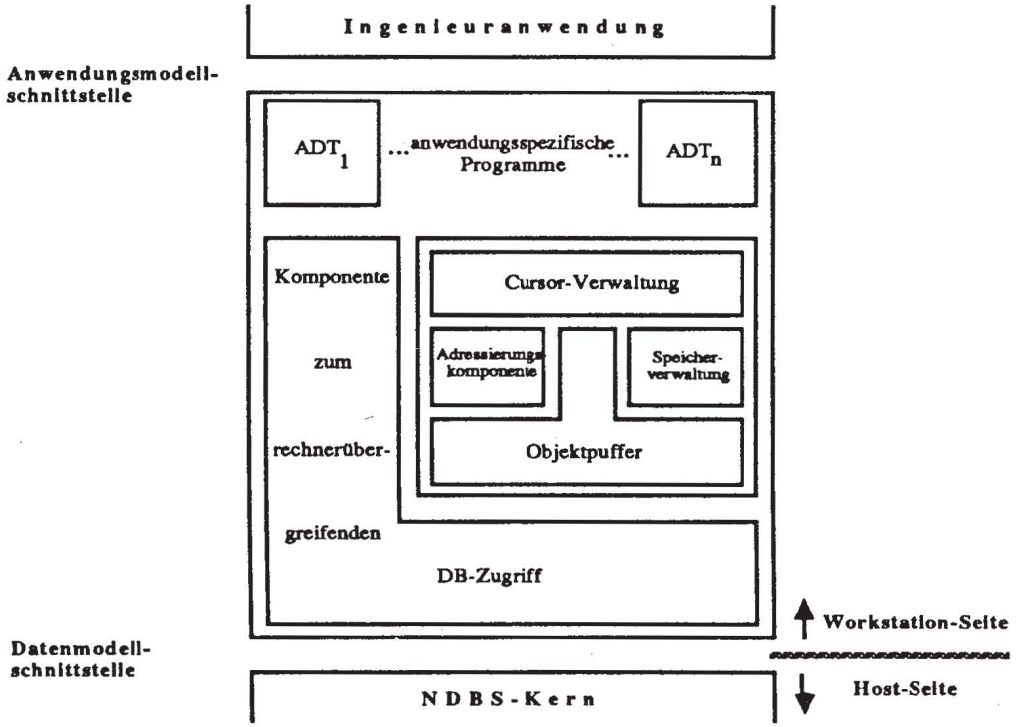


Bild 5
Aufbau der Modellabbildung

wird sicherlich in der ein oder anderen Form diese drei Varianten berücksichtigen. Da aber insgesamt noch wenig Klarheit über die konkreten Auswirkungen auf die Verarbeitung der komplexen Objekte besteht, haben wir uns aus pragmatischen Gründen zunächst für die zuerst vorgestellte Interpretationsvariante, die anfragebezogene Ergebnismenge, entschieden. Diese Variante bildet daher die Grundlage der im folgenden vorgestellten Objektpufferstruktur.

4.2. Objektpufferverwaltung

Bild 6 stellt die Datenstruktur zur Repräsentation einer Ergebnismenge im Objektpuffer dar. Ausgehend von einem Ergebnismengen-Deskriptor läßt sich die gesamte Struktur in zwei Teile aufspalten. Die Area-Tabelle und die Area-Blöcke (kurz Area) dienen zur Kontrolle des durch die Atome belegten Speicherbereichs. Atomtyp- und Atom-Tabellen dagegen ermöglichen eine Adressierung über logische Identifikatoren. Das Area-Konzept soll einer zu starken Zerstückelung des Hauptspeichers bei einer großen Anzahl kleiner Atome entgegenwirken. Darüberhinaus wird eine leichte Verschiebbarkeit der gesamten Ergebnismenge erreicht. Diese Eigenschaft ist durchaus bedeutungsvoll, denkt man nur daran, daß sich die Verarbeitung i. a. über einen längeren Zeitraum (Tage, Wochen) erstreckt und es daher notwendig sein wird, ein Aus- bzw. Einlagern zwischen Workstation und zugeordneten Sekundärspeicherbereich zu unterstützen. Die Atomtyp-Tabelle nimmt eine Separierung der Atome einer Ergebnismenge nach ihrer Typ-Zugehörigkeit vor. Hierdurch wird der typbezogene Zugriff auf Atome ermöglicht. Im wesentlichen enthält ein Eintrag der Atomtyp-Tabelle den internen Namen des Atomtyps und einen Verweis auf die entsprechende Atom-Tabelle. Eine Komponente dieser Tabelle beinhaltet die Verwaltungsinformation eines Atoms. Neben einem Modifikations-Flag (Mod-Flag), das angibt, ob und wie ein Atom verändert wurde, ist durch den Area-Index und den Area-Offset weitere Adressierungsinformation gegeben. Der Area-Index bestimmt den Eintrag in der Area-Tabelle, der dann die Basisadresse der entsprechenden Area enthält. Innerhalb dieser Area befindet sich das gewünschte Atom an der durch den Area-Offset bestimmten Position. Die Verwaltung der Adressierungsinformation sowie die Durchführung der Adreßberechnung ist Aufgabe der Adressierungskomponente. Die Organisation der Areas übernimmt die Speicherverwaltungskomponente. Sie kontrolliert neben der Area-Tabelle die Freispeicherinformation jeder einzelnen Area.

