Abbildung von Frames auf neuere Datenmodelle

Th. Härder, N. Mattos, B. Mitschang
Universität Kaiserslautern

Übersicht
Es wird die Abbildung von Frames mit ihren Modellierungskonzepten und charakteristischen Operationen auf objektorientierte Datenmodelle untersucht, um Wissensrepräsentation in sogenannten Non-Standard-Datenbanksystemen - beispielsweise für Expertensystem-Anwendungen - unterstützen zu können. Nach einem Vergleich der Eigenschaften von Relationenmodell, NF^2-Modell und MAD-Modell für diese Aufgabe wird eine Bewertung der verschiedenen Ansätze vorgenommen, um ihre Tauglichkeit für die Frame-Modellierung deutlicher herauszukristallisieren.

1. Einleitung

Praxistaugliche Expertensysteme (XPS) erfordern eine effiziente Verwaltung sehr großer Wissensbasen auf Sekundärspeicher, Mehrbenutzerfähigkeit und ein gewisses Maß an Fehlertoleranz. Diese Aspekte werden bereits in Datenbanksystemen (DBS) realisiert, die seit Jahren zur Verwaltung großer Datenmengen vor allem in kommerziellen Bereichen erfolgreich eingesetzt werden. Eine bloße Übernahme solcher konventioneller DBS für den Einsatz bei XPS-Anwendungen würde zumindest zu schwerfälliger Wissensmodellierung und erheblicher Leistungseinbuße führen, da herkömmliche Datenmodelle - ihre Datenstrukturen, Operationen und Konzepte zur Integritätssicherung - sowie ihre unterstützenden Maßnahmen in einer DBS-Implementierung nicht für solche Non-Standard-Anwendungen [HR85] entworfen wurden. Deshalb wird in der DB-Forschung momentan auf breiter Front die Frage untersucht, wie die Architektur und Implementierung künftiger DBS für Non-Standard-Anwendungen (NDBS) aussehen soll. Die DBS-Kern-Architektur, die als aussichtsreicher Lösungsvorschlag gilt, wird im Moment in mehreren Prototyp-Implementierungen erprobt [Da86,HMMS87,PSSWD87]. Für unsere Diskussion benutzen wir diese Architektur als Rahmenvorstellung, wie in Bild 1 gezeigt. Sie besteht grob gesprochen aus einem anwendungsunabhängigen Kern und einer Zusatzschicht - Modellabbildung genannt -, in der eine Anwendungsorientierung erreicht wird. Neben einer Reihe von Vorteilen [HR85] soll diese Architektur durch ihre Zweiteilung auch die Abbildung des NDBS auf Workstation und Server unterstützen.

Als Datenmodelle, die für den Kern in Frage kommen, stellt man sich solche vor, die bereits eine Objektorientierung aufweisen und damit die Abbildung komplexer Anwendungsobjekte erleichtern [BB84,Mi87,SS86]. Aufgabe der Modellabbildung ist die Bereitstellung von Anwendungsobjekten an der NDBS-Schnittstelle, deren Repräsentationsformen, Operationen und Integritätsbedingungen letztlich mit Hilfe der Primitive des Datenmodells realisiert werden müssen.

Die speziell auf das Anwendungsgebiet Expertensysteme zugeschnittene Modellabbildung - auch als Wissensadministrator bezeichnet - stellt ein Modellierungswerkzeug zur Unterstützung der XPS-Arbeit zur Verfügung. Dieses soll beispielsweise allgemeine Modellierungskonzepte umfassen, mit denen die verschiedenen Formen der Wissensrepräsentation leicht nachgebildet werden können. Es sollen daher zumindest die vier in vielen Wissensrepräsentationsformen implizit existierenden grundlegenden Abstraktionskonzepte unterstützt werden: Generalisierung (is-a), Klassifikation (instance-of), Aggregation (part-of) und Assoziation (member-of). Neben diesen Konzepten bietet der Wissensadministrator noch zusätzlich spezielle Objektdarstellungen und Operationen an. Für das Frame-Konzept [Mi75,FK85] sind beispielsweise Units, Slots und Aspekte als Objekte sowie die darauf definierten Operationen Einfügen eines Frames, Lesen von Slotwerten, Ausführung von Methoden usw. anzubieten. Die Aufgabe des Wissensadministrators besteht nun darin, diese Objekte und Operationen geeignet auf das Datenmodell des NDBS-Kerns abzubilden. Der Wahl des richtigen Datenmodells kommt hierbei eine entscheidende Bedeutung zu.

