Hochschule Neubrandenburg
University of Applied Sciences

Studiengang Geoinformatik

Speicherung raum-zeitlicher Daten und deren Visualisierung

Bachelorarbeit

vorgelegt von: Markus Bradke

Zum Erlangen des akademischen Grades

„Bachelor of Engineering“ (B.Eng.)

Erstprüfer: Prof. Dr.-Ing. Tobias Hillmann
Zweitprüfer: Prof. Dr.-Ing. Ernst Heil

Bearbeitungszeitraum: 31. Juli 2009 – 25. September 2009

urn:nbn:de:gbv:519-thesis2009-0295-5

Eidesstattliche Erklärung

Hiermit versichere ich, die vorliegende Bachelorarbeit ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus den Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Neubrandenburg, den 24.09.2009

Markus Bradke

Kurzfassung

Die Zeit ist eine allseits gegenwärtige Komponente der realen Welt. Letzten Endes unterliegen alle denkbaren Informationen, vor allem aber räumliche Informationen, zeitlichen Veränderungen. Nicht-temporale Datenmodelle und deren Implementierung in Geoinformationssystemen (GIS) und Datenbankmanagementsystemen (DBMS) können aber lediglich einen bestimmten Zustand der realen Welt speichern, sodass die Dynamik, denen Geoobjekte unterliegen, im Datenmodell nicht berücksichtigt werden kann. Ein temporales GIS muss aber Funktionen und Methoden zur spatio-temporalen Speicherung, Analyse und Visualisierung zur Verfügung stellen. Die vorliegende Arbeit beschränkt sich auf die Entwicklung eines temporalen 3D-Datenmodells zur Speicherung raum-zeitlicher Veränderungen auf Basis zweier relationaler Datenbanktechnologien: Oracle und TimeDB. Das 4D-Datenmodell wurde auf Grundlage der vorangegangenen Untersuchungen im Bereich der Datenbanksysteme sowie der topologischen und temporalen Modelle implementiert. Darüber hinaus werden Möglichkeiten zur Visualisierung raum-zeitlicher Dynamik aufgezeigt.

Abstract

Time is a current component of real world phenomena. At least all conceivable information, especially spatial features, change their states over time. Non-temporal data models and their implementations in geoinformation systems (GIS) and database management systems (DBMS) capture a single state of the real world and cannot cover the dynamics of spatial features. A temporal GIS needs to provide functionality for spatio-temporal data management, analysis as well as visualization. The present bachelor thesis restricts on the implementation of a temporal 3D model for the storage of spatio-temporal changes based on existing relational database technologies: Oracle and TimeDB. The implementation of the 4D data model is based on the analysis of the database systems and the topological and temporal models described in the previous chapters. Further more some options for the visualization of geo-scientific processes are mentioned.

Inhaltsverzeichnis

1 Einführung

1.1 Problemstellung

Objekte der realen Welt unterliegen neben thematischen vor allem räumlichen Veränderungen im Zeitverlauf. Die als raum-zeitliche Variabilität bezeichnete Eigenschaft eines Geoobjekts ist von einem Geoinformationssystem (GIS) allerdings nur schwer zu erfassen und darzustellen, da die meisten konventionellen GIS zumeist statisch ausgelegt sind und die Dynamik, denen Geoobjekte unterliegen, nur unzureichend in ihrem Datenmodell und ihrer Funktionalität berücksichtigen [ZK01]. Derartige GIS werden auch als Schnappschuss-GIS bezeichnet, da sie lediglich einen bestimmten Zustand der realen Welt beschreiben können. Diese Restriktion wird in Anwendungen kritisch, in denen die Entwicklung von Objekten im Zeitverlauf berücksichtigt werden muss. Es besteht daher ein Bedarf an neuen Datenmodellen, die den Ansprüchen zeitabhängiger Daten genügen.

In den letzten Jahren wurden zwei Ansätze zur Berücksichtigung zeitvarianter Geoobjekte in verschiedenen Studien verfolgt. Auf der einen Seite Versuche vorhandene Geoinformationssysteme um temporale Aspekte (TGIS) zu erweitern und auf der anderen Seite Untersuchungen im Bereich der spatio-temporalen Datenbankentwicklung. Da klassische GIS zumeist dateibasiert arbeiten und somit Schwächen in der persistenten und effizienten Verwaltung großer Datenmengen aufweisen, erweist es sich als sinnvoll, entsprechende datenbankbasierte Modelle zu entwickeln. Das Ziel der vorliegenden Arbeit war die Realisierung eines temporalen 3D-Datenmodells auf Basis eines geeigneten Datenbankmanagementsystems.

1.2 Gliederung der Arbeit

Im nachfolgenden Kapitel wird zunächst der Begriff 4D-GIS erläutert. Dabei werden, ausgehend von der Definition eines konventionellen GIS, Anforderungen an ein temporales 3D-GIS (4D-GIS) definiert. Darüber hinaus werden mögliche Anwendungsgebiete eines 4D-GIS dargelegt.

Im dritten Kapitel werden verschiedene Datenbanksysteme untersucht, die für die Entwicklung eines 4D-GIS in Frage kommen. Dabei werden Vor- und Nachteile der Datenbanksysteme vor allem in Bezug auf die räumliche Datenhaltung untersucht. Des Weiteren wird der temporale Aspekt in Datenbanksystemen beschrieben und anhand dessen ein begründeter Entscheidungsvorschlag für die Implementierung eines 4D-Datenmodells in einem der vorgestellten Datenbanksysteme gegeben.