Die Adressierungskomponente erlaubt lediglich den direkten Zugriff auf ein identifiziertes Atom. Zur Unterstützung des navigierenden Zugriffs innerhalb einer Ergebnismenge dient die Cursor-Verwaltungskomponente. Sie stellt den anwendungsbezogenen Programmen der Modellabbildung ein mächtiges Cursor-Konzept zur Verfügung, das eine Verarbeitung der komplex-strukturierten Atomnetze über hierarchische Cusor gestattet. Hierbei setzt sich der hierarchische Cusor aus einer Reihe voneinander abhängiger, flacher, atomtyp-bezogener Cursor zusammen und ermöglicht das Durchlaufen von klar definierbaren Untereinheiten des durch eine Ergebnismenge gegebenen Atomnetzes. Neben dem Cursor-Zugriff stellt die Cursor-Verwaltungskomponente weitere Operationen, wie z. B. das Anlegen, das Zuweisen und das Löschen der Cursor, zur Verfügung.

4.3. Datenmodellanbindung

Die bisher vorgestellten Komponenten ermöglichen die lokale Verarbeitung der molekularen Objekte im Arbeitsplatzrechner. Die Evaluierung der betreffenden Ergebnismenge sowie die Realisierung der eigentlichen MAD-Funktionalität (vgl. Kapitel 3) erfolgt auf dem zentralen Rechner durch das NDBS-Kernsystem. Der rechnerübergreifende DB-Zugriff wird durch eine spezielle NDBS-Schnittstellen-Komponente angeboten. Ihre Aufgabe besteht u. a. in der Abwicklung des notwendigen Datenaustausches. Die erforderlichen Informationen gehen über die der unmittelbaren Verarbeitungsobjekte hinaus. Neben den „Rohdaten" müssen Anweisungs- und Metainformationen transferiert werden. Die folgende Zusammenstellung gibt eine Charakterisierung der zu übertragenden Informationen:

1. **Anweisungen**
   Die einfachste Möglichkeit zur Übertragung von DB-Anweisungen besteht in dem Transfer von Zeichenketten. Auf der Seite des NDBS-Kerns sind diese dann zu interpretieren bzw. zu übersetzen. Zu einer optimierten Transfer-Darstellung von Datendefinitions- und Datenmanipulationsanweisungen gelangt man durch eine syntaktische Überprüfung und Voranalyse auf der Seite des Arbeitsplatzrechners. Hierzu ist allerdings dort ein entsprechender Parser vorzusehen. Einen weiteren und wesentlichen Performance-Gewinn sehen wir in einer möglichst vollständigen Anweisungs-Vorübersetzung. Das Übersetzungsergebnis kann als Ausführungsmodul durch den NDBS-Kern werwaltet werden. Eine längerfristige Speicherung der Ausführungsmodule ist, speziell in dem von uns unterstellten Szenario, durchaus