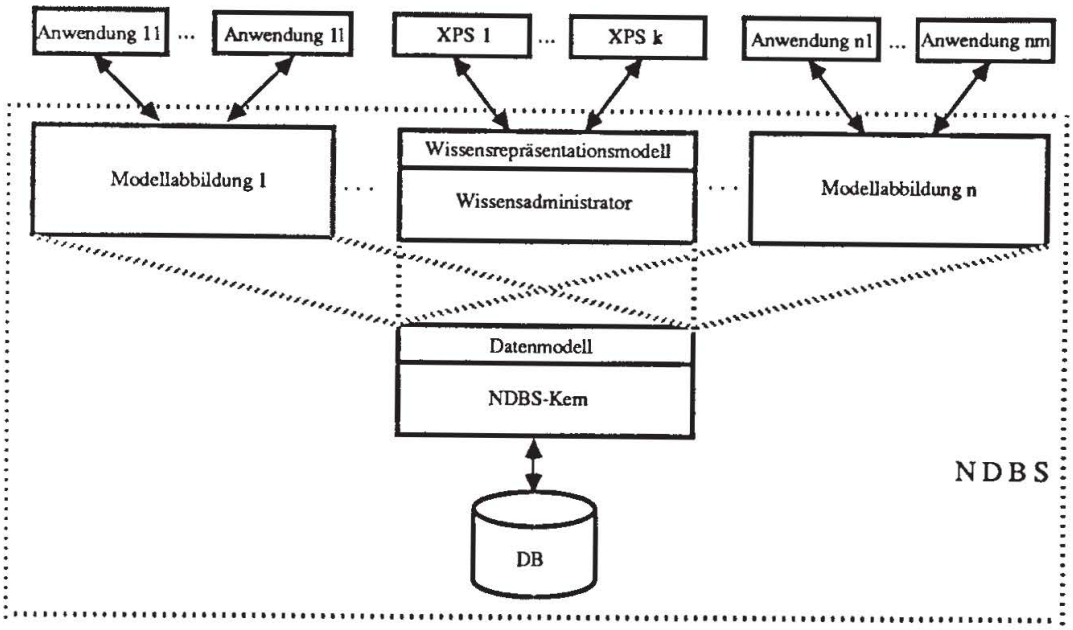

Bild 1: Grobarchitektur von DBS für Non-Standard-Anwendungen

Ziel dieser Arbeit ist, die Tauglichkeit von Datenmodellen zur Frame-Abbildung zu belegen. Es wird zunächst ein an [Mi75,FK85,BS83] angelehntes Frame-Modell vorgestellt. Anschließend wird eine vergleichende Betrachtung einiger Datenmodelle (Relationenmodell, NF^2-Modell [SS86] und das Molekül-Atom-Datenmodell (MAD-Modell) [Mi87]) hinsichtlich ihrer Modellierungs- und Manipulationsmöglichkeiten zur Abbildung des eingeführten Frame-Modells durchgeführt.

2. Das Frame-Modell

Im Bereich der XPS stellt die Repräsentation von Wissen, bestehend aus Daten und Programmen, den Versuch dar, die Vorgehensweise von Spezialisten (Experten) bei der Lösung von Problemen zu formalisieren. Je nachdem, ob man Wissensinhalte für einen passiv-deklarativen Gebrauch oder für einen aktiv-prozeduralen Gebrauch beschreibt, gelangt man zu verschiedenen Formen der Wissensrepräsentation - deklarativ oder prozedural [Ra82]. Viele der verschiedenen Wissensrepräsentationen enthalten jedoch sowohl deklarative als auch prozedurale Aspekte. Dies gilt auch für Frames [Mi75,FK85], mit denen in meist deklarativer Weise stereotypische Situationen und Objekte beschrieben werden [Pu86]. In dieser Arbeit werden die prozeduralen Aspekte von Frames (Programme, Methoden und Regelmengen) nicht berücksichtigt, da es hier mehr um die Abbildung des deklarativen Bestandteils von Frames geht. Daher beschränken wir uns in diesem Kapitel auf die Beschreibung eines Frame-Modells unter dem Aspekt der Repräsentation von deklarativem Wissen.

Dieses Frame-Modell (siehe Tabelle 1) kennt zunächst nur Objekte (hier als Unit bezeichnet), die die Grundelemente für die Wissensrepräsentation sind. Eine Unit setzt sich aus einer Liste von Attributen (Slots) mit Attributwerten (Werte) zusammen (Konzept der Aggregation) und wird durch einen Unit-Namen eindeutig identifiziert. Jedem Slot ist eine Liste von Aspekten mit den zugehörigen Aspektwerten zugeordnet, die dazu dienen, den Slot mit seinem Wert genauer zu spezifizieren (Bild 2). Mögliche Aspekte sind beispielsweise "Comment", den der Benutzer als Kommentar verwenden kann, "Default" zur Definition eines Standardwerts im Fall eines undefinierten Slotwerts und "Wertmenge" zur Festlegung einer Menge von Werten, aus denen der Wert des Slots ausgewählt werden muß.

Die Units entsprechen den Objekten (auch Entities genannt) eines Ausschnitts der realen Welt, die zu modellieren sind. Dabei unterscheidet man zwischen Units, die eine Menge von Entities repräsentieren und daher als Klasse bezeichnet werden, und Units, die Entities einer Klasse repräsentieren und daher als Member einer Klasse bezeichnet werden. Die Unterscheidung zwischen Klassen, die den Ausprägungstyp von Member-Objekten (d.h. Instances) darstellen - entsprechend dem Abstraktionskonzept der Klassifikation -, von denen, die eine Menge von Objekten im streng mengentheoretischen Sinn beschreiben - entsprechend dem Abstraktionskonzept der Assoziation - wird jedoch nicht gemacht. Ebenso unterscheidet man nicht zwischen Member, die Instances eines Ausprägungstyps darstellen, von denen, die die Elemente einer Menge repräsentieren. Daher entspricht die Unterteilung der Units (Objekte) in Klassen und Member sowohl dem Konzept der Assoziation als auch der Klassifikation, mittels denen sogenannte Member-Objekte über eine "member-of" Relation mit Klassenobjekten in Beziehung gesetzt werden. Weiterhin ist es möglich, daß eine Unit sowohl Member als auch Klasse sein kann bzw., daß eine Unit gleichzeitig Member verschiedener Klassen sein kann. Neben dieser Beziehung zu ihren Member haben Klassen Beziehungen mit anderen Klassen. Diese entsprechen dem Abstraktionskonzept der Generalisierung, durch das eine Hierarchie von Objektklassen gebildet wird, zwischen denen eine "is-a" Relation existiert.

