Sinz, Elmar J.; Amberg, Michael

Objektorientierte Datenbanksysteme aus der Sicht der Wirtschaftsinformatik

Objektorientierte Datenbanksysteme aus der Sicht der Wirtschaftsinformatik

Elmar J. Sinz, Michael Amberg*

1 Einführung

Objektoriente Datenbanksysteme (ooDBS) werden in der Datenbankforschung seit mehreren Jahren intensiv diskutiert. Inzwischen haben eine Reihe von ooDBS das Forschungs- und Prototyp-Stadium verlassen und sind als Produkte verfügbar. Dies macht ooDBS auch für Anwendungen der Wirtschaftsinformatik interessant. Aus diesem Blickwinkel heraus werden ooDBS im folgenden betrachtet: Was sind ooDBS? Was leisten sie im Vergleich zu relationalen Datenbanksystemen? Wo liegen Nutzen und Defizite?

Nach übereinstimmender Auffassung werden ooDBS wie folgt abgegrenzt: Ein **ooDBS** ist ein **Datenbanksystem**, das auf **einem objektorientierten Datenbankmodell** beruht [1], [2], [3], [7]. Auf eine weitere Differenzierung von ooDBS in ein objektorientiertes Datenbankverwaltungssystem (ooDBVS) und eine objektorientierte Datenbank (ooDBS) wird im folgenden verzichtet.

2 Objektmodelle

Gemäß obiger Abgrenzung ist eines der zentralen Merkmale von ooDBS die Unterstützung eines objektorientierten Datenbankmodells (kurz: **Objektmodell**). Ein Objektmodell ist ein Meta-Modell, in dem das Schema einer objektorientierten Datenbank (ooDB-Schema) beschrieben wird.

Historisch gesehen entstanden ooDBS entweder aus der Erweiterung bestehender Datenbanksysteme (DBS) um objektorientierte Konzepte oder aus der Erweiterung objektorientierter Programmiersprachen um Datenbankkonzepte. Aufgrund dieser Entstehungsgeschichte gibt es im Gegensatz zum Relationenmodell kein standardisiertes Objektmodell und keine allgemein akzeptierte theoretische Fundierung von Objektmodellen (vgl. [5]). Die Erläuterung von Objektmodellen erfolgt daher anhand repräsentativer Struktur- und Verhaltensmerkmale eines ooDBS-Schemas (vgl. [8]).

Im Mittelpunkt der Definition eines ooDB-Schemas steht eine Menge von **Klassendefinitionen,** bestehend aus je einem Klassennamen sowie aus Spezifikationen der Struktur und des Verhaltens der Klasse. Eine **Klasse** besteht aus der Klassendefinition und einer Menge von **Objekten (Instanzen).**

Zur Spezifikation der Struktur von Klassen dienen folgende Eigenschaften:

- Jede Klassendefinition ist an einen bestimmten Datenobjekttyp gebunden. Das zugehörige **Typsystem** stellt z.B. die Basistypen *integer, real, string, boolean* zur Definition **elementarer Attribute** sowie Konstruktoren, z.B. *tuple, set, list,* zur Bildung **komplexer Attribute** bereit. Auf diese Weise sind Klassen mit komplexen Objekten beschreibbar.
- Über den Klassendefinitionen existiert eine partielle Ordnung ***isa***, die Paare von Klassendefinitionen in der Form (**Superklasse, Subklasse**) ordnet. Die Superklasse **vererbt** ihre **Attribute** an die Subklasse,

*) Prof. Dr. Elmar J. Sinz, Dipl.-Inf. Michael Amberg, Otto-Friedrich-Universität Bamberg, Lehrstuhl für Wirtschaftsinformatik, insbesondere Systementwicklung und Datenbankanwendung, Feldkirchenstraße 21, D-8600 Bamberg, Tel. (09 51) 863-84 78/84 96, Fax (09 51) 3 96 36, X.400: {sinz|amberg} @ sowi.uni-bamberg.dbp.de

wobei die Subklasse diese Attributmenge erweitern kann. An *isa* können weitere Bedingungen geknüpft sein, z.B. daß die partielle Ordnung als Wald darstellbar ist. Dies schließt Mehrfachvererbung aus und fordert nicht, daß alle Klassendefinitionen Subklassen einer einzigen Superklasse sind.