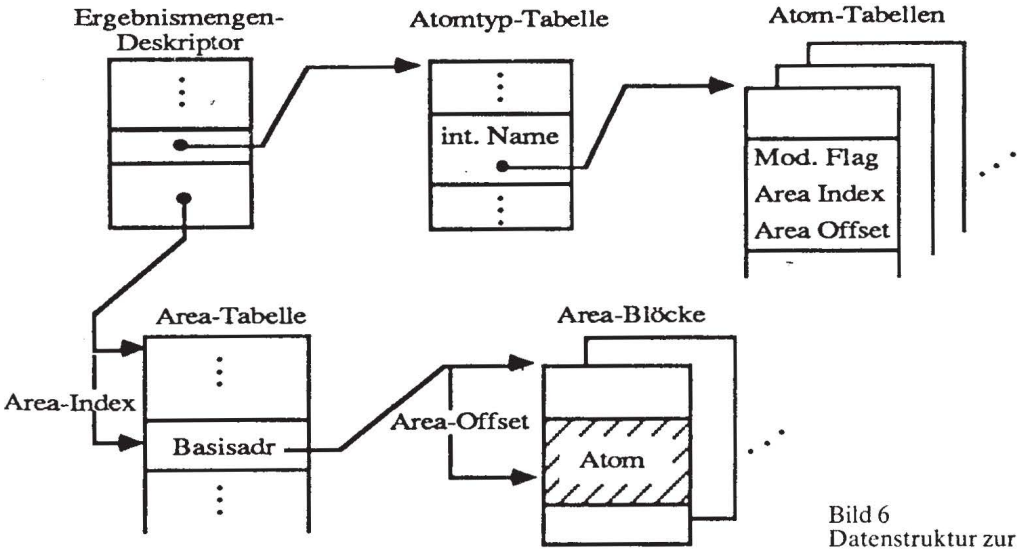


Bild 6
Datenstruktur zur Repräsentation einer Ergebnismenge

sinnvoll, da mit einer hohen Stabilität der einmal etablierten Modellabbildungskomponenten zu rechnen ist. Dies gilt insbesondere, da das MAD-Modell eine Anweisungsparametrisierung erlaubt (Parameter werden durch das Symbol „$" gekennzeichnet; siehe Bild 4 $ Begrenzungsflächen id). Bei eienr späteren Anweisungsaktivierung durch Programme der Modellabbildung muß dann lediglich eine Kennung des entsprechenden Ausführungsmoduls zusammen mit den aktuellen Parameterwerten übertragen werden.

2. **Daten**
Für alle MAD-Anweisungen zur Datenmodifikation (INSERT, DELETE, UPDATE) ist die Möglichkeit einer „direkten" Datenspezifikation vorgesehen; dies entspricht der Bereitstellung eines Resultats nach einer Retrieval-Anweisung. Die Daten, die gelesen, eingefügt, geändert und gelöscht werden sollen, werden unmittelbar im Anschluß an eine entsprechende Anweisung übergeben. Daher sind Antwortinformationen bzw. Einfüge-, Änderungs- und Löschinformationen vom NDBS-Kern zur Modellabbildung (bzw. umgekehrt) zu übertragen. Die Anwortinformation beschreibt die Menge derjenigen Atome, die zur Bildung der Moleküle, die sich bzgl. einer Anfrage qualifizieren konnten, erforderlich sind. Die spezifizierte Molekülstruktur muß sich in der Anwortinformation wiederfinden. Die Einfüge- und Änderungsinformation umfaßt diejenigen Daten, die zum Einbringen von neuen bzw. geänderten Atomen in den Datenbestand des NDBS-Kernsystems notwendig sind. Das „direkte" Einfügen und Ändern von Atomen dient zur Modifikation der Moleküle, die durch eine vorangegangene Retrieval-Anweisung evaluiert wurden. Daher kann die entsprechende Information als eine Art „Delta-Information" organisiert sein und muß nicht notwendigerweise eine bestimmte Molekülstruktur aufweisen. Das gleiche gilt für die Löschinformation, die beim „direkten" Löschen aus einer Liste von Identifikatoren der zu eliminierenden Atome besteht.

3. **Metadaten**
Zwischen NDBS-Kern und Modellabbildung werden äußerst komplexe Datenstrukturen ausgetauscht. Die Verarbeitung dieser Daten erfordert eine Absprache über deren konkrete Strukturierung. Diese ist i. a. anweisungsspezifisch. Daher erfolgt die Übertragung anweisungsbezogener Metainformationen. Hierbei sind zwei Varianten denkbar: zum einen können die Metainformationen so eng mit den Daten verbunden sein, daß sie einen integralen Bestandteil bilden. Die Daten sind dann „selbstbeschreibend". Zum anderen kann der Teil der Metadaten, der anweisungsspezifisch aber ausprägungsunabhängig ist (als Typ-Information), von den „eigentlichen" Daten separiert und eigenständig übertragen werden. Hierdurch kann der wiederholte Austausch gleicher Metadaten weitestgehend vermieden werden.