Kapitel 4 legt die Grundlagen raum-zeitlicher Modellierung dar, d.h. dass verschiedene topologische und temporale Datenmodelle vorgestellt und die Vorzüge der jeweiligen Modelle herausgearbeitet werden.

Das objektrelationale Geodatenbanksystem Oracle und dessen grundlegenden räumlichen Konzepte sowie die temporale Datenbank TimeDB sind in Kapitel 5 beschrieben. Im sechsten Kapitel erfolgt die Realisierung eines relationalen temporalen 4D-Datenmodells auf Basis der in Kapitel 4 vorgestellten Modelle. Darüber hinaus wird das entwickelte Modell kritisch hinterfragt und weiterführende Lösungsvorschläge unterbreitet.

In Kapitel 7 werden verschiedene Möglichkeiten zur Visualisierung raum-zeitlicher Dynamik aufgezeigt. Dabei werden sowohl Geoinformationssysteme als auch Game-Engines in die Betrachtung mit einbezogen.

Abschließend folgen eine kurze Diskussion des Erreichten sowie ein Ausblick für mögliche weiterführende Arbeiten.

2 4D-GIS

2.1 Begriff Geoinformationssystem

Nach BILL und FRITSCH [BF94] ist ein Geoinformationssystem „ein rechnergestütztes System, das aus Hardware, Software, Daten und den Anwendungen besteht. Mit ihm können raumbezogene Daten digital erfasst und redigiert, gespeichert und reorganisiert, modelliert und analysiert sowie alphanumerisch und graphisch präsentiert werden“. Ein Geoinformationssystem grenzt sich demnach durch die Spezialisierung auf Geodaten ab, die gegenüber alphanumerischen Daten in herkömmlichen Informationssystemen besonderen Anforderungen unterliegen. Im Allgemeinen werden bei einem Geoinformationssystem, unabhängig von der zu betrachtenden Dimension, die vier Komponenten Eingabe, Verwaltung, Analyse sowie Präsentation (EVAP) unterschieden [BF94]. Abbildung 2.1 beschreibt das Zusammenspiel dieser Komponenten. Bei näherer Betrachtung dieser Darstellung wird deutlich, dass der Kern eines jeden GIS die Geodatenbank darstellt, in der Geodaten verwaltet und gespeichert werden, um interaktive Manipulationen und Verarbeitungsschritte zuzulassen.

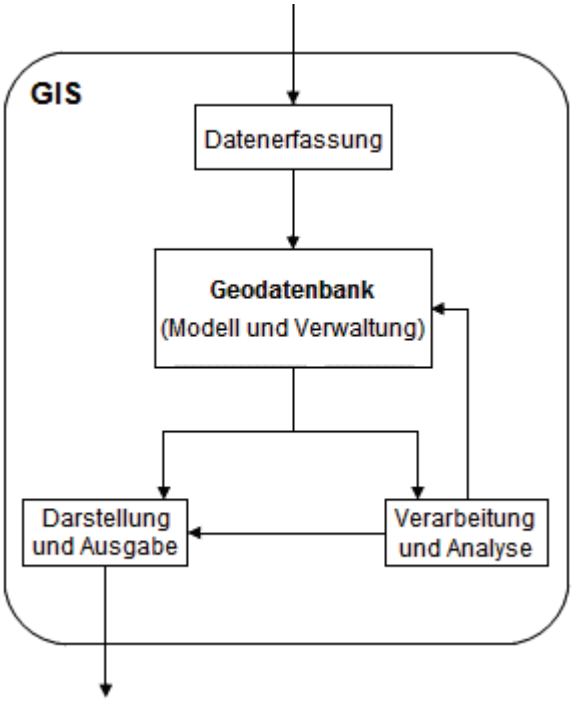

Abb. 2.1: Vier-Komponentenmodell eines Geoinformationssystems (vgl. [Brink08])

Die meisten konventionellen GIS sind darauf ausgelegt, Geoinformationen im zweidimensionalen Raum abzubilden (2D-GIS). Viele Experten waren bis vor wenigen Jahren der Ansicht, dass die Abbildung von Geodaten in der Ebene vollkommen ausreiche. Dies lag nach GIGER [in CZ05] v. a. daran, dass den Kosten für die Erfassung dreidimensionaler Daten kein entsprechender Nutzen gegenüber stand. Durch neue Technologien im Bereich der Erfassung und Auswertung von photogrammetrischen und Laserscanning-Daten hat sich dieser Zustand allerdings geändert. Zunächst wurde zur Lagegeometrie die Höhe lediglich als zusätzliches Attribut gespeichert (2.5D-GIS). Dies führte in der Darstellung zu Flächen, die ein Höhenrelief, aber kein Volumen aufweisen. Dreidimensionale GIS (3D-GIS) sind hingegen in der Lage, die Höhe als gleichwertige Dimension zu betrachten, sodass Volumina von Körpern entsprechend dargestellt und analysiert werden können. Viele kommerzielle GIS sind bereits in der Lage, die dritte Dimension abzubilden. Zu nennen sind hierbei v. a. ArcGIS und AutoCAD Map 3D (vgl. Abschnitt 7.1: Geoinformationssysteme).

An Geoinformationssysteme, in denen zusätzlich zu den drei räumlichen Komponenten *x, y, z* ein zeitlicher Parameter *t* mitgeführt wird, der ein Geoobjekt kontrolliert, werden andere Anforderungen gestellt.