Diese Hierarchie macht die wichtige Bedeutung der Unit-Beziehungen deutlich: Informationen über Units werden entlang dieser Hierarchie transportiert. Slots (Eigenschaften) von übergeordneten Units können an untergeordnete Units übertragen werden. Man unterscheidet dabei zwischen zwei Arten von Slots: "Klassen-Slots" und "Member-Slots". Klassen-Slots bezeichnen Eigenschaften der Unit, während Member-Slots die Eigenschaften der zugehörigen Member beschreiben. Daher werden die Member-Slots an die untergeordneten Units übertragen. Da sie zur Spezifikation der Member-Eigenschaften dienen, besitzen sie keinen Wert. Erst wenn der Slot an die Member-Units vererbt wird, wird ihm ein Wert zugeordnet (siehe Bild 4). Unter Umständen ist es jedoch sinnvoll einen Standard-Wert zu definieren, der generell weitervererbt werden soll. Dies wird mit Hilfe des Default-Aspekts modelliert. Klassen-Slots werden dagegen nicht übertragen. Sie spezifizieren entweder Eigenschaften, die für alle Member der Klasse gelten, oder Eigenschaften, die nur für die Klasse relevant sind. Die Zuordnung eines Wertes ist deshalb notwendig. Bei der Übertragung von Slots an untergeordnete Units, auch Vererbung von Eigenschaften genannt, wird dann wie folgt unterschieden: Member-Slots werden entlang der Klasse/Klasse-Beziehungen an alle untergeordneten Klassen (Units) als Member-Slots und entlang der Klasse/Member-Beziehungen an die untergeordneten Member (Units) als Klassen-Slots vererbt; Klassen-Slots werden überhaupt nicht vererbt. Diese Unterscheidung macht deshalb die Zuordnung sowohl eines Slot-Typs (Klassen-Slot, Member-Slot) als auch einer Kennung (eigener Slot, geerbter Slot) zu jedem Slot erforderlich. Um diese Klasse/Member- bzw. Klasse/Klasse-Beziehungen modellieren zu können, existieren spezielle Slots, die für jede Unit definiert sind. Die Slots ist_subklasse_von und hat_als_subklassen werden dazu benutzt die Klasse/Klasse-Beziehung zu modellieren, wobei ist_subklasse_von die "Super"-Klassen einer Klasse und hat_als_subklassen die "Sub"-Klassen einer Klasse referenziert. ist_member_von und hat_als_member stellen die Klasse/Member-Beziehungen dar; ist_member_von spezifiziert die Referenz zu den Klassen bei denen die Unit ein Member ist und hat_als_member gibt die Referenz zu allen Member dieser Klasse an.

Unitname			
ist_subklasse_von:		Liste von Units	
hat_als_subklassen:		Liste von Units	
ist_member_von:		Liste von Units	
hat_als_member:		Liste von Units	
Slot 1:	Typ1	Kennung 1	Wert 1
		Aspekt 11	Aspektwert 11
		Aspekt 12	Aspektwert 12
		Aspekt 1m	Aspektwert 1m
Slot n:	Typn	Kennung n	Wert n
		Aspekt n1	Aspektwert n1
		Aspekt n2	Aspektwert n2
		Aspekt nm	Aspektwert nm

Bild 2: Struktur der Units

Bild 2 zeigt die komplette Struktur einer Unit. Es wird deutlich, daß die vier oben angesprochenen Slots auf Grund ihrer speziellen Bedeutung, nämlich der Modellierung der Unit-Beziehungen, aus der Menge der übrigen Slots herausgehoben sind. Die zugehörigen primitiven Operationen des Frame-Modells umfassen einerseits allgemeine objektbezogene Operationen wie das Einspeichern, Löschen, Lesen und Modifizieren von Units. Andererseits existieren Operationen, die Slots und Aspekte direkt manipulieren.

Tabelle 1:
Einige Aspekte des Frame-Modells

Sloteigenschaften	
Typ	Klassen-Slot / Member-Slot
Kennung	eigener / geerbter
Aspekte	Default, Comment, Wertmenge

Modellierung	
von	mittels
Generalisierung	ist_subklasse_von / hat_als_subklassen
Assoziation	ist_member_von / hat_als_member, Klassen-Slot
Klassifikation	ist_member_von / hat_als_member, Member-Slot
Aggregation	nur von Aspekten / zu Slots / zu Units

3. Datenmodelle zur Wissensrepräsentation

Im folgenden werden verschiedene Datenmodelle für die Abbildung des eingeführten Frame-Modells verwendet. Damit soll die Adäquatheit bzw. Unzulänglichkeit dieser Datenmodelle zur Wissensrepräsentation aufgezeigt werden. Zur Vereinfachung wird vorab eine Zwischenabbildung mit Hilfe des Entity- Relationship-Modells (ER-Modell) durchgeführt.

3.1 Eine Entity-Relationship-Modellierung des Frame-Modells

Das ER-Modell [Ch76] dient zur Strukturierung der in einem Weltausschnitt enthaltenen Information. Die Gegenstände dieses Weltausschnitts werden auf "Entities", ihre Beziehungen auf "Relationships" abgebildet. Die Eigenschaften dieser Gegenstände und ihrer Beziehungen können durch Entity- und Relationship-Attribute dargestellt werden. Gleichartige Gegenstände bzw. damit assoziierte Entities werden zu sog. "Entity-Mengen" (Typen) zusammengefaßt; gleiches führt zu den sog. Relationship-Typen. Bei den Beziehungen unterscheidet man drei Arten: (1:1), (1:n) und (n:m); damit werden eindeutige, funktionale und komplexe Beziehungen zwischen Entities charakterisiert. Zur graphischen Darstellung des ER-Modells werden i.a. ER-Diagramme (Bild 3) verwendet.