4.4. Dynamische Aspekte

Nachdem wir nun die mehr statischen Aspekte der Datenmodell-Anbindung anhand einzelner Komponenten diskutiert haben, wollen wir unsere Vorstellungen über die dynamischen Aspekte der Anbindung erläutern. Es sollen nur die grundlegenden Konzepte vorgestellt werden, ohne daß dabei die Auswirkungen auf ein zu unterstellendes Transaktionsmodell Berücksichtigung finden. Diese werden in [HHMM88] diskutiert. Das auslösende Moment für das Bereitstellen von selektierten Daten im Objektpuffer und für das Einbringen von Änderungen in die Datenbank sehen wir in der expliziten Aktivierung eines Checkout- bzw. Checkin-Operators. Die Checkout-Operation führt zur Bearbeitung einer oder mehrerer Datenmodell-Operationen (SELECT) durch das NDBS-Kern-System und damit zur Evaluierung derjenigen Daten, die zur Repräsentation eines Anwendungsobjektes erforderlich sind. Das Reflektieren von Modifikationen der Anwendungsobjekte im NDSB-Kern wird dagegen durch die Checkin-Operation angestoßen und auf ein oder mehrere INSERT-, DELETE- oder UPDATE-Operationen abgebildet. In [HHMM88] wird eine Einschränkung auf eine bestimmte Sequenz von Checkout- und Checkin-Operationen innerhalb eines Verarbeitungsabschnittes begründet. Einer beliebigen Anzahl von Checkout-Aufrufen folgt hierbei maximal ein Checkin-Aufruf. Die wesentlichen Gründe basieren auf der Assoziation von Transaktionseigenschaften mit einem Verarbeitungsabschnitt sowie auf den Auswirkungen auf die Fehlerisolation bei einer Workstation-basierten Hardware-Architektur. Als weitere Operationen zur Strukturierung der Dynamik von Verarbeitungsabschnitten werden vorgeschlagen:

- **Start/End**, zum Starten und Beenden von Verarbeitungsabschnitten.
- **Save**, zur Sicherung des gegenwärtigen Verarbeitungsstandes (es wird der aktuelle Objektpufferinhalt auf privaten, dem Arbeitsplatzrechner zugeordneten, Sekundärspeicher ausgelagert),
- **Restore**, zur Wiederherstellung eines definierten, vorher gesicherten Verarbeitungsstandes,
- **Suspend**, zur Unterbrechung des aktuellen Verarbeitungsabschnittes sowie
- **Resume**, zur Fortsetzung einer bestimmten Verarbeitung.

Bild 7 skizziert unsere Vorstellungen über das Zusammenspiel der einzelnen Operationen.
Die weiteren Komponenten der Modellabbildung sind anwendungsorientiert. Sie realisieren die Operationen einer Reihe abstrakter, anwendungsbezogener Datentypen (ADT) und ermöglichen somit die Modellierung anwendungsnaher Objekte. Die Frage der Anbindung dieser ADT's, als Schnittstelle für die darüberliegende Anwendung, ist Gegenstand des folgenden Kapitels.

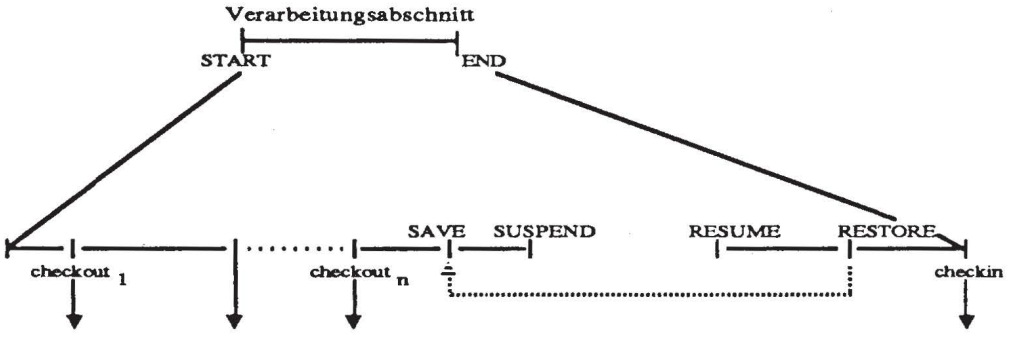


Bild 7
Ablaufstruktur eines Verarbeitungsabschnittes

5. Anwendungsanbindung

Bevor wir in diesem Kapitel die Frage der Anwendungsmodellanbindung diskutieren, soll zunächst der Begriff der Anwendung näher erläutert werden. Die Modellabbildung stellt an ihrer oberen Schnittstelle ein Anwendungsmodell bereit. Alle Programme, die mit den Objekten dieses Anwendungsmodells arbeiten, werden als Anwendungsprogramme bezeichnet. Im einfachsten Fall besteht die Anwendung lediglich aus Programmen zur Realisierung einer Benutzeroberfläche. Der Benutzer arbeitet dann direkt mit den Anwendungsmodell-Objekten. In einem anderen Fall arbeiten komplexe Programmpakete auf einem Anwendungsmodell. Der Benutzer ist hierbei weniger an den Objekten des Anwendungsmodells als an den Resultaten der Anwendungsprogramme interessiert (vgl. Bild 8). Das folgende gibt ein Beispiel aus dem CAD-Bereich: Das Anwendungsmodell besteht aus einem geometrischen 3 D-Modell, z. B. auf der Grundlage von Basiskörper- oder Begrenzungsflächen-Repräsentationen. Als ein einfaches Anwendungsprogramm könnte ein Werkstück-Editor als Benutzerschnittstelle vorhanden sein. Daneben kommen u. a. Festigkeitsberechnungen oder Bewegungssimulationen als komplexere Anwendungen in Betracht.