2.2 Anforderungen an ein 4D-GIS

Abbildung 2.1 beschreibt die Komponenten eines konventionellen GIS. Diese Komponenten können als Grundlage genutzt werden, um die Anforderungen an ein vierdimensionales GIS zu definieren. Ein vierdimensionales Geoinformationssystem grenzt sich von anderen Softwaresystemen im Allgemeinen durch die folgenden Aspekte ab:

4D Datenmodellierung
Ein 4D-GIS muss ein Datenmodell zur Verfügung stellen, in dem sowohl die räumlichen als auch temporalen Eigenschaften eines dreidimensionalen Geoobjekts in geeigneter Weise beschrieben werden können.

Spatio-Temporales Geodatenmanagement
Ein 4D-GIS muss eine leistungsfähige Geodatenbank bereitstellen, in der große Datenmengen effizient und dauerhaft gehalten werden können. Des Weiteren müssen spatio-temporale Zugriffsmethoden auf die Geodaten sichergestellt werden.

Spatio-Temporale Analysen
Ein 4D-GIS muss raum-zeitliche Analysen für temporale 3D-Geoobjekte zulassen, um Erklärungen, Auswertungen und Vorhersagen zeitvarianter Daten liefern zu können.

Spatio-Temporale Visualisierung
Ein 4D-GIS muss die zeitliche Veränderung von dreidimensionalen Geoobjekten in anschaulicher Weise darstellen können. Die Darstellung kann sowohl statisch als auch dynamisch, z.B. als Animation für geo-wissenschaftliche Prozesse, erfolgen.

2.3 Anwendungsgebiete

Die meisten bisher existierenden Geoinformationssysteme bilden Geodaten nur im zweidimensionalen Raum zu einem bestimmten Zeitpunkt ab. Sie werden deshalb auch als Schnappschuss-GIS bezeichnet. Das Interesse an der Verwaltung, Analyse und Visualisierung dynamischer 3D-Geoobjekte stieg in den letzten Jahren erheblich.

Anwendungsgebiete lassen sich in allen Bereichen der Geowissenschaften finden. In der Geologie können Plattentektonische Bewegungen und Gebirgsbildungen anhand dynamischer Modelle rekonstruiert werden. Im Bereich der Klimaforschung werden Modellrechnungen zur Klimaanalyse und –vorhersage aufgestellt. In der Geodäsie sind 4D-Modelle für die Analyse und Visualisierung von Deformationen an Bauwerken gefragt. Eine Gesamtlösung zur Darstellung raum-zeitlicher Daten gibt es derzeit allerdings nicht. Zumeist werden individuell zugeschnittene und auf die jeweiligen Anforderungen ausgelegte Programme entwickelt. Dabei wird auf verschiedene einzelne Softwarekomponenten zurückgegriffen, die miteinander interagieren (z.B. externe Datenbank und Visualisierungssoftware).

3 Datenbanksysteme

Der Datenverwaltung und –speicherung kommt in einem Geoinformationssystem eine zentrale Rolle zu. Ein Datenbanksystem ist ein System zur elektronischen Datenverwaltung, das aus zwei Komponenten besteht: dem Datenbankmanagementsystem (DBMS) und der eigentlichen Datenbank. Die Datenbank ist eine strukturierte Sammlung einheitlich beschriebener, persistent gespeicherter Daten [Brink08]. Das DBMS dient als Schnittstelle zwischen der Datenbank und den Anwendern und ermöglicht einen einheitlichen und effizienten Zugriff auf die Daten, ohne das deren interne Datenstruktur bekannt ist. Abbildung 3.1 beschreibt den Zugriff eines Anwenderprogramms über die Programmschnittstellen des DBMS auf die Datenbank.

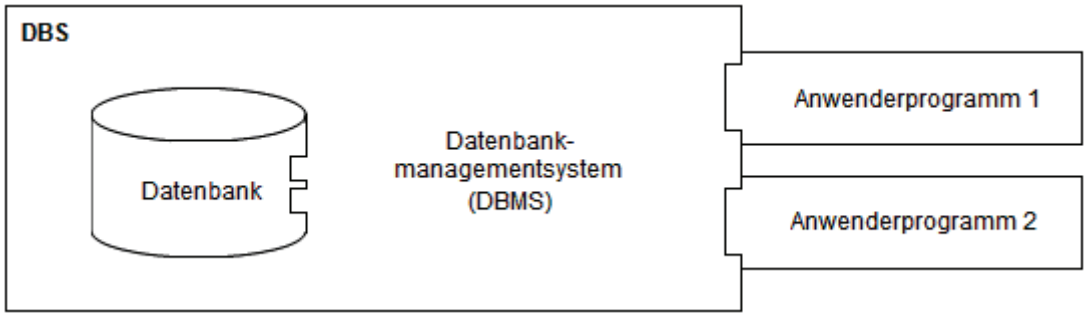

Abb. 3.1: Aufbau eines Datenbanksystems (vgl. [Bart05])

Die Motivation zum Einsatz eines Datenbanksystems ergibt sich aus den Unzulänglichkeiten beim Nutzen von Dateisystemen. Aus diesen Schwächen resultieren die nachfolgend beschriebenen Eigenschaften eines Datenbanksystems.