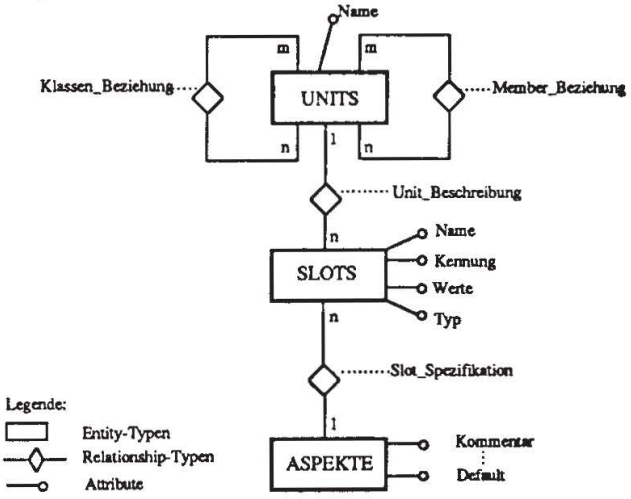

Bild 3:
ER-Diagramm des Frame-Modells

Zur Darstellung der Objekte des Frame-Modells scheint es auf den ersten Blick sinnvoll, für jede Klasse des abzubildenden Weltausschnitts einen Entity-Typ zu definieren. Diese Modellierungsweise offenbart allerdings größere Probleme: zum einen kann ein Objekt sowohl Member wie auch Klasse sein (d.h., ein Objekt ist als Typ und auch als Ausprägung zu modellieren); zum anderen kann es gleichzeitig Member von unterschiedlichen Klassen sein (d.h., ein Objekt ist gleich-

zeitig Ausprägung von verschiedenen Entity-Typen). Will man diese Abbildungsprobleme lösen, so muß man Redundanzen einführen, die allerdings auch ein erhebliches Maß an Overhead bedeuten.

Betrachtet man das Frame-Konzept jedoch etwas näher, so erkennt man, daß es von einem semantisch höheren Standpunkt aus nur abstrakte Objekte, die sog. Units, gibt. Deren Bedeutung als Klassen bzw. Member wird eigentlich nur durch die Beziehungen zwischen den Units realisiert und nicht durch die Units selbst. Eine Modellierung des Frame-Konzepts auf dieser abstrakteren Ebene vermeidet die oben beschriebenen Probleme. Man hat dabei einen Entity-Typ UNITS, der dazu benutzt wird, alle Objekte (Units) des zu modellierenden Weltausschnitts darzustellen. Alle diese Objekte, egal ob sie Member oder Klassen sind, werden dann als Ausprägung dieses Entity-Typs repräsentiert. Man erhält dadurch ein sehr generisches Schema, das es erlaubt, beliebige Frame-Anwendungen geeignet darzustellen.

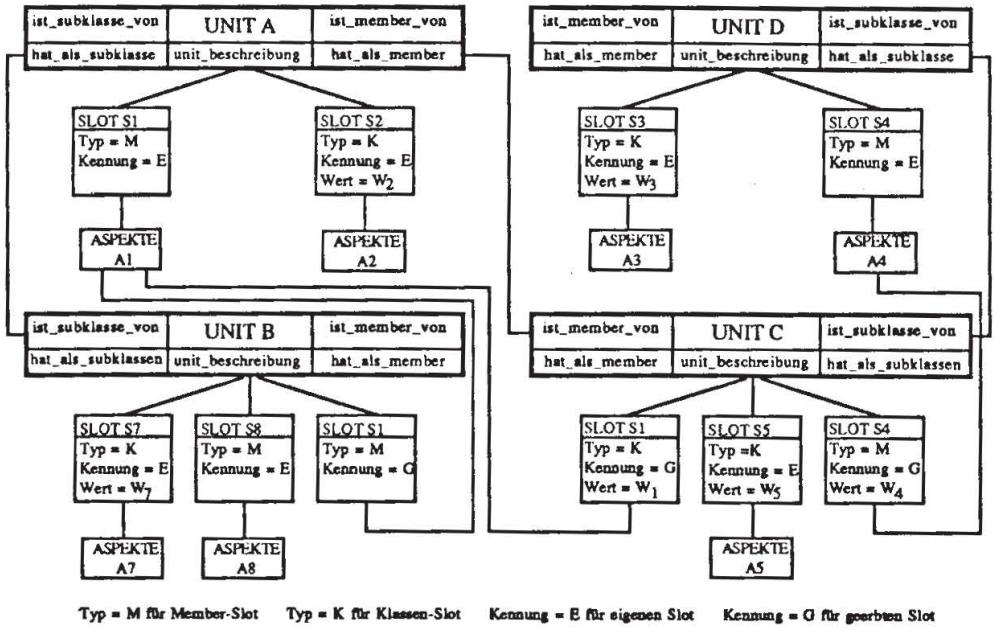

Bild 4: Ausprägungsbeispiel des Frame-Modells

In Bild 3 ist das resultierende ER-Diagramm der angesprochenen Frame-Modellierung dargestellt. Bild 4 zeigt ein dazu passendes Ausprägungsbeispiel. Die Semantik der Units (nämlich Member- und/oder Klassen-Unit) wird durch die beiden (n:m)-Beziehungen definiert: Die Klassen_Beziehung (mit den Teilbeziehungen hat_als_subklassen und ist_subklasse_von) baut die Generalisierungshierarchie und die Member_Beziehung (mit den Teilbeziehungen ist_member_von und hat_als_member) die Klassifikations- und Assoziationsstrukturen auf. Über die Beziehung Unit_Beschreibung werden die einzelnen Slotdaten zur Beschreibung der Units aggregiert (Aggregation). Ebenso führt die Beziehung Slot_Spezifikation eine aggregierende Beschreibung um die Aspektdaten des zugehörigen Slots durch. Dieser Entity-Typ realisiert eine generische Wertebereichs- bzw. Typdefinition der zugehörigen Slotwerte.