5.1 ADT-Ansätze

Die Modellabbildungsschnittstelle zur Anwendung hin wird in unserer bisherigen Vorstellung durch das Konzept der abstrakten Datentypen bestimmt. Neben der Durchführung der speziell auf den ADT's definierten Operationen, muß es in der Anwendung die Möglichkeit geben, ADT-Ausprägungen zu generieren bzw. zu löschen oder einander zuzuweisen. Neben den spezifischen sind also eine Reihe allgemeiner Operationen erforderlich.
Entsprechende Ansätze sind u. a. [DKM86, LKD87] aufgezeigt. [LKD87, KKLW87] beschreiben die Integration von sog. „Benutzer-definierten Datentypen" (BDT). Die BDT's selbst werden durch Pascal-Funktionen und -Strukturen definiert und können durch das zugrundeliegende Datenbanksystem verwaltet werden [Da86]. Darüber hinaus kann die BDT-Funktionalität in Form von „built-in"-Funktionen bei DB-Anfragen genutzt werden. In [RS87] wird ein ähnlicher Ansatz verfolgt. Das POSTGRES-Datenmodell erlaubt die Handhabung und die Integration von sog. „user defined procedures" (UDP).
Die angebotenen Konzepte ermöglichen in der Regel eine horizontale oder vertikale Erweiterung der Systemschnittstelle (Bild 9). Durch die Definition neuer ADT's (BDT's, UDP's) ist eine Extension eines bestehenden Anwendungsmodells um weitere Objekte, besser Objektklassen, zu erreichen (horizontale Erweiterung). Ein Beispiel hierfür ist die Ergänzung eines auf Begrenzungsflächen beruhenden 3 D-Geometriemodells um eine Basiskörper-Geometrie. Werden bei der ADT-Definition vorhandene ADT's benutzt, so entsteht eine Schachtelung von ADT's; man bewirkt eine Verfeinerung bzw. Spezialisierung des Anwendungsmodells (vertikale Erweiterung). Eine solche Spezialisierung wäre z. B. gegeben durch den Übergang von einem Geometriemodell auf ein „Bauteilmodell". Die Objekte dieses Bauteilmodells, also Ausprägungen der neu definierten ADT's, basieren auf den Objekten des Geometriemodells. Die bauteilspezifischen Operationen, wie das Anfertigen einer Bohrung, das Fräsen einer Nut oder das Verbinden von Bauteilen, werden auf die geometriespezifischen Operationen, wie u. a. Differenz oder Vereinigung zurückgeführt.
Durch den ADT-Ansatz können Objekte mit komplexer Internstruktur als Einheit nach anwendungsnahen Gesichtspunkten manipuliert werden. Ein solcher „intra" oder „vertikaler" Objektzugriff ist an einer Anwendungsmodellschnittstelle unbedingt erforderlich. Darüber hinaus erscheint ein „inter" oder „horizontaler" Objektzugriff in Form einer deskriptiven Anfragemöglichkeit äußerst brauchbar. Hierdurch wird der Zugriff auf Objekte einer homogenen Menge von solchen komplexen Objekten möglich. Die oben bereits erwähnten und in [LKD87] beschriebenen built-in-Funktionen der BDT's bieten hierzu eine Grundlage. Eine Anfrage wie etwa „finde alle Flächen eines Körpers mit einer bestimmten Mindestgröße" ist damit formulierbar, wenn für den BDT „Fläche" eine Operation „Bestimmung der Flächengröße" definiert ist. Aufbauend auf den entsprechenden Operationen sind an dieser Stelle beliebig komplexe Anfragekriterien denkbar (z. B. Suche nach topologischen, geometrischen, temporalen oder funktionalen Gesichtspunkten).

5.2. Integritätserhaltung

Zur Manipulation einer ADT-Ausprägung sind keine Informationen über deren Realisierung erforderlich. Einen weiteren, für uns wichtigen Vorteil des ADT-Konzeptes sehen wir in der Zuverlässigkeit der ADT-Operationen. Sie können als „sicher" und damit als integritätserhaltend angenommen werden, da sie Bestandteil der Modellabbildung und damit in das NDBS integriert sind. Die Objekte des Anwendungsmodells, die ja durch die ADT-Ausprägungen repräsentiert werden, sind demnach konsistent (wenigstens in einem eingeschränkten Sinne), solange sie ausschließlich durch die ADT-Operationen verändert werden. Diese ADT-interne Integrität kann zum einen durch eine Vielzahl von Abfragen bzw. einfach durch einen „korrekten" Operationscode gewährleistet sein. Zum anderen ist es möglich, im Gegensatz zu einer eher prozeduralen Integritätsbeschreibung, diese als Prädikate an der Datenmodellschnittstelle zu formulieren und durch das NDBS-Kern-

