Einsatz objektorientierter Datenbanksysteme in geographischen Informationssystemen

Johann-Christoph Freytag
Institut für Informatik
Humboldt-Universität zu Berlin
Unter den Linden 6
10099 Berlin
freytag@informatik.hu-berlin.de

Zusammenfassung

In den vergangenen fünf Jahren sind objektorientierte Datenbanksysteme (OODBS) soweit entwickelt worden, daß sie in verschiedenen Anwendungsgebieten zum Einsatz kommen können. Die Frage stellt sich nun, inwieweit sie sich auch für Geo-Informationssysteme (GIS) eignen. Das ist Gegenstand dieses Artikels. Zunächst werden die wichtigsten Merkmale objektorientierter Datenbanksysteme genannt, ehe Anforderungen von GIS an die Datenhaltung diskutiert werden. Daraus ergeben sich offene Fragen, anhand derer Weiterentwicklungen im (objektorientierten) Datenbankbereich aufgezeigt werden sollen.

1. Einleitung

In den vergangenen 20 Jahren hat sich die Datenbanktechnologie in Form verschiedener Datenmodelle und Datenbanksysteme in der Informationsverarbeitung durchgesetzt. Auch wenn heute noch vielerorts Dateisysteme bzw. Datenbanksysteme der ersten Generation (IMS der IBM oder Codasyldatenbanken) im Einsatz sind, läßt sich dennoch beobachten, daß in vielen Fällen relationalen Datenbanksystemen der Vorzug gegeben wird.

Frühzeitig hat man jedoch erkannt, daß das relationale Modell bzw. relationale Datenbanksysteme in ihrer jetzigen Form an Grenzen stoßen, die in ihrer begrenzten Fähigkeit der Modellierung und in der nur begrenzt vorhandenen Anzahl an Datentypen zu suchen ist.

Aus diesem Grunde wurde in den vergangenen 15 Jahren der objektorientierte Ansatz aus der Programmiersprachenwelt in die Datenbankwelt übernommen, wo er zur Entwicklung objektorientierter Datenbanksysteme (OODBS) geführt hat. Da diese Systeme mit dem Anspruch entwickelt wurden, weit besser als relationale Systeme den Anforderungen komplexer Anwendungsgebiete gerecht zu werden, stellt sich die Frage, inwieweit Geo-Informationssysteme von den Vorteilen dieser Technologie profitieren können, und welche Funktionalität zu einer besseren Realisierung notwendig ist.

Ziel dieses Artikels ist es, neben einer Diskussion über die wichtigsten Eigenschaften eines OODBS wesentliche Anforderungen aus dem GIS-Bereich der Funktionalität eines OODBS gegenüberzustellen. Dabei werden fehlende Eigenschaften und Schwächen der heutigen Systeme herausgearbeitet und die künftigen Entwicklungstendenzen im OODBS-Bereich dargestellt, die eine weit bessere Nutzung dieser Datenbanktechnologie ermöglichen sollen.

2. Merkmale objektorientierter Datenbanksysteme

Wesentliche Grundlage zur Beurteilung der Frage, ob objektorientierte Datenbanksysteme (OODBSe) für den Einsatz in GIS geeignet sind, ist eine genauere Definition derartiger Systeme. Im wesentlichen handelt es sich bei OODBSen um die Verschmelzung objektorientierter Konzepte aus dem Bereich der Programmiersprachen mit Konzepten aus dem Datenbankbereich. Allerdings sei an dieser Stelle darauf verwiesen, daß im Gegensatz zu relationalen Datenbanksystemen, die auf ein *einheitlich definiertes* Datenmodell zurückgreifen können, bei den OODBSen kein einheitliches objektorientiertes Datenmodell exisitert, auf das sich unterschiedliche OODBSe stützen können.

Im einzelnen sollen von einem OODBS folgende in datenbankorientiert und objektorientiert unterteilte Eigenschaften unterstützt werden, wobei folgende Datenbankeigenschaften in jedem OODBMS wiederzufinden sein sollten:

- **Datenpersistenz:** Daten, die in der Datenbank gespeichert werden, bleiben dort so lange erhalten, bis sie explizit (durch den Benutzer oder auch durch Regeln, die durch das OODBS festgelegt sind) gelöscht werden. Sie sind somit unabhängig von der Ausführungszeit (Lebensdauer) eines Anwendungsprogramms, das auf den Daten operiert.