Zur Spezifikation des Verhaltens von Klassen dienen weitere Eigenschaften:

- Jeder Klassendefinition ist eine Menge von **Nachrichtendefinitionen** zugeordnet. Eine Superklasse **vererbt** ihre **Nachrichtendefinitionen** an die Subklassen, der Subklasse können weitere Nachrichtendefinitionen zugeordnet sein.
- Jede Nachrichtendefinition wird in der zugehörigen Klassendefinition einem **Operator (Methode)** zugeordnet. Eine Nachrichtendefinition, die in mehreren Klassendefinitionen auftritt, kann dort jeweils unterschiedlichen Operatoren zugeordnet sein (**Polymorphie**). Nachrichtendefinitionen, die von einer Superklasse an eine Subklasse vererbt werden, können auf folgende Arten mit Operatoren verknüpft sein: (1) nur in der Superklasse (Vererbung des Operators an die Subklasse), (2) in der Superklasse und in der Subklasse (Überschreiben des Operators in der Subklasse) oder (3) nur in der Subklasse (virtueller Operator aus Sicht der Superklasse).

Zur besseren Verdeutlichung werden die zentralen Begriffe eines ooDB-Schemas mit den korrespondierenden Begriffen eines Schemas im Relationenmodell verglichen (Tab. 1).

Tabelle 1 Zentrale Begriffe von Objektmodell und Relationenmodell

Objektmodell	Relationenmodell
Allgemeine Merkmale	
Klassendefinition	Relationstyp
Klasse	Relation
Objekt	Tupel
Strukturmerkmale	
Attribut	Attribut
-elementar	- 1 NF
-komplex	-NF^2, eNF^2
Vererbung von Attributen	* fehlt *
Verhaltensmerkmale	
Außensicht	
Nachrichtendefinition	* fehlt *
Vererbung von Nachrichtendefinition	* fehlt *
Innensicht	
Operator (Methode)	* fehlt *
Vererbung von Operatoren	* fehlt *

Die Begriffe Klassendefinition und Relationstyp sowie Klasse und Relation entsprechen sich unmittelbar. Während ein Tupel einer Relation ausschließlich über seine Attributwerte identifiziert wird, besitzt jedes Objekt einer Klasse eine von seinen Attributwerten unabhängige Objektidentität. Der dazu verwendete **Objekt-Identifikator (OID)** wird vom ooDBS vergeben, ist systemweit eindeutig, identifiziert ein Objekt über dessen gesamte Lebensdauer und wird nicht wiederverwendet. Die Unterscheidbarkeit von Objekten ist damit nicht an die Vergabe von nutzerdefinierten Primärschlüsseln, die Referenzierung von Objekten nicht auf die Vergabe von Fremdschlüsseln angewiesen. Nutzerdefinierte Schlüssel werden dennoch weiterhin in Objektmodellen benötigt. Sie dienen zur Identifikation von Objekten aus Sicht der nutzenden Umgebung des ooDBS.

Elementare Attribute entsprechen den 1NF-Attributen des (flachen) Relationenmodells. Komplexe Attribute sind nur im (erweiterten) NF^2-Relationenmodell (NF^2 = NF * NF = Non First Normal Form) verfügbar. Vererbung sowie Nachrichtendefinitionen und Operatoren zur Spezifikation des Verhaltens sind im Relationenmodell nicht verfügbar.

3 Datenbankkonzepte

Als zweites zentrales Merkmal gemäß obiger Abgrenzung werden nun die modellunabhängigen Datenbankkonzepte von ooDBS betrachtet.