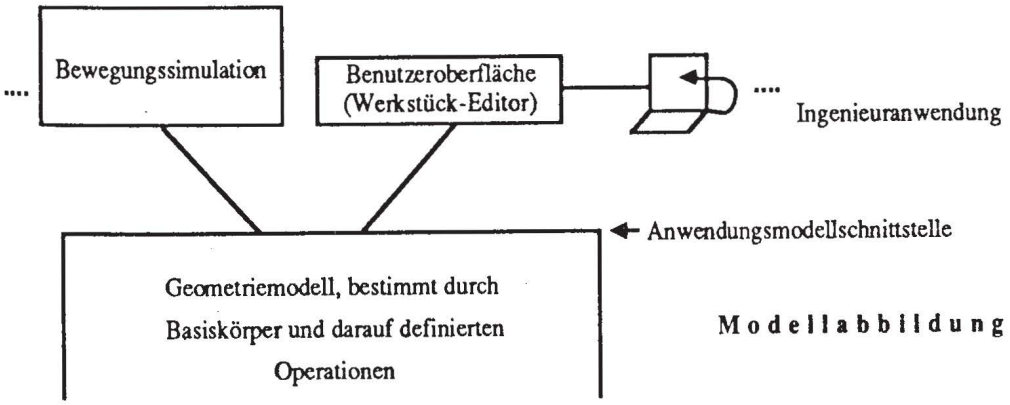


Bild 8
Beispiele für Anwendungen oberhalb der MA

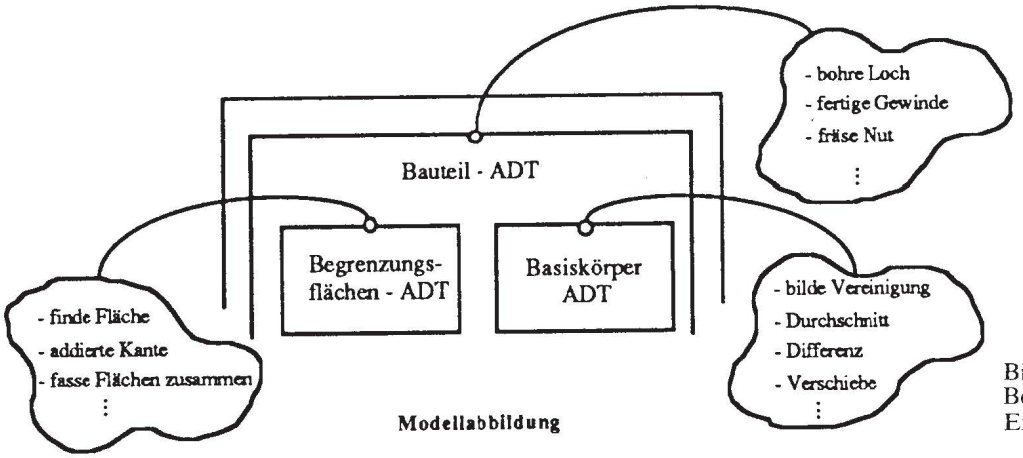


Bild 9
Beispiel für horizontale und vertikale Erweiterung des Anwendungsmodells

system überwachen zu lassen. Ein objekt- oder operationsübergreifender Konsistenzbegriff kann allerdings nicht so einfach in die ADT's integriert werden. Hier muß die Anwendung bislang selbst dafür Sorge tragen, daß die Objekte durch eine Folge von ADT-Operationen so manipuliert werden, daß sie am Ende einen Zustand erreicht haben, der dem übergeordneten Konsistenzbegriff genügt. In manchen Fällen kann die Anwendung durch spezielle ADT-Prüfoperationen unterstützt werden. Wünschenswert ist allerdings darüber hinaus eine (allgemeine) Beschreibungsmöglichkeit für solche Integritätsbedingungen, wodurch auch die Einhaltung eines objektübergreifenden Konsistenzbegriffes in die Modellabbildung verlagert und damit innerhalb des NDBS garantiert werden kann.