Datenunabhängigkeit

Datenunabhängigkeit fordert, dass die enge Verknüpfung zwischen den Daten und den Anwenderprogrammen aufgelöst wird, sodass Änderungen bei der Datenspeicherung möglichst ohne Einfluss auf das Anwenderprogramm bleiben [Lange02, Brink08]. Es wird zwischen logischer und physischer Datenunabhängigkeit unterschieden. Logische Datenunabhängigkeit bedeutet, dass Änderungen am Datenmodell ohne Bedeutung für den Anwender bleiben, wohingegen bei der physischen Datenunabhängigkeit Änderungen an der physischen Repräsentation der Daten ohne Einfluss auf den Anwender bleiben.

Zentrale Datenhaltung, Mehrbenutzerbetrieb und Datensicherheit

Die Daten werden zentral auf einem Server gehalten, sodass diese den Anwendern in gleicher Form zur Verfügung stehen. Dies hat zur Folge, dass das Datenbanksystem einen Mehrbenutzerbetrieb sicherstellen muss, in dem mehrere Nutzer gleichzeitig sowohl lesend als auch schreibend auf die Datenbank zugreifen können [Brink08]. Um Fehler bei Transaktionen zu vermeiden, muss das DBMS ein Transaktionskonzept unterstützen, das die konsistente Datenhaltung in Datenbanksystemen gewährleistet.

Dies bedeutet, dass das DBMS sicherstellen muss, dass alle Manipulationsschritte einer Transaktion entweder ganz oder gar nicht durchgeführt werden. Ausfälle in der Datenbank (z.B. Plattencrash) oder Fehler in den Anwenderprogrammen dürfen nicht dazu führen, dass die Manipulationen an der Datenbank physische Strukturen zerstören oder fehlerhafte Manipulationen in der Datenbank hinterlassen [BZ01].

Datenintegrität

Bei der Speicherung von Daten ist darauf zu achten, sämtliche Informationen nur einmal abzuspeichern (Redundanzfreiheit), um die Gefahr von Dateninkonsistenzen zu vermeiden und somit Widersprüche im Datenbestand auszuschließen.

Anfragebearbeitung

Im Gegensatz zu einem Dateisystem stellt ein Datenbanksystem eine strukturierte Datenbanksprache (SQL, *Structured Query Language*) zur Verfügung, die es erlaubt, den Datenbestand hinsichtlich vielfältiger Kriterien abzufragen (IQL, *Interactive Query Language*), zu beschreiben (DDL, *Data Definition Language*) oder zu manipulieren (DML, *Data Manipulation Language*).

Zugriffskontrolle und Datenschutz

In Datenbanksystemen ist eine Benutzerverwaltung integriert, über die der Zugriff auf die Datenbank gesteuert werden kann. Diese erlaubt es, verschiedenen Nutzern unterschiedliche Lese- und Schreibrechte bezüglich der Datenbank zuzuweisen. Des Weiteren ist es von großer Bedeutung die in einer Datenbank gespeicherten Informationen gegenüber unbefugtem Zugriff zu schützen.

3.1 Relationale Datenbanksysteme

Das auf den theoretischen Grundsätzen von EDGAR F. CODD beruhende relationale Datenmodell von 1970 hat sich seit Mitte der 1980er Jahre als Standard kommerzieller Datenbankmanagementsysteme etabliert [Lange02]. Das relationale Datenmodell basiert auf der relationalen Algebra, die auch als theoretische Grundlage der von CODD entwickelten Datenbanksprache SQL dient.

3.1.1 Relationen

Relationale Datenbanken organisieren Daten in Tabellen (*Relation*) und verwalten die zwischen diesen Tabellen bestehenden Beziehungen. Eine Zeile einer Tabelle wird auch als *Tupel* (entspricht einem Datensatz) bezeichnet. *Attribute* sind einzelne Spalten einer Tabelle, in denen die verschiedenen Eigenschaften der Datensätze gespeichert sind. Das nachfolgende Beispiel zeigt die Relation „Gebäude“, die durch die Attribute ID, ADRESSE und GRUNDFLAECHE beschrieben wird.

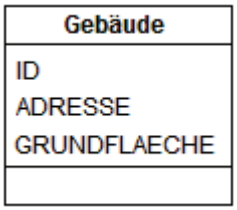

Abb. 3.2: Relation "Gebäude" in UML-Notation

Ein Datensatz muss eindeutig identifizierbar sein. Für die eindeutige Kennzeichnung eines Datensatzes in einer Tabelle werden deshalb *Schlüssel* (engl. *Key*) eingesetzt, die aus einem Attribut oder einer Kombination mehrerer Attribute bestehen können. Der *Primärschlüssel* (engl. *Primary Key*) dient zur eineindeutigen Identifikation der Datensätze einer Tabelle. In der Tabelle „Gebäude“ enthält das Attribut ID den Primärschlüssel, über den alle anderen Attribute identifiziert werden können.

3.1.2 Beziehungen

Die Realisierung von *Beziehungen* zwischen Datensätzen erfolgt im relationalen Datenmodell über *Fremdschlüssel* (engl. *Foreign Key*) [Brink08]. Fremdschlüssel sind Verweise eines Datensatzes einer Tabelle auf einen Datensatz einer anderen Tabelle. Grundsätzlich werden drei Basis-Beziehungstypen unterschieden:

Eins-zu-eins-Beziehung (1:1)
Jedem Datensatz in Tabelle A steht genau ein Datensatz in Tabelle B gegenüber und umgekehrt. In den meisten Fällen können beide Entitäten zu einer zusammengefasst werden.

Eins-zu-mehrfach-Beziehung (1:n)
Jedem Datensatz in Tabelle A stehen *n* Einträge in Tabelle B gegenüber. Diese Form sollte beim Datenbankdesign angestrebt werden.