Für ein Datenbanksystem sind folgende Datenbankkonzepte unentbehrlich (vgl. [4]):

- **Persistenz:** (Daten-) Objekte werden dauerhaft gespeichert.
- **Mehrbenutzerbetrieb:** Der parallele Zugriff mehrerer unabhängiger Nutzer wird unter Sicherung von Konsistenz ermöglicht.
- **Transaktionskonzept**: Synchronisation, Integritäts- und Verlustsicherung von zusammenhängenden Datenbankoperationen werden sichergestellt.
- **Datenbanksprache:** Datenbankoperationen können mit Hilfe einer adäquaten Sprache formuliert werden. Der Zugriff erfolgt sowohl mit Hilfe von Programmiersprachen als auch interaktiv.

In den letzten Jahren wurden unter Berücksichtigung der fortgeschrittenen Technologie weitere Datenbankkonzepte entwickelt, die insbesondere aus der Sicht der Wirtschaftsinformatik für moderne DBS zweckmäßig erscheinen:

- **Verteilung:** (Daten-) Objekte werden verteilt verwaltet und den jeweiligen Nutzern transparent zur Verfügung gestellt.
- **Client-Server-Architekturen:** Die Realisierung von DBS erfolgt in Form von Multiple-Clients/Single-Server- oder Multiple-Clients/Multiple-Server-Architekturen.
- **Versions-Verwaltung:** Das Erstellen und Zusammenführen von (Daten-) Objekt-Versionen sowie die simultane Verwaltung mehrerer Objekt-Versionen werden unterstützt.

4 Objektorientierte Datenbanksysteme

Mittlerweile werden neben einer Vielzahl von Forschungsprototypen bereits eine Reihe von kommerziellen ooDBS auf dem Markt angeboten. Stellvertretend seien genannt: ObjectStore (Object Design), Ontos (Ontos), O_2 (O_2-Technology) und Versant (Versant Object Technologies). Diese ooDBS sind seit zwei bis drei Jahren als Produkte verfügbar. Eine aktuelle Produktübersicht gibt [6].

Für die Implementierung von ooDBS werden häufig objektorientierte Programmiersprachen, in der Regel C++, eingesetzt, seltener Lisp-Varianten oder C. Die Implementierungssprache beeinflußt zur Zeit die Eigenschaften eines ooDBS (z.B. die dynamische Veränderbarkeit eines ooDB-Schemas). Ein standardisiertes Objektmodell fehlt. Einige ooDBS unterstützen als Objektmodell genau die objektorientierten Konzepte der Programmiersprache C++ (z.B. ObjectStore, Ontos).

An der Schnittstelle zur Anwendungsprogrammierung werden in der Regel objektorientierte Programmiersprachen (C++, Smalltalk) unterstützt. In einigen Fällen gibt es Schnittstellen zu Lisp oder C. Über C++ können Funktionen der Programmiersprache C eingebunden werden und damit auch alle konventionellen Programmiersprachen, die über eine C-Schnittstelle verfügen (z.B. Cobol, Fortran, Pascal).

Der fehlende Standard bei Objektmodellen macht sich auch im Fehlen einer standardisierten Anfragesprache bemerkbar. In der Regel werden datenbankspezifische Anfragesprachen angeboten. Zum Teil wird der Versuch unternommen, SQL objektorientiert zu erweitern. Vielfach stehen grafische Nutzeroberflächen für die Schema-Definition mit Editor, Browser und/oder Debugger zur Verfügung.