- **Datenzugriffssynchronisation:** Greifen mehrere Benutzer auf Daten der Datenbank zu, so muß das Datenbanksystem einen geordneten Zugriff sicherstellen, der jedem Benutzer (bzw. jedem Anwendungsprogramm) einen konsistenten Zustand der Daten zur Verfügung stellt, einen korrekten Zustand der Daten bei mehreren Veränderungen garantiert und trotz gleichzeitigem Zugriff auf die Daten sicherstellt, daß Veränderungen nicht verloren gehen. Diese Eigenschaften werden meist als ACID-Prinzp bezeichnet (Atomicity, Consistency, Isolation, Durability) (GR93).

- **Datenzuverlässigkeit:** Tritt ein Fehler in der Hardware oder der Software auf, so liegt es in der Verantwortung des Datenbanksystems, eine Fehlererholung durchzuführen, die die in der Datenbank gespeicherten Daten in einen konsistenten Zustand zurücksetzt.

- **Datensicherheit:** Es liegt in der Verantwortung des Datenbanksystems, gespeicherte Daten vor unberechtigtem Zugriff zu schützen, so daß nur authorisierte Benutzer (oder Benutzergruppen) auf die Daten zugreifen können.

Wie schon zuvor angedeutet, hat ein OODBS auch ein Datenmodell zu unterstützen, das wesentliche Konzepte objektorientierter Programmiersprachen widerspiegelt. Leider kann jedoch nicht auf *das* objektorientierte Modell verwiesen werden, das als „Referenz“ dienen könnte. Aus diesem Grunde wurde in dem Bericht „The Objekt-oriented Database Manifesto“ von Atkinson et al. (Atki89) Kriterien zusammengestellt, die ein OODBS unterstützen sollte. Dabei werden an dieser Stelle nur die wichtigsten Eigenschaften objektorientierter Datenbanksysteme genannt, die von den Autoren des Artikels als „goldene Regeln“ bezeichnet werden.

- **Komplexe Objekte:** Objekte, die in einem OODBS gespeichert werden, müssen aus anderen Objekten durch sogenannte Objektkonstruktoren zusammengesetzt werden können. Das OODBS stellt Basisobjekte zur Verfügung, die den Ausgangspunkt neuer Objektdefinitionen bilden. Mit Hilfe der Objektkonstruktoren soll es dem Benutzer gestattet und möglich sein, „komplexe“ Objekte seiner Modellierungswelt in vollständiger Weise nachzubilden.

- **Objektidentität:** Jedes (gespeicherte) Objekt muß mit seine eigene „Identität“ besitzen. Mit der Erzeugung eines Objektes wird dieses vom OODBS mit einer Identität versehen, die dieses Objekt von allen anderen Objekten der Datenbank unterscheidet. Diese Identität wird unabhängig vom Wert (von den Werten) des Objektes vergeben und verwaltet und bleibt für die Lebensdauer des Objektes unverändert. Nur auf Grund der Objektidentität kann auf Objekte von anderen Objekten (gemeinsam) zugegriffen werden. Der englische Begriff „sharing“ trifft diesen Sachverhalt weit besser. Weiterhin erlaubt die Objektidentität, zyklische Objekte ohne Schwierigkeit zu definieren, da ein Objekt sich durch seine Identität selbst referenzieren kann.

- **Kapselung:** Bei der Kapselung geht es um eine Eigenschaft aus modernen Programmiersprachen, bei der Spezifikation eines Objektzugriffs oder einer Objektmodifikation die Schnittstellenbeschreibung von der Implementation zu trennen. Dem Benutzer soll nur die Spezifikation durch die Schnittstelle bekannt sein, die Implementation jedoch sollte für die Verwendung der „Methode“ unerheblich sein. Diese konsequente Trennung erlaubt es, die Implementation einer Methode (in gewissen Grenzen) zu verändern, ohne daß der Benutzer dieser Methode davon betroffen ist, solange die Schnittstelle und Semantik der Methode unverändert bleiben.