5.3. Beschreibungsmodelle für das Anwendungsmodell

Die skizzierten Konzepte suggerieren im Grunde ein „generisches" Anwendungsmodell. Insgesamt besteht allerdings wenig Klarheit darüber, wie denn solche Modelle generell zu beschreiben sind. Das ADT-Konzept kann hier sicherlich nur als ein erster Ansatz angesehen werden. Gegenwärtige Forschungsbemühungen zielen darauf ab, fundierte Aussagen über eine allgemeine Brauchbarkeit verschiedener Beschreibungskonzepte zu erhalten. Methoden aus dem Bereich der Wissensrepräsentation und Konzepte aus dem Umfeld neuerer Programmiersprachen, die im weitesten Sinn auf eine Objektmodellierung hinzielen, kommen als aussichtsreiche Kandidaten in Betracht. Allerdings ist in Frage zu stellen, ob eines der konkret existierenden Beschreibungswerkzeuge unverändert übernommen werden kann.
Ein wesentlicher Grund hierfür ist darin zu sehen, daß die Miniwelt „Ingenieuranwendung" eine große Heterogenität bzgl. der in ihr vorherrschenden Objekte aufweist. So existieren in bestimmten Bereichen eine große Anzahl von Objektausprägungen von insgesamt nur wenigen Objekttypen. Für andere Aspekte gilt dann ein umgekehrtes Verhältnis; viele Objekttypen mit nur wenigen Ausprägungen. Manche im Anwendungsmodell zu modellierenden Sachverhalte lassen sich angemessen nur prozedural und nicht faktisch, d. h. in Form von Daten beschreiben. Andere Zusammenhänge lassen sich nur prädikativ oder durch Regeln adäquat formulieren. Modellierungskonzepte, die sich nur auf eine dieser Darstellungsformen beschränken, führen ggf. zu einer unangemessenen Miniwelt-Beschreibung und damit zu einer inadäquaten und mit hoher Wahrscheinlichkeit ineffizienten Handhabung der dort relevanten Objekte. In [Wi87] wird daher eine sog. „mixed-representation" vorgeschlagen. Hierdurch wird jeweils die Verwendung einer adäquaten Beschreibungsform aus mehreren möglichen unterstützt. Dieser Ansatz erscheint uns vielversprechend, wenngleich das Zusammenwirken und die gegenseitige Beeinflussung der einzelnen Beschreibungsmethoden einer genaueren und intensiven Analyse bedürfen.

6. Zusammenfassung

Ausgehend von den in Ingenieuranwendungen auftretenden Anforderungen an Rechnersysteme, speziell an deren Datenhaltung, konnten wir eine Zweiteilung der Hardware- und Software-Architektur solcher Systeme begründen. Die physische Trennung von Arbeitsplatz- und Zentralrechner korrespondiert mit der logischen Systempartitionierung in einen anwendungsbezogenen und einen anwendungsunabhängigen Teil. Dies spiegelt sich in unserem Ansatz für eine NDBS-Architektur wider. Ein neutraler NDBS-Kern übernimmt die Kontrolle über den gemeinsamen Datenbestand, garantiert die Sicherung der Daten, überprüft ein Mindestmaß an Integrität und regelt den Mehrbenutzerbetrieb durch die anwendungsorientierten Systemkomponenten (Modellabbildung). Neben der reinen Rechnerleistung gibt es zwei weitere, wesentliche Einflußfaktoren auf die Leistungsfähigkeit des Gesamtsystems und damit auf die Tauglichkeit des NDBS-Architekturansatzes insgesamt. Dies sind zum einen die Modellierungsmächtigkeit an der NDBS-Kern-Schnittstelle und zum anderen die Möglichkeit, die modellierten Objekte in der Modellabbildung auch tatsächlich zu verarbeiten.
Das MAD-Modell erfüllt die essentiellen Eigenschaften, die vor allem im Bereich der Ingenieuranwendungen an die Modellierungsfähigkeit von Datenhaltungssystemen gestellt werden. Es erlaubt insbesondere:

- die direkte und symmetrische Modellierung von rekursiven und netzwerkartigen Strukturen über die explizite Beschreibung von Referenzattributen,
- die dynamische Definition der gewünschten Verarbeitungsgranulate durch das Konzept der dynamischen Molekülbildung sowie
- eine adäquate Objektverarbeitung basierend auf molekülbezogenen Zugriffs- und Modifikationsfunktionen.

Die Datenmanipulationssprache des MAD-Modells orientiert sich an relationalen Sprachen und bietet somit eine deskriptive Beschreibungsmöglichkeit derjenigen Daten und Strukturen, die dann letztendlich in die Modellabbildung den Gegenstand der Verarbeitung ausmachen.
Als wesentliche Konzepte zur Verarbeitung der durch das MAD-Modell beschreibbaren Objekte in der Modellabbildung sind hier zu nennen:

- Pufferung der komplex-strukturierten Atommengen in Übergabe- und Verarbeitungsbereichen (Objektpuffer), um somit eine „Lokalität in Anwendungsnähe" zu erreichen,

- Programmiersprachliche Verarbeitung von Molekülmengen unterstützt durch ein mächtiges Cursor-Konzept,
- Reduktion der zwischen NDBS-Kern und Modellabbildung auszutauschenden Daten durch Vorübersetzung und Speicherung von Ausführungsmoduln im Kernsystem, durch getrennte Übertragung von Daten und Metadaten und, falls möglich, durch den Austausch von „Delta-Information".