3.2 Modellierung unter Verwendung des Relationenmodells und des NF^2-Modells

Das Relationenmodell stellt für die Darstellung von Entity- und Relationship-Mengen nur ein Konstrukt - die Relation (Tabelle) - zur Verfügung. Daher entspricht ein Tabellenaufbau im wesentlichen dem Entity- bzw. Relationship-Typ, die Spalten dessen Attributbeschreibung und die Zeilen (auch als Tupel bezeichnet) konkreten Entities oder Beziehungen. Eine wesentliche Eigenschaft des Relationenmodells ist, daß alle Informationen - auch die Beziehungen zwischen Tupeln - explizit durch Attributwerte der Tupel ausgedrückt werden.

Die Modellierung des Frame-Modells mittels des Relationenmodells (RM) sieht dann wie folgt aus: Jedem Entity-Typ wird eine Relationenbeschreibung zugeordnet; die Attribute werden dabei übernommen. Die (1:1)- und (1:n)-Beziehungen werden durch entsprechende Attributwerte (Primär- und Fremdschlüssel) dargestellt, wohingegen alle (n:m)-Beziehungen über Hilfsrelationen beschrieben werden. Die im RM definierten Operationen erlauben ein deskriptives Ansprechen und Manipulieren einzelner Tupel bzw. Tupelmengen sowie eine mächtige tupelübergreifende Operation (Join-Operation). Modellierbar und ansprechbar sind nur homogene Tupelmengen. Daher führt jede Join-Operation eine Zusammenlegung der beiden beteiligten Tabellen durch, wobei allerdings auch Strukturinformation verloren geht. Der Versuch eine vollständige Unit-Aggregation aufzubauen, führt zu einer flachen Ergebnisrelation (Unit-, Slot- und Aspektdaten sind nebeneinandergereiht und nicht aggregiert) aufgrund der zu verwendenden Join-Operationen. Noch schwieriger stellen sich die Operationen auf der Generalisierungshierarchie dar: z.B. erfordert die Vererbung eines Slot-Eintrags wegen der Abwesenheit von Rekursion das explizite, iterative Berechnen der Unit-Hierarchie. Insgesamt sind alle in unserem Frame-Modell enthaltenen Abstraktionskonzepte durch die Modellierung verloren gegangen und können durch die angebotenen Operationen auch nicht wiedergewonnen werden. Das bedeutet, daß die Abstraktionshierarchien sowie die Assoziations- und Aggregationseigenschaften durch externe Programmierung aufgebaut werden müssen. Dies bedeutet einen erheblichen Effizienzverlust, der aufgrund nicht adäquater Modellierungs- und Manipulationsmöglichkeiten entstanden ist.

Das NF^2-Modell (non-first normal form, [SS86]) ist eine Erweiterung des RM um relationenwertige, also nicht-atomare Attribute. Dadurch, daß damit die Attribute wiederum Relationen sein dürfen, wird die Kombination von Aggregation und Assoziation direkt unterstützt - die dem Modell inhärente "Schachtelung" von Relationen "materialisiert" sozusagen Aggregation und auch Assoziation. Durch unterschiedliche Anwendung dieses Schachtelungskonzepts können sehr verschiedene Frame-Modellierungen erzeugt werden: Verzichtet man gänzlich auf die Relationenschachtelung, so erhält man eine ähnliche Modellierung wie mit dem RM und damit einhergehend auch alle o.a. Nachteile. Zur Aggregation aller Unit-Informationen muß eine NF^2-Tabelle bestehend aus den ineinandergeschachtelten Relationenbeschreibungen von Units, Slots und Aspekten definiert werden. Diese Aggregation erzeugt eine Datenredundanz bzgl. der Aspektdaten, welche ein gewisses Maß an Verwaltungsoverhead zur Redundanzkontrolle verursacht. Unterstellt man, was sicherlich plausibel erscheint, daß die Änderungshäufigkeiten auf den Aspektdaten ziemlich gering sind, so kann der Aufwand zur Redundanzkontrolle vernachlässigt werden. Das Schachtelungskonzept des NF^2-Modells kann in diesem Bereich also gewinnbringend eingesetzt werden. Zur Verarbeitung dieser geschachtelten Tabellen wird vom NF^2-Modell ein mächtiger Operationsvorrat zur Verfügung gestellt, den man durch die geschachtelte Anwendung der bisherigen Relationenalgebra erhält. Versucht man jedoch auch die typmäßig rekursiven Netzstrukturen der Generalisierung sowie der Klassifikation und Assoziation (d.h. Klassen- und Memberbeziehung) ebenfalls mit Hilfe des Schachtelungskonzepts abzubilden, so stößt man sowohl auf modellierungstechnische als auch auf operationale Schwierigkeiten: rekursive Strukturen sind im NF^2-Modell nicht darstellbar und müssen deshalb "flachgeklopft" werden; netzwerkartige Strukturen können entweder analog zum RM über Primär-/Fremdschlüsselbeziehungen oder teilweise redundant mit Hilfe des Schachtelungskonzepts (siehe Aspektmodellierung) dargestellt werden. Die vom NF^2-Modell angebotenen Operationen erlauben nicht den nachträglichen Aufbau der gewünschten typmäßig rekursiven Netzwerkstrukturen. Aktuelle Forschungsarbeiten [Li87] sehen nur eine Erweiterung der NF^2-Anfragesprache auf rekursive Anfragen, allerdings keine Erweiterung des Datenmodells vor.