- **Typen und Klassen:** Die Begriffe Typen und Klassen sind im objektorientierten Bereich von ihrer Bedeutung her am meisten umstritten. An dieser Stelle soll unter einer Typdefinition in einem OODBS die Zusammenfassung gemeinsamer Eigenschaften einer Menge von (möglicherweise noch zu erzeugenden) Objekten verstanden werden. Klassen sind die Mengen der momentan in der Datenbank gespeicherten Objekte, die entsprechend gemeinsamen Eigenschaften auch den Zugriff durch Methoden teilen.

- **Vererbung:** Das Vererbungskonzept ist aus den objektorientierten Programmiersprachen wohl bekannt. Es erlaubt, ähnlich strukturierte Objekte mit gemeinsamen Eigenschaften miteinander in eine wohldefinierte Beziehung Subtyp/ Supertyp“ zu setzen, so daß gleiche Methoden auf ähnliche Objekte

angewendet werden können. Durch diese gemeinsame Nutzung wird Redundanz bei der Methodenimplementation vermieden.

- **Überladen und spätes Binden:** Es soll gestattet sein, Methodennamen für unterschiedliche Objekttypen bzw. -klassen wiederzuverwenden, um dem Benutzer größtmögliche Freiheit bei der Namensgebung einzuräumen. Mit anderen Worten, es soll möglich sein, mit gleichen Namen verschiedene Programme zu benennen, die sich aber anhand ihrer Parameterdefinition unterscheiden müssen, um eine Zuordnung zu verschiedenen Typen und Klassen zu erlauben.

- **Vollständigkeit:** Jedes OODBS sollte mit einer Sprache „ausgestattet" sein, die in ihrer Ausdrucksfähigkeit in keiner Weise einer Programmiersprache nachsteht. Diese Forderung ist als Gegensatz zu den Sprachen für relationale Datenbanksysteme zu sehen (i.e. SQL), die in ihrer Mächtigkeit nicht der von Programmiersprachen gleicht.

Mit dieser Zusammenfassung sind die wesentlichsten Merkmale eines OODBS genannt. Diese Aufzählung kann aber nur einen ersten Anhaltspunkt geben, eine ausführlichere Behandlung der angesprochenen Eigenschaften ersetzt sie nicht. Der interessierte Leser wird auf die umfassende Literatur in diesem Bereich hingewiesen.

In den vergangenen zehn Jahren sind auf Grund dieser Ideen verschiedene Prototypsysteme und kommerzielle Systeme entstanden. In der letzteren Kategorie sei auf Firmen wie Objectstore, Objectivity, Ithasca und O2-Technology (Paris, Frankreich) verwiesen, die seit fünf (und mehr) Jahren ihre Produkte im objektorientierten Datenbankbereich anbieten.

3. Nutzung objektorientierter Datenbanksysteme für GIS

Ausgehend von der Beschreibung der Funktionaltät eines OODBS, kann nun der Frage nachgegangen werden, inwieweit diese Systeme zur Unterstützung geographischer Informationssysteme (GIS) geeignet sind.

Hinsichtlich der Modellierung von GIS-Daten stellen OODBSe eine Ausdrucksmächtigkeit zur Verfügung, die es erlaubt, aus einfachen geometrischen Objekten wie Punkten, Linien, Polygonen und Kreisen (den sogenannten Basisobjekten) diejenigen Objekte mit Hilfe der angebotenen Konstruktoren zu erzeugen, die in dieser Anwendung notwendig sind. So ist es sicherlich (wenn auch mit Aufwand verbunden) ohne Schwierigkeiten möglich, Landkarten zu modellieren und zu speichern, wie sie als Beispiel in Abbildung 1 dargestellt sind. Auch vielfältige strukturelle Beziehung zwischen Teilobjekten (beispielsweise die Umrisse eines Bundeslandes und seiner Beschriftung) lassen sich bei entsprechender Sorgfalt ohne weiteres modellieren und speichern.