Mehrfach-zu-mehrfach-Beziehung (n:m)
Bei einer n:m-Beziehung kann es zu einem Datensatz einer Tabelle A mehrere zugehörige Datensätze einer Tabelle B geben und umgekehrt. Diese Konstellation erfordert in relationalen Datenbanken das Auflösen der n:m-Beziehung in zwei 1:n-Beziehungen und das Einführen einer Verknüpfungstabelle, über die die Datensätze eindeutig zugeordnet werden können.

Das nachfolgende Beispiel soll eine n:m-Beziehung zwischen der Relation „Gebäude“ und der Relation „Eigentümer“ verdeutlichen.

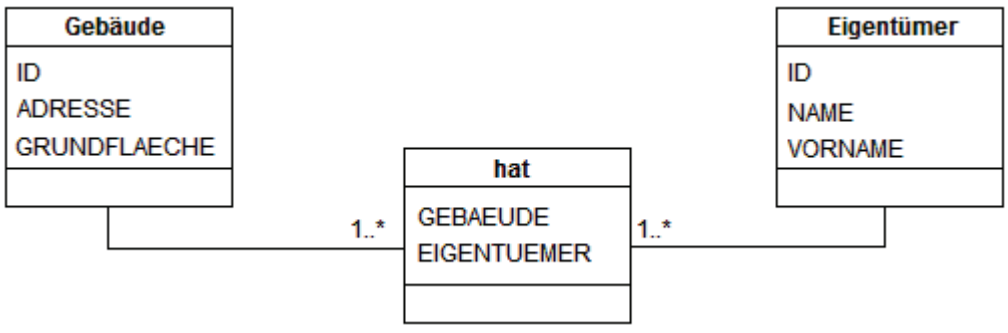

Abb. 3.3: Beziehungsdiagramm zwischen "Gebäude" und "Eigentümer" in UML-Notation

Die Beziehung „hat“ bildet eine n:m-Beziehung zwischen den Datensätzen der Relationen „Gebäude“ und „Eigentümer“. Die Tabelle „hat“ besitzt dabei die Fremdschlüssel GEBAEUDE und EIGENTUEMER und baut somit die Beziehung zwischen den beiden Tabellen „Gebäude“ und „Eigentümer“ auf.

3.1.3 Normalisierung

Zu einer der wichtigsten Eigenschaften eines Datenbanksystems zählt die Dateninteg rität. Bei der Datenspeicherung ist darauf zu achten, Datenredundanzen und somit Dateninkonsistenzen und Widersprüche im Datenbestand zu vermeiden. Durch mehrfach abgespeicherte Werte oder infolge einer suboptimalen Strukturierung entstehen ein hoher Speicherbedarf sowie ein langsamerer Zugriff auf die Daten. Aus diesem Hintergrund wurde das Konzept der *Normalformen* entwickelt, mit dem eine relationale Datenbank optimiert werden kann [Lange02].

Eine Relation befindet sich in *erster Normalform*, wenn es für jeden Entitätstypen eine eineindeutige ID (Primärschlüssel) gibt, die über alle Instanzen dieser Entität eindeutig und einen Wert ungleich *Null* für jede Instanz besitzt. Des Weiteren muss die Atomarität für jedes Attribut gesichert sein. Die Relation „Gebäude“ aus Abbildung 3.2 befindet sich nicht in erster Normalform, da das Attribut ADRESSE aus einer Werteliste (Straße, Hausnummer, Postleitzahl, Ort) besteht und somit nicht atomar ist. Um die Atomarität des Attributs herzustellen, muss dieses in mehrere Attribute aufgespaltet werden. Das Ergebnis der atomisierten Attribute ist in Abbildung 3.4 dargestellt.

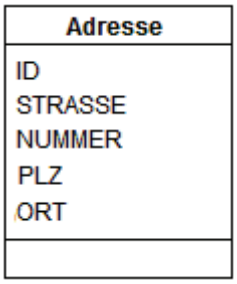

Abb. 3.4: Relation "Adresse" in erster Normalform

Eine Relation befindet sich in *zweiter Normalform*, sofern die erste Normalform vorliegt und alle nichtidentifizierenden Attribute voll funktional von der eindeutigen ID der Entität abhängen. Es dürfen keine Abhängigkeiten zwischen Nicht-Schlüsselattributen und Teilen des Schlüssels bestehen. Diese Abhängigkeiten können infolge dessen also nur bei zusammengesetzten Schlüsseln auftreten.

Eine Relation befindet sich in *dritter Normalform*, sofern die zweite Normalform vorliegt und keine transitiven Abhängigkeiten zwischen einem Schlüssel und weiteren Nicht-Primatattributen vorliegen. In den meisten Fällen reicht die Untersuchung der ersten drei Normalformen aus. Ferner können die nachfolgenden Normalformen als Optimierungskriterien für relationale Datenbanken herangezogen werden, die in [Sauer98] ausführlich beschrieben sind.