Zusammenfassend läßt sich sagen, daß das RM deutliche Schwächen bei der Abbildung des Frame-Modells sowie der dort enthaltenen Abstraktionskonzepte offenbart. Im Gegensatz dazu erlaubt das NF^2-Modell aufgrund seines Schachtelungskonzepts hier zumindest die direkte Abbildung der Aggregation, wobei allerdings häufig ein gewisses Maß an Datenredundanz erzeugt wird, welche es zu kontrollieren gilt (Zusatzaufwand). Zur Abbildung der Operationen des Frame-Modells

wird in beiden Fällen wenig Unterstützung angeboten. Die durch die Abstraktionsstrukturen definierten typmäßig rekursiven Netzstrukturen müssen extern, d.h. im Anwendungsprogramm, aufgebaut werden. Ebenso sind die auszuführenden Manipulationen auch extern durchzuführen bevor die aktualisierten Einzelteile wieder zurückgeschrieben werden.

3.3 Modellierung mit Hilfe des Molekül-Atom-Datenmodells

Beim Molekül-Atom-Datenmodell (MAD-Modell, [Mi87]) handelt es sich ebenfalls um eine RM-Erweiterung, dessen Ziel die konsistente Erweiterung der Verarbeitung von homogenen zu heterogenen Satzmengen ist (sprich: die Erweiterung von der bisherigen Tupelverarbeitung zur Molekülverarbeitung).

Die Grundelemente des MAD-Modells sind die Atomtypen. Sie sind vergleichbar mit den Relationen des Relationenmodells, d.h., sie fassen Attribute verschiedener Typen zusammen. Aus diesen Atomtypen können nun dynamisch 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 symmetrisch (d.h. 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 geforderten 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 noch 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 nun einfach zur Realisierung der Abstraktionskonzepte verwendet werden: Aggregation, Assoziation, Generalisierung und Klassifikation (inklusive der netzwerkartigen Strukturen) werden direkt durch das Bindungskonzept, d.h. durch die Molekülbildung, unterstützt. Moleküle bzw. Molekültypen können damit vom Wissensadministrator so definiert werden, daß die Verarbeitung der Objekte des Frame-Modells möglichst effektiv abgewickelt werden kann.

Die Abbildung des Frame-Modells aus Kapitel 2 vereinfacht sich hier zu einer direkten Umsetzung des Entity-Relationship-Schemas in das äquivalente MAD-Schema. Jedem Entity-Typ wird ein Atomtyp zugeordnet und jeder Relationship-Typ wird als Beziehungstyp mit seinen beiden Teilbeziehungen dargestellt. Aufgrund dieser direkten Abbildbarkeit der Konzepte kann das Diagramm aus Bild 3 auch als MAD-Schema interpretiert werden; Bild 4 zeigt demnach ein Ausprägungsbeispiel des MAD-Schemas, d.h. Atome und deren konkrete Beziehungen (Teilbeziehungen sind benannt).

Die Datenmanipulation des MAD-Modells (DML) ist damit "molekül-mengenorientiert", d.h., sie erlaubt die Verarbeitung einer Molekülmenge, wobei jedes Molekül wiederum aus einer Menge von Atomen potentiell verschiedener Typen besteht. Der Typ der Moleküle entspricht dabei entweder einem bereits vordefinierten Molekültyp oder er wird dynamisch in der Manipulationsanweisung festgelegt. 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 FROM-Klausel (Spezifikation der für die Anweisung relevanten Molekültypen), der WHERE-Klausel (Spezifikation der Restriktionen/Qualifikationsbedingungen) und der SELECT-Klausel (Spezifikation der Projektionen).

Im folgenden wird exemplarisch für einige ausgewählte Frame-Operationen deren Umsetzung mit Hilfe der MAD-DML aufgezeigt. Bild 5 enthält die DML-Anweisungen sowie die zugehörigen Moleküldarstellungen zur näheren Erläuterung. Beispiel 1 zeigt das "Lesen einer Unit". Diese Frame-Operation umfaßt das Holen der Unit-Beschreibung inklusive der zugehörigen Slot- und Aspektdaten und kann in einer einzigen DML-Anweisung ausgedrückt werden. Die Molekül-definition innerhalb der FROM-Klausel umfaßt die zu aggregierenden Atomtypen (UNITS, SLOTS und ASPEKTE) und die entsprechenden Beziehungstypen (unit_beschreibung, slot_spezifikation), die die Aggregation spezifizieren. Jeder definierte Molekültyp ist ein zusammenhängender Teilausschnitt oder Teilgraph des DB-Schemas, dessen Knoten die Atomtypen und dessen Kanten die Beziehungstypen darstellen. Die Qualifikationsbedingung der WHERE-Klausel beschränkt die Ergebnismengen auf die Moleküle, die den Namen 'Eulen' besitzen. Der Ergebnisbereich der spezifizierten Projektion wird im Moleküldiagramm immer gestrichelt umrahmt. Das "Einfügen eines Member-Slots" ist eine sehr umfangreiche Operation, die in vier DML-Anweisungen aufzugliedern ist (Beispiel 2). Die erste Teiloperation fügt den Slot inkl. der zugehörigen Aspektdaten als Member-Slot bei der entsprechenden Unit ein. Dieses Einfügen umfaßt das Eintragen neuer Atome in die Atomtypen SLOTS und ASPEKTE sowie die Zuordnung zur entsprechenden Unit. Der Molekültyp der INSERT-Operation besteht aus dem aggregierten Unit-Molekül und legt die "Umgebung" fest, in die das zu speichernde Slot-Molekül eingefügt wird. Die zweite Teiloperation liefert den IDENTIFIER-Wert des neu eingefügten ASPEKTE-Atoms. Dieser Wert wird von Teiloperation drei und vier für das Attribut slot_spezifikation benötigt, um dort die Referenz auf das bereits gespeicherte ASPEKTE-Atom zu setzen. Hierdurch wird die Mehrfachbenutzung der Aspekt-daten gewährleistet. Die zugehörige Anfrage zeigt eine wichtige Eigenschaft der MAD-DML auf: die qualifizierte Projek-tion (Anfrageanweisung innerhalb der SELECT-Klausel) erlaubt ein wertabhängiges Projezieren; hier werden nur die Slots