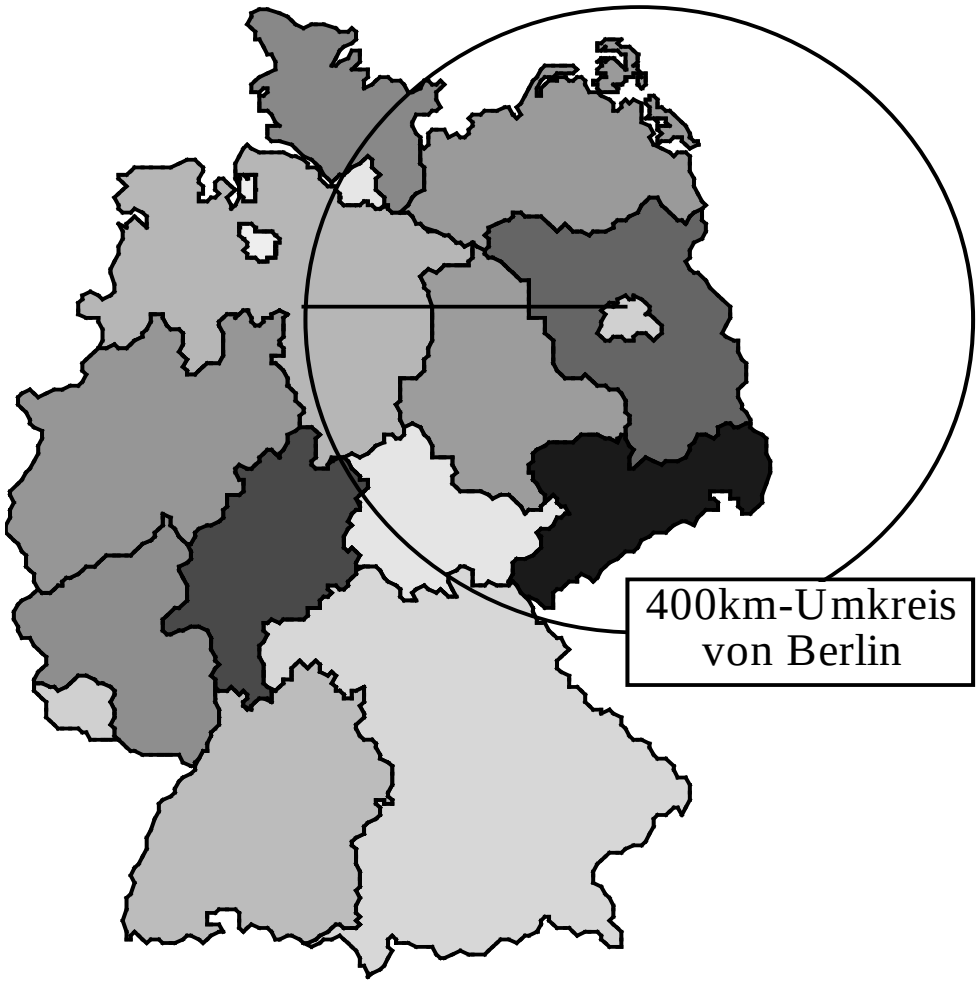

Abbildung 1: Karte von Deutschland

Neben der strukturellen Modellierung läßt die Verwendung von Methoden auch die operationale Seite durch ein OODBS abdecken. Funktionen zur Selektion von Daten oder zu deren Verknüpfung unter Berücksichtigung räumlicher, zeitlicher oder inhaltlicher Beziehungen lassen sich in dem gegebenen Rahmen realisieren. Häufig ist es auch wichtig, bei komplexen Objekten ihre konsistente Speicherung - ebenfalls durch räumliche, zeitliche oder logische Zusammenhänge vorgegeben - nach Veränderungen sicherzustellen. Auch diese kann mit Hilfe programmierter „Datenbankmethoden“ sichergestellt werden.

Alle diese Vorteile sollten aber nicht darüber hinwegtäuschen, daß OODBSe zunächst „anwendungsneutral“ sind. Das bedeutet, daß diese Systeme nur auf Grund der gegebenen Funktionalität die Möglichkeit bieten, die komplexen Strukturen eines GIS relativ problemlos nachzubilden. Dies ist insbesondere als Gegensatz zu relationalen Datenbanksystemen zu sehen, bei denen durch die Tupeldarstellung eine nur sehr beschränkte Möglichkeit der Modellierung gegeben ist. Nur durch den Entwurf und die Implementation notwendiger GIS-Funktionen als Teil einer „GIS-Schicht“, die auch ein OODBS aufsetzt, lassen sich diese Vorteile nutzen. Die Realisierung einer solchen Schicht kann durchaus recht aufwendig sein, da neben der Bereitstellung der notwendigen Funktionalität auch noch die Effizienz des erstellten Systems durchaus wichtig ist. Beispielsweise sind zur effizienten