Unsere gegenwärtigen Vorstellungen über die Schnittstelle zwischen Modellabbildung und der eigentlichen Anwendung (Anwendungsmodell) sind durch das ADT-Konzept bestimmt. Wir sehen hierbei das ADT als ein „Werkzeug" zum vertikalen Zugriff innerhalb eines komplexen Objektes. Die Notwendigkeit einer allgemeinen Anwendungsmodellierung, d. h. der Beschreibung eines konkreten Anwendungsmodells durch eine allgemeine Beschreibungsmethodik, wurde aufgezeigt. Hierzu konnten noch keine eindeutigen Lösungen vorgeschlagen werden. Es wurde allerdings versucht, die zentralen Anforderungen und Fragestellungen zu formulieren sowie die zukünftigen Forschungsaktivitäten zu motivieren.

Literatur

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

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

DKM86 *Dittrich, K. R., Kotz, A. M., Mülle, J. A.:* Database Design for VLSI-Design: The DAMASCUS System, in: CAD-Schnittstellen und Datentransferformate im Elektronik-Bereich, Springer-Verlag, 1986.

HHLM87 *Härder, T., Hübel, Ch., Langenfeld, S., Mitschang, B.:* KUNICAD – A Database System Supported Geometrical Modeling Tool for CAD Applications (in German), Informatik Forschung und Entwicklung, Vol. 2, No. 1, 1987, pp. 1–18.

HR85 *Härder, T., Reuter, A.:* Architecture of Database Systems for Non-Standard Applications (in German), in: Proc. GI-Fachtagung Datenbanken für Büro, Technik und Wissenschaft, Karlsruhe, IFB, Springer Verlag, 1985, pp. 253–286.

KKLW87 *Kemper, A., Küspert, K., Linnemann, V., Wallrath, M.:* Pascal Structures for HDBL Types: Layout, Naming Conventions, Storage Allocation and Usage in Functions, Technical Note, No. 87.05, IBM Deutschland, Wissenschaftliches Zentrum Heidelberg, Nov. 1987.

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

LKD87 *Linnemann, V., Küspert, K., Dadam, P., Pistor, P., Erbe, R., Kemper, A., Südkamp, N., Walch, G., Wallrath, M.:* Design and Implementation of an Extensible Database Management System Supporting User Defined Types and Functions, Technical Report, No. 87.12.011, IBM Deutschland, Wissenschaftliches Zentrum Heidelberg, Dez. 1987.

Lo85 *Lockemann, P. C., et al.:* Anforderungen technischer Anwendungen an Datenbanksysteme, in: Proc. GI-Fachtagung Datenbanken für Büro, Technik und Wissenschaft, Karlsruhe, IFB, Springer Verlag, 1985, S. 1–26.

Mi85 *Mitschang, B.:* Charakteristiken des Komplex-Objekt-Begriffs und Ansätze zu dessen Realisierung, in: Proc. of the GI-Conf. on „Database Systems for Office Engineering and Science Environments", 1985, S. 382–400.

Mi87 *Mitschang, B.:* MAD – ein Datenmodell für den Kern eines Non-Standard-Datenbanksystems, in: Proc. GI-Fachtagung „Datenbanksysteme in Büro, Technik und Wissenschaft", Darmstadt, 1985, IIFB, Springer Verlag, 1985, S. 180–195.

Mi88 *Mitschang, B.:* Towards a unified view to design data und knowledge representation, in Proc. 2nd Int. Conf. on Expert Database Systems, Tysons Corner VA, 1988 (Further Publication by Benjamin/Cummings Publ. Co.).

PSSWD87 *Paul, H.-B., Schek, H.-J., Scholl, M. H., Weikum, G., Deppisch, U.:* Architecture and Implementation of the Darmstadt Database Kernel System, SIGMOD87.

RS87 *Rowe, L. A., Stonebraker, M. R.:* The POSTGRES Data Model, in: Proc. 13th VLDB Conf., Brighton 1987, pp. 83–96.

SS86 *Schek, H. J., Scholl, M. H.:* The Relational Model with Relation-Valued Attributes, in: Information System, Vol. 11, No. 2, 1986, pp. 137–147.

Wi87 *Wiederhold, G., et al.:* Architectural Computer for Large Knowledge Bases, in: Proc. GWAI-87, 11th Workshop on Artificial Intelligence, Informatik Fachberichte, Springer Verlag, 1987, pp. 366–385.