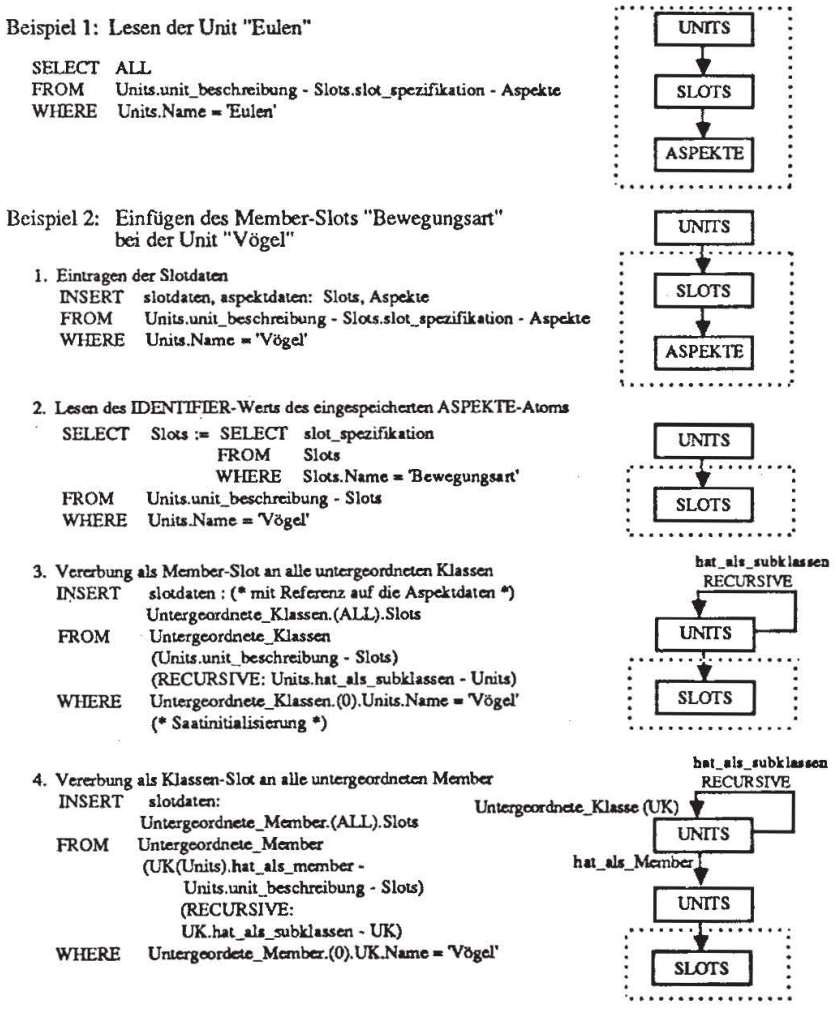

Bild 5: Umsetzung von FRAME-Operationen mittels der MAD-DML

in das Ergebnis aufgenommen, deren Name 'Bewegungsart' ist. Die dritte Teiloperation "vererbt" den Slot als Member-Slot an die untergeordneten Klassen. Hier wird ausgenutzt, daß das MAD-Modell eine rekursive Molekülbildung erlaubt. Die Subklassenstruktur wird durch die RECURSIVE-Klausel spezifiziert und im Sinne eines "transitiven Hülle"-Operators ausgewertet (im Relationen- und NF^2-Modell mußte dies extern programmiert werden). Ausgehend von der in der WHERE-Klausel qualifizierten Wurzel des Rekursivmoleküls werden in einer Rekursionsschleife alle untergeordneten Subklassen-Units über die 'hat_als_subklassen'-Teilbeziehung aufgesucht und deren Slots um den neuen Member-Slot ergänzt. Die vierte Teiloperation "vererbt" den Slot als Klassen-Slot an alle untergeordneten Member sowie an alle Member der untergeordneten Klassen. Die Umsetzung dieser Teiloperation geschieht dabei analog oben.

Auf eine ausführliche Diskussion von allen Frame-Operationen (siehe [Ma86]) muß hier aus Platzgründen verzichtet werden. Die Eignung des MAD-Modells sowohl hinsichtlich der angebotenen Modellierungskonzepte als auch hinsichtlich der verfügbaren Manipulationsmöglichkeiten kommt trotzdem sehr deutlich zum Vorschein.

3.4 Vergleichende Betrachtung

Eine Zusammenfassung der beiden vorigen Abschnitte wird in Tabelle 2 in Form einer vergleichenden Betrachtung der drei Datenmodelle hinsichtlich ihrer Eignung zur Frame-Modellierung aufgezeigt. Diese Gegenüberstellung soll zum Ausdruck bringen, wie einfach bzw. umständlich es ist, die Strukturen des Frame-Modells nachzubilden. Die Schwächen des RM kommen dabei sehr deutlich zum Vorschein. Die im RM enthaltenen Modellierungskonzepte reichen keinesfalls aus: sämtliche Abstraktionsstrukturen können zwar modelliert werden, müssen aber im Anwendungsprogramm explizit aufgebaut werden. Im NF^2-Modell können durch das Konzept der Relationenschachtelung einige Aspekte der Abstraktionsstrukturen bereits im Datenmodell integriert werden. Allerdings sind die Abstraktionshierarchien der Generalisierung und Assoziation analog zum RM wiederum explizit im Anwendungsprogramm zu realisieren. Das MAD-Modell bietet mit dem allgemeinen Konzept der Molekülbildung die Integration von Netzstrukturen und auch rekursiven Strukturen. Damit können dann alle Abstraktionsstrukturen in einfacher Weise dargestellt werden. Deren jeweilige semantische Interpretation bleibt natürlich den Ebenen oberhalb des Datenmodells (im Wissensadministrator) vorbehalten. Von der Datenmodellseite können immer nur allgemeine Abbildungskonzepte (z.B. Schachtelung oder Molekülbildung) angeboten werden.