Verwaltung geometrischer räumliche Zugriffsstrukturen notwendig, die für zwei- oder mehrdimensionale Daten einen schnellen Zugriff ermöglichen. Solche Strukturen sind nicht „per se“ in OODBSen zu finden, sondern gehören ebenfalls zu der schon angesprochenen GIS-Schicht.

Bei der Nutzung OODBSe soll an dieser Stelle auch auf noch existierende Defizite bei diesen Systemen verwiesen werden, die entweder momentan beseitigt werden oder noch zu beseitigen sind. Zu diesen Defiziten gehört das schon erwähnte Problem der unterschiedlichen Datenmodelle, die von den verschiedenen Systemen unterstützt werden. Datenbanksysteme unterschiedlicher Hersteller sind weder von der Funktionalität noch von der Anbindung an eine Programmiersprache her (meist C++) identisch. Diese Uneinheitlichkeit spiegelt sich nicht nur an der Benutzeroberfläche wider, sondern setzt sich auch bei der Implementation und Architektur der OODBSe fort, die dann bei selbst gleicher Oberfläche zu unterschiedlichen Semantiken des Datenmodells führen.

Diese für den OODBS-Markt nachteilige Entwicklung ist von den verschiedenen Herstellern erkannt worden. Durch den Zusammenschluß zur „Object Database Management Group“ (ODMG) haben diese ein Gremium geschaffen, das die Unterschiede verringern bzw. beseitigen soll. Dies führte 1993 zum sogenannten „Object Database Standard: ODMG-93“ (ODM94), in dem ein einheitliches Objektmodell, eine einheitliche Objekt-Definitionssprache, eine Anfragesprache (s. u.) und Bindungen an C++ und Smalltalk als Programmiersprachen festgelegt werden. Dabei wird auch deutlich, daß es weitere Bestrebungen im Bereich objektorientierter Technologie gibt, die auch noch andere Vorstellungen in diesem Bereich entwickeln.

Eine weitere Schwäche der OODBSe ist die Unterstützung des Mehrbenutzerbetriebes und der Fehlererholung durch das Transaktionskonzept. Zwar wird von fast allen Systemen diese Funktionalität unterstützt, doch es bleibt in vielen Fällen offen, inwieweit dieses effizient und dem objektorientierten Datenmodell angepaßt erfolgt. Hier sei auf das Prinzip der Mehrschichtentransaktionen verwiesen (GR93), die sich den meist hierarchischen Strukturen komplexer Objekte ideal anpassen würden. In allen OODBSen fehlen jedoch Mehrschichtentransaktionen.

Weiterhin soll auf die Leistungsfähigkeit existierender OODBSe eingegangen werden, die in vielen Fällen zu wünschen übrigläßt. In diesem Artikel ist es schwer, auf diesen Schwachpunkt im Detail einzugehen, ohne sofort den Widerspruch kommerzieller Anbieter herauszufordern. Aus diesem Grund wird auf den OO7-Benchmark verwiesen (CAR93), der nicht nur Vorschläge unterbreitet, wie die Leistungsfähigkeit eines OODBSs zu messen ist, sondern auch durch konkrete Zahlen nachweist, daß in diesem Bereich noch großer Handlungsbedarf besteht.

4. Anfragesprachen in objektorientierten Datenbanksystemen

Als letzter wichtiger Aspekt soll hier auf die Notwendigkeit und die Schwierigkeit einer Anfragesprache für OODBSe eingegangen werden. In den Jahren des

Entstehens von OODBSen wurde die Meinung vertreten, man könne ohne eine deklarative Anfragesprache auskommen, wie sie in relationalen Datenbanksystemen zu finden ist. Doch gerade die Einschränkung, in OODBSen Datenbankzugriffe nur „navigierend“ gestalten zu können und sich dabei auf existierende Referenzen zwischen Objekten verlassen zu müssen (die möglicherweise in Zukunft Veränderungen unterworfen sind), stellt einen wesentlichen Rückschritt gegenüber relationaler Datenbanktechnologie dar.