Boyce-Codd-Normalform (BCNF) → Vierte Normalform → Fünfte Normalform

3.1.4 Eignung als 4D-GIS

Das relationale Modell eignet sich durch seine Einfachheit und Universalität sehr gut zur Speicherung thematischer Daten. Aber gerade bei der Modellierung von Geodaten lassen sich aus diesen Vorteilen die Hauptkritikpunkte ableiten. Das Hauptproblem bei relationalen Datenbanken ist, dass nur atomare Attribute zulässig sind. Zur Repräsentation eines Gebäudes kann ein Multipolygon als geometrisches Primitiv herangezogen werden. Ein solches Multipolygon, das aus einer Folge von Polygonen besteht, kann nicht atomar in einer relationalen Datenbank gespeichert werden, sodass dieses auf mehrere Tabellen verteilt und über Fremdschlüssel miteinander verbunden werden muss (siehe Abbildung 3.5). [Brink08]

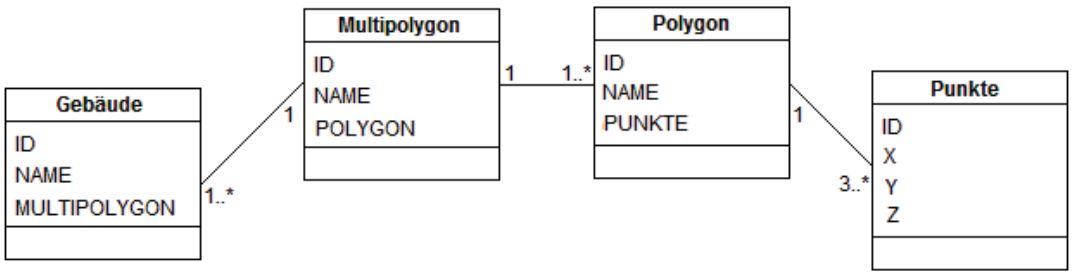

Abb. 3.5: Beziehungsdiagramm für Multipolygone in UML-Notation

Aus dem dargestellten Modell lassen sich die nachfolgend beschriebenen Nachteile relationaler Datenbanken bei der Speicherung von Geodaten ableiten [Bart05, Breu01, Brink08]:

Geringe Separabilität von Geodaten
Geodaten sind eng miteinander verflochten. Im dargestellten Modell werden zusammengehörige Informationen aber verteilt gehalten, sodass ein Zusammenführen der Daten sehr aufwändig wäre. Des Weiteren sind Geodaten nicht so separabel wie andere Daten, d.h. nicht beliebig weit in atomare Bestandteile zerlegbar. Darüber hinaus fehlen Konstruktoren zur Beschreibung komplexer Geoobjekte.

Fehlen benutzerdefinierter Datentypen
Ein weiterer Nachteil des relationalen Modells ist das Fehlen benutzerdefinierter Datentypen, z.B. geometrische Datentypen zur Beschreibung komplexer Geometrien. Darüber hinaus lassen sich entsprechende geometrische Operationen und multidimensionale Zugriffsmethoden schwer definieren.

Komplexe Anfragen
Durch die verteilte Haltung zusammengehöriger Informationen lassen sich entsprechende SQL-Abfragen nur schwer handhaben. Die Anfragen sind sehr komplex und benutzerunfreundlich. Beziehungen zwischen Geoobjekten lassen sich nur über aufwendige Verbindungs-Operationen herstellen.

Auf der einen Seite sprechen die beschrieben Nachteile gegen den Einsatz eines rein relationalen DBMS als Basis für ein 4D-GIS. Auf der anderen Seite ist es dennoch möglich, dreidimensionale Geodaten in einer relationalen Datenbank effizient zu verwalten. Durch Einführung geeigneter Restriktionen, z.B. durch Verwendung lediglich einer zulässigen Geometrie, lassen sich räumliche Daten wesentlich einfacher zusammenführen. Entsprechende SQL-Anfragen sind weniger komplex und dadurch auch erheblich einfacher zu formulieren. Darüber hinaus basieren viele temporale Datenbanken auf relationalen Modellen. Diese sind sehr gut geeignet zur temporalen Repräsentation von Attributen. Des Weiteren bieten die temporalen Anfragesprachen ein hohes Maß an Integration mit der nicht-temporalen Anfragesprache SQL. Für eine erste prototypische Umsetzung eines 4D-Datenmodells er-

weist sich der Einsatz relationaler Datenbanktechnologien daher als durchaus sinnvoll.

3.2 Objektorientierte/ Objektrelationale Datenbanksysteme

Mit der Verbreitung objektorientierter Programmiersprachen (z.B. Java, Python, C#) in den 1990er Jahren wurden auch zunehmend *objektorientierte Datenbanksysteme* entwickelt und angeboten. Diese unterscheiden sich von relationalen Datenbanksystemen in der Hinsicht, dass Objekte anstelle von Tabellen gespeichert werden. Bisher konnten sich objektorientierte Datenbanksysteme auf dem kommerziellen Markt allerdings nicht durchsetzen. Gründe hierfür sind vor allem in der Dominanz und weiten Verbreitung relationaler Datenbanksysteme sowie in den explizit auf diese Datenbanksysteme ausgerichteten Anwendungen zu sehen. Die Umstellung eines Datenbanksystems auf ein neues Datenbanksystem mit geänderten Schnittstellen und neuer Datenbanksprache ist nicht umsetzbar. [Brink08] Ein weiterer Nachteil objektorientierter gegenüber relationaler Datenbanken wird bei der Verarbeitung großer Datenmengen und der damit verbundenen schlechten Performance deutlich. Weiterhin bieten vorhandene Geoinformationssysteme lediglich Schnittstellen zu relationalen und objektrelationalen Datenbanksystemen an.