Datenmodelle / Vergleichskriterien	RM	NF^2-Modell	MAD-Modell
Netzstrukturen	Nachbildung von (n:m)-Beziehungen über Verknüpfungsrelationen	Nachbildung teilweise durch Schachtelungskonzept unterstützt (ggf. Verzicht auf Symmetrie; Kontrolle etwaiger Redundanz)	darstellbar durch Modellkonstrukte
Rekursion	Modellierung flachgeklopft; Aufbau im AP	Modellierung flachgeklopft; Aufbau im AP	darstellbar durch Modellkonstrukte
Klassifikation	entspricht der Unterscheidung von Typ/Schema und Ausprägung	entspricht der Unterscheidung von Typ/Schema und Ausprägung	entspricht der Unterscheidung von Typ/Schema und Ausprägung
Aggregationsstrukturen	nur auf Attributebene; Aufbau höherer Strukturen im AP	durch Schachtelungskonzept unterstützt	durch Molekülbildung unterstützt
Assoziationsstrukturen	keine direkte Unterstützung; Aufbau der Strukturen im AP	durch Schachtelungskonzept unterstützt *	durch Molekülbildung unterstützt
Generalisierungsstrukturen	keine direkte Unterstützung; Aufbau der Strukturen im AP	keine direkte Unterstützung	durch Molekülbildung unterstützt

* rekursiv aufgebaute Hierarchien werden nicht unterstützt, d.h., diese sind im Anwendungsprogramm (AP) zu realisieren

Tabelle 2: Datenmodellvergleich

4. Zusammenfassung

Hier wurde die Abbildung von Frames mit ihren Modellierungskonzepten und charakteristischen Operationen auf objektorientierte Datenmodelle untersucht. Die Tauglichkeit der verschiedenen Ansätze ist in entscheidendem Maße von den verfügbaren Konzepten zur Modellierung und Verarbeitung der Abstraktionsstrukturen von Klassifikation, Aggregation, Assoziation und Generalisierung abhängig. Das Molekül-Atom-Datenmodell bietet, im Vergleich zu anderen untersuchten Datenmodellen, hierfür allgemeine Modellierungskonzepte an, die es erlauben

- Netzstrukturen direkt und symmetrisch darzustellen,
- rekursive Strukturen abzubilden und
- die Abstraktionsstrukturen mit dem Konzept der dynamischen Molekülbildung direkt zu unterstützen.

Durch die zugehörigen Datenmodelloperationen können die modellierten Strukturen entsprechend verarbeitet werden.

Die Ergebnisse dieser Arbeit sind auch übertragbar auf andere Wissensrepräsentationsmodelle, die ebenfalls die genannten Abstraktionskonzepte enthalten (etwa die Klasse der Semantischen Netze).

Literaturverzeichnis

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

[BS83] Bobrow, D., Stefik, M.: The LOOPS Manual, Palo Alto, California, Xerox, 1983.

[Ch76] Chen, P.P.: The Entity-Relationship-Model - Toward a Unified View of Data, in: ACM TODS, Vol. 1, No. 1, 1976, pp. 9-36.

[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.

[FK85] Fikes, R., Kehler, T.: The Role of Frame-based Representation in Reasoning, in: Communications of the ACM, Vol. 28, No. 9, Sept. 1985, S. 904-920.

[HMMS87] Härder, T., Meyer-Wegener, K., Mitschang, B., Sikeler, A.: PRIMA - a DBMS Prototype Supporting Engineering Applications, in: Proc. Int. Conf. on VLDB, Brighton, 1987.

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

[Li87] Linnemann, V.: Non-First Normal Form Relations and Recursive Queries: An SQL-Based Approach, in: Proc. 3rd Int. Conf. on Data Engineering, Los Angeles, CA, 1987.

[Ma86] Mattos, N.: Modellierung von FRAME-Konzepten mit dem MAD-Modell, Interner Bericht 164/86 des FB Informatik der Univ. Kaiserslautern, 1986.

[Mi87] Mitschang, B.: MAD - ein Datenmodell für den Kern eines Non-Standard-Datenbanksystems, in: Proc. GI-Fachtagung "Datenbanksysteme in Büro, Technik und WIssenschaft", Darmstadt, 1987, S. 180-195.

[Mi75] Minsky, M.: A Framework for Representing Knowledge, in: The Psychology of Computer Vision (editor: Winston, P.), McGraw-Hill Book Company, 1975.

[PSSWD87] Paul, H.-B., Schek, H.-J., Scholl, M.H., Weikum, G., Deppisch, U.: Architecture and Implementation of the Darmstadt Database Kernel System, in: Proc. SIGMOD, San Francisco, 1987.

[Pu86] Puppe, F.: Expertensysteme - Übersicht und Exemplarische Einführung, in: Informatik-Spektrum, Vol. 9, No. 1, Februar 1986, S. 1-13.

[Ra82] Raulefs, P.: Expertensysteme, in: Künstliche Intelligenz Frühjahrsschule, Teisendorf, März 1982, S. 61-98.

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