Soll beispielsweise eine Anfrage an das OODBS gestellt werden, die (bezugnehmend auf die Karte in Abbildung 1) alle deutschen Städte im Umkreis von 400 km rund um Berlin als Ergebnis liefern soll, so kann diese Anfrage deklarativ sehr einfach wie folgt gestellt werden:

```
select c.name
from c in cities, d in cities
where d.name = "Berlin" and distance(c.position,d.position) < 400
```

Das Fehlen einer deklarativen Anfragesprache und ihrer Notwendigkeit wurde ebenfalls in der schon genannten ODMG-Gruppe erkannt und im „Object Database Standard: ODMG-93“ durch die Sprachdefinition von OQL (Object Query Language) mitberücksichtigt. Mit der Einführung einer solchen Sprache sind komplexe Zugriffe auf Objekte der Datenbank nicht nur einfacher zu formulieren, sondern es wird auch dem OODBS überlassen, in welcher Weise vorhandene Zugriffsstrukturen mit in die Bearbeitung einbezogen werden, falls diese im System vorhanden sind und genutzt werden sollen. Ergeben sich Veränderungen in der physischen Struktur der Datenbank, so bleibt - wie in relationalen Datenbanksystemen - die Anfrage davon unberührt, auch wenn der notwendigerweise vorhandene Anfrageprozessor (der diese Anfrage in eine prozedurale Form überführt) eine Neuübersetzung vornehmen muß, um einen neuen „Zugriffsplan” zu generieren. Dieser Vorteil der Unabhängigkeit von der physischen Datenorganisation bzw. von Referenzen zwischen Objekten trägt mit zur geringeren Notwendigkeit der Änderung von existierenden Programmen bei. Dabei steht diese Einbettung von Anfragesprachen in OODBSen noch am Anfang, auch wenn das System O2 der Firma O2-Technology als einziges System von Anfang an eine Anfragesprache angeboten hat. Als wesentliche Schwierigkeit stellt sich dabei das Problem der Anfragebearbeitung, insbesondere der Optimierung von Anfragen heraus. Auf Grund mächtigerer Modellierungsmöglichkeiten gestaltet sich dieses Problem weit komplexer als in relationalen Datenbanksystemen, auch wenn es schon genügend Ansätze für eine kommerzielle Umsetzung dieser Idee gibt.

5. Weiterentwicklungen im Bereich OODBMS

Mit der Entwicklung OODBSe wurde eine Alternative zu relationalen Datenbanksystemen geschaffen. Es stellt sich bei dieser Entwicklung die Frage, inwieweit OODBSe und relationale Datenbanksysteme miteinander konkurrieren und ob für beide Systeme ein Zukunft besteht.

Um diese Frage auf eine technische Grundlage zu stellen, wird auf eine von *M. Stonebraker* während der VLDB-Konferenz (Very Lare Database) 1995 in Zürich während eines Vortrags geäußerten Meinung zurückgegriffen, die auf eine Klassifizierung der unterschiedlichen Datenspeicherungsmöglichkeiten beruht, wie sie in Abbildung 2 gezeigt wird.

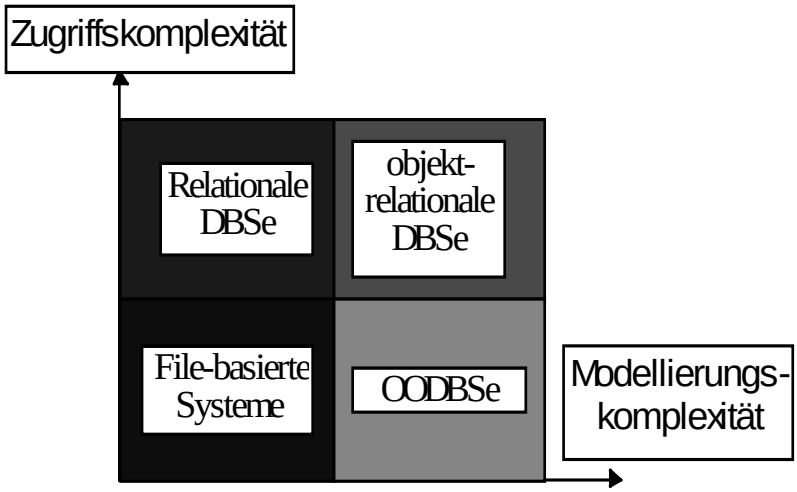

Abbildung 2: Kategorien von Datenbanksystemen (nach M. Stonebraker)