Aus den beschriebenen Gründen ist ein flächendeckender Einsatz objektorientierter Datenbanksysteme in naher Zukunft nicht absehbar (vgl. [Brink08]). Die Alternative stellt der Einsatz *objektrelationaler Datenbanksysteme* dar. Als objektrelationale Datenbanksysteme werden relationale Datenbanksysteme bezeichnet, die um objektorientierte Konzepte erweitert wurden. Dazu zählen unter anderem die Modellierung von Relationen durch Klassen, Typkonstruktoren, die Objektidentität, Beziehungen, Methoden sowie die Vererbung.

3.2.1 Vererbung und Generalisierung

Eines der wesentlichsten Konzepte der Objektorientierung ist die *Vererbung*. Dieses Konzept soll nachfolgend kurz an einem einfachen Beispiel erläutert werden. Abbildung 3.6 beschreibt die Vererbungsbeziehung zwischen einer Oberklasse

„Gebäude“ und den Unterklassen „PrivatGebäude“ und „ÖffentlichesGebäude“ in einem einfachen UML-Klassendiagramm.

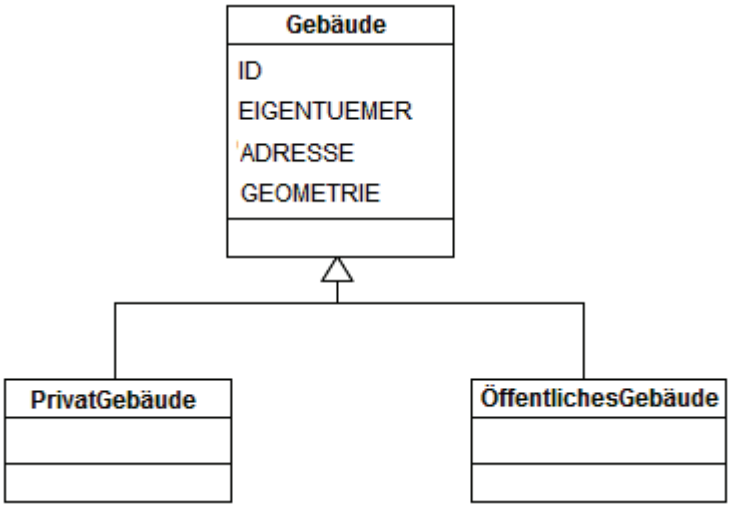

Abb. 3.6: Gebäude und Spezialisierungen in UML-Notation

Die Unterklassen übernehmen alle Attribute und Operationen der Oberklasse und können zudem zusätzliche Attribute und Operationen aufweisen. In diesem Zusammenhang spricht man oft auch von *Spezialisierung*, da die Unterklasse spezieller als die Oberklasse ist. Auf der anderen Seite stellt die Oberklasse eine *Generalisierung* der Unterklassen dar, da in dieser die gemeinsamen Eigenschaften der Unterklassen zusammengefasst werden.

Im vorliegenden Beispiel übernimmt ein Privat-Gebäude alle Attribute und Operationen der Klasse „Gebäude“ (ID, EIGENTUEMER, ADRESSE, GEOMETRIE) und grenzt sich gegenüber einem öffentlichen Gebäude durch speziell definierte Attribute und Operationen ab.

3.2.2 Objekte in objektrelationalen Datenbanken

Objektrelationale Datenbanksysteme unterscheiden sich von relationalen Datenbanksystemen durch die Unterstützung strukturierter Datentypen und benutzerdefinierter Klassen. Die gängigen relationalen Konzepte (vgl. Abschnitt 3.1: Relationale Datenbanksysteme) bleiben in objektrelationalen Datenbanksystemen erhalten. Lediglich die Atomarität von Attributen (1. Normalform) wird durch den Einsatz komplexer Datentypen aufgegeben.

3.2.3 Eignung als 4D-GIS

Objektorientierte und objektrelationale DBMS weisen einige Vorteile auf, die sie attraktiv für die Modellierung und die Speicherung geo-wissenschaftlicher Daten macht. Die DBMS besitzen die Fähigkeit zur Modellierung räumlicher und thematischer Daten in einem einheitlichen Datenmodell. Datenbanktypen und benutzerdefinierte Datentypen lassen sich eins zu eins in Beziehung setzen. Die benutzerdefinierten Datentypen können zur Beschreibung geometrischer Datentypen und zugehöriger räumlicher Zugriffsmethoden genutzt werden. Weitere Vorteile sind in der Vererbungstheorie sowie dem Polymorphismus zu sehen. Der Nachteil dieser Datenbanktechnologien ist allerdings im hohen Entwicklungsaufwand zu sehen. Entsprechende räumliche Datentypen sowie zugehörige geometrische Operationen müssen zunächst programmiert und implementiert werden. Des Weiteren gibt es sehr wenige Arbeiten im Bereich der temporalen Objektmodellierung und dementsprechend wenige prototypische Umsetzungen, sodass temporale Datentypen und zugehörige Operationen integriert werden müssen. Trotz des hohen Entwicklungsaufwands eignen sich Datenbanksysteme mit objektorientierten Konzepten sehr gut zur Modellierung von Phänomenen der realen Welt.

3.3 Geodatenbanksysteme

Als *Geodatenbanksysteme* (oft auch *räumliche Datenbanksysteme* genannt) werden Datenbanksysteme bezeichnet, die die Speicherung von Geodaten und die Bearbeitung räumlicher Anfragen unterstützen. Ein Geodatenbanksystem dient der Speicherung von Geoobjekten, die durch verschiedene Eigenschaften gekennzeichnet sind. [Brink08] Die Merkmale, die ein Geodatenbanksystem erfüllen muss, werden im Nachfolgenden näher erläutert.