5 Nutzen und Defizite

Unter dem Gesichtspunkt der Datenbankkonzepte betrachtet, stellen ooDBS zunächst moderne Datenbanksysteme dar, die über alle Merkmale klassischer Datenbanksysteme verfügen und darüber hinaus i.d.R. Client-Server-Architekturen, Verteilung usw. unterstützen. Der spezifische Nutzen von ooDBS entsteht aufgrund der Unterstützung eines Objektmodells:

- ooDBS unterstützen die Bildung und Verwaltung komplexer Objekte. Diese sind nicht nur in sogenannten Nicht-Standardanwendungen (CAx-Techniken, Robotics, CASE usw.), sondern auch in klassischen betrieblichen Anwendungen hilfreich.
- Durch die Kapselung von Struktur und Verhalten der Objekte sowie durch die schwache Kopplung der Objekte mit Hilfe von Nachrichten unterstützen ooDBS eine verbesserte, objektorientierte Anwendungsarchitektur. Diese eröffnet neben einer höheren Daten- und Funktionsunabhängigkeit möglicherweise einen Weg zu flexiblen, evolutiv erweiterbaren Anwendungssystemen.
- Der *impedance mismatch,* d.h. die unzureichende Anbindung eines Datenbanksystems an die Programmiersprache eines Anwendungsprogramms, ist bei ooDBS in Verbindung mit einer objektorientierten Programmiersprache (z.B. C++) günstiger als bei Verwendung eines relationalen DBS in Verbindung mit einer herkömmlichen prozeduralen Programmiersprache.

Den genannten Nutzenpotentialen steht auch eine Reihe von Defiziten gegenüber:

- Es existiert gegenwärtig kein Standard für Objektmodell und Anfragesprache. Dies erschwert die Migration von relationalen DBS zu ooDBS sowie zwischen unterschiedlichen ooDBS.
- ooDBS werden derzeit überwiegend von kleineren Firmen entwickelt und vertrieben. Die dadurch begründete, mit Unsicherheiten behaftete Abhängigkeit dürfte viele potentielle Nutzer davon abhalten, eine strategische Entscheidung zum Einsatz von ooDBS zu treffen.
- ooDBS sind derzeit nur in Verbindung mit objektorientierten Programmiersprachen einsetzbar.

Relationale DBS haben von ihren ersten theoretischen Wurzeln bis zum breiten Einsatz in der Wirtschaft nahezu 20 Jahre benötigt. Verglichen damit dürfte es ooDBS schwerfallen, in den nächsten Jahren breite Einsatzfelder zu erobern. Die Verbreitung von ooDBS wird nicht zuletzt davon abhängen, welche Einsatzfelder die Wirtschaftsinformatik für ooDBS im Bereich von Wirtschaft und Verwaltung erschließt.

Literatur

[1] *Atkinson, M., Bancilhon, F., DeWitt D., Dittrich, K., Maier, D., Zdonik, S.:* The Object-Oriented Database System Manifesto. Proc. First International Conference on Deductive and Object-Oriented Databases (Kyoto 1989), North-Holland, Amsterdam 1991.

[2] *Bancilhon, F., Kim, W.:* Object-Oriented Database Systems: In Transition. In: IEEE Data Engineering, Vol. 13, No. 4 (1990), 24–28.

[3] *Dittrich, K. R.:* Object-Oriented Database Systems: The Next Miles of the Marathon. In: Information Systems, Vol. 15, No. 1 (1990), 161–167.

[4] *Date, C. J.:* An Introduction to Database Systems. Volume I, 5th Edition, Addison-Wesley, Reading, Massachusetts 1990.

[5] *Kim, W., Lochovsky, F. H.:* Object-oriented Concepts, Databases and Applications. Addison-Wesley, Reading, Massachusetts 1989.

[6] *Koch, D.:* Objektorientierte Datenbanksysteme – Marktübersicht 1992. In: Bullinger, H.-J. (Hrsg.): Objektorientierte Informationssysteme II. IAO-Forum, Bd. T29, Springer, Berlin 1992.

[7] *Vossen, G.:* Bibliography on Object-Oriented Database Management. In: SIGMOD RECORD, Vol. 20, Nr. 1 (1991).

[8] *Vossen, G.:* Objekt-orientierte Datenbank-Systeme. Vorlesungsskript Informatik, Universität Koblenz-Landau, Koblenz 1991.