Danach eignen sich relationale Datenbanksysteme bevorzugt für einfach strukturierte Daten mit komplexen Anforderungen hinsichtlich des Datenzugriffs, während sich OODBSe für die komplexe Datenmodellierung mit einfachen Anforderungen an die Manipulation eignen. Systeme, die sowohl komplexen Datenmodellierungsanforderungen als auch komplexen Manipulationsmöglichkeiten gerecht werden, sind sogenannte objekt-relationale Systeme, die auch als hybride Systeme aus relationalen und objektorientierten Datenbanksystemen verstanden werden können. In ihrer Realisierung folgen sie meist der Architektur relationaler Systeme, während das Datenmodell die wichtigsten Konzepten objektorientierter Systeme einbezieht. Diese Systeme sind stark von der Weiterentwicklung des relationalen Gedankens geprägt, wie sie sich in der Erweiterung der SQL-Sprache zum SQL-3- Standard widerspiegelt. Als wichtigste kommerzielle Vertreter dieser Systeme, die erst seit kurzem auf dem Markt zu finden sind, sollen das Datenbanksystem Illustra (ILL95) und das System UniSQL (KIM94) genannt werden. Mit einigen Einschränkungen fällt in diese Kategorie auch das IBM-Datenbanksystem DB2/AIX Version 2, das mit der Einführung von benutzerdefinierten Funktionen (UDFs) und benutzerdefinierten Typen (UDTs) einen ersten Schritt in diese neue Richtung einschlägt (IBM95).

6. Ausblick

In diesem Artikel wurde mit der Nennung von Eigenschaften objektorientierter Datenbanksysteme eine Grundlage geschaffen, auf der dann der Einsatz dieser Systeme bei der Realisierung von Geo-Informationssystemen diskutiert wurde. Während OODBSe den strukturellen Anforderungen eines GIS gerecht wird, ist auch

auf den erheblichen Aufwand verwiesen worden, der bei der Realisierung eines GIS mit Hilfe von OODBEen noch notwendig ist. Dabei wurde auch auf vorhandene Schwächen von OODBSen und insbesondere auf die Notwendigkeit einer deklarativen Anfragesprache eingegangen. Zum Abschluß wurden die Weiterentwicklung und die alternativen Ansätze genannt, die momentan auf dem kommerziellen Datenbankmarkt entstehen. Zum jetzigen Zeitpunkt ist nicht abzusehen, in welcher Weise sich die unterschiedlichen Systeme in der Zukunft weiterentwickeln und auf dem kommerziellen Markt behaupten werden.

7. Literaturhinweise

ATK89 Atkinson, M., Bancilhon, F., DeWitt, D., Dittrich, K., Maier, D., Zdonik, S.: The Object-Oriented Database Manifesto, Proceedings DOOD 1989, Kyoto, Seiten 40-57

GR93 Gray, J., Reuter, A., Transaction Processing: Concepts and Technics, Morgan Kaufmann Publishers, Inc., San Francisco, 1993

ILL95 Illustra Information Technologies, Inc., Illustea User's Guide, Release 2,4,1, Oakland, CA, März 1995

IBM95 IBM DATABASE 2 Version 2, Administration Guide, Version 2.3, IBM Copr. 1995

KIM94 Kim, Won: UniSQL/X Unified Relational and object-Oriented Database System, Proceedings SIGMOD Conference, Minneapolis, USA, 1994 Seite 481

CAR93 Carey, M., DeWitt, D., Naughton, J.F.: The 007 Benchmark, Proceedings SIGMOD Conference, Washington, D.C., 1993, Seiten 12-21

ODM94 Cattell, R.G.G. (editor): The object Database Standard: ODMG-93, Release 1.1, Morgan Kaufmann Publishers, Inc., San Francisco, 1994