3.3.1 Merkmale von Geodatenbanksystemen

Funktionsumfang eines Datenbanksystems

Ein Geodatenbanksystem baut stets auf einem Datenbanksystem auf, d.h. dass räumliche Informationen immer in Verbindung zu nicht-räumlichen Informationen

stehen. Ein Geodatenbanksystem muss demnach alle Kriterien eines Datenbanksystems bezüglich Datenmodellierung und Anfragebearbeitungen erfüllen. Ferner ist ein Geodatenbanksystem ein voll-funktionsfähiges Datenbanksystem mit zusätzlichen Fähigkeiten für den Umgang mit Geoobjekten. [Güt94]

Bereitstellung geometrischer Datentypen

Ein Geodatenbanksystem muss geometrische Datentypen (z.B. Punkte, Linien, Polygone) für die Modellierung der geometrischen Eigenschaften von Objekten der realen Welt bereitstellen.

Bereitstellung geometrischer Funktionen

Des Weiteren müssen Methoden für die geometrischen Datentypen zur Verfügung stehen, über die geometrische Funktionen ausgeführt werden können. Funktionen, wie z.B. die Bestimmung der Distanz zwischen Punkten oder Polygonen, können in der Anfragesprache des Datenbanksystems eingebettet werden. [Brink08]

Implementierung effizienter Algorithmen

Bei der Verarbeitung von Operationen mit Raumbezug ist es von großer Bedeutung effiziente Algorithmen zu implementieren, um geometrischen Operationen und die Suche nach Tabellendaten zu beschleunigen. Aufgrund dessen müssen Geoobjekte mit Hilfe räumlicher Indexe vom DBMS verwaltet werden. [Brink08]

Interoperabilität

Interoperabilität beschreibt die Möglichkeit geometrische Datentypen und Funktionen anderen Applikationen außerhalb des Geodatenbanksystems zugänglich zu machen. Dies setzt voraus, dass Syntax und Semantik der Daten dem Anwender in einheitlicher Form zur Verfügung gestellt werden [BZ01].

3.3.2 Objektrelationale Geodatenbanksysteme

Die in Abschnitt 3.3.1 beschriebenen Merkmale für Geodatenbanksysteme lassen sich durch entsprechende Erweiterungen auch auf objektrelationale Datenbanksysteme übertragen. Dazu müssen lediglich Klassen inklusive Methoden sowie Verfahren zur Anfragebearbeitung bereitgestellt werden. Als *objektrelationale Geodaten-*

banksysteme werden demnach objektrelationale Datenbanksysteme mit Spatial-Erweiterung bezeichnet. Als Beispiele objektrelationaler Geodatenbanksysteme sind zu nennen:

- IBM Informix mit Spatial Datablade
- IBM DB2 mit Spatial Extender
- PostgreSQL mit PostGIS
- MySQL
- Oracle mit Oracle Spatial

Für die Entwicklung eines geeigneten 4D-Datenmodells ist es daher sinnvoll, die 3D-Funktionalitäten, die kommerzielle 3D-Geodatenbanken anbieten, zu nutzen und um temporale Aspekte zu erweitern.

3.4 Temporaler Aspekt von Datenbanksystemen

Die in den vorherigen Abschnitten beschriebenen DBMS haben gemein, dass sie die temporalen Eigenschaften, denen Realweltobjekte unterliegen, nur in unzureichender Weise unterstützen. Es besteht zwar die Möglichkeit, Informationen über die Zeit in Form eines speziellen Datentyps (z.B. DATE) abzulegen und zu verarbeiten, allerdings kann dabei nur ein einziger Zustand der realen Welt beschrieben werden [Ste98]. Bei der Betrachtung von dynamischen 3D-Geoobjekten besteht aber die Notwendigkeit zeitabhängige Informationen, wie z.B. die Gültigkeit dieses Objekts, auswerten zu können. Da temporale Datenbanksysteme zumeist nur auf temporale Aspekte fokussiert sind und den Raumbezug nur unzureichend berücksichtigen, wäre es von Vorteil, nicht-temporale relationale oder objektrelationale Datenmodelle für die problemgerechte Verwaltung temporaler Daten anzupassen.

Einen ersten Ansatz zur Bewältigung dieses Problems bietet die temporale Erweiterung des Datenbanksprachstandards SQL-92, die als TSQL2 (Temporal SQL) bezeichnet wird. Darüber hinaus wurde *Applied* TSQL2 (ATSQL2) als Weiterentwicklung von TSQL2 bereits in verschiedenen Varianten implementiert. Hierbei ist v. a. das von STEINER [Ste98] in der Programmiersprache Java entwickelte TimeDB zu

nennen, das als Frontend auf ein herkömmliches relationales DBMS (z.B. Oracle, Sybase, Cloudscape) aufsetzt.

3.5 DBMS Oracle und TimeDB

Die erste prototypische Implementierung eines 4D-Datenmodells basiert auf zwei Datenbanktechnologien: Oracle und TimeDB. Als Marktführer[1] im Bereich DBMS bietet Oracle eine Vielzahl von Funktionen und Methoden zur Speicherung dreidimensionaler Geodaten. Die Spatial-Erweiterung von Oracle dient somit als Grundlage für die Implementierung eines dreidimensionalen geometrischen Datenmodells. Für die vollständige Beschreibung zeitabhängiger Daten bietet aber selbst die neuste Version von Oracle (11g) noch unzureichende Mittel. Daher erfolgt die zeitliche Beschreibung der Geodaten über die temporale Datenbank TimeDB, die die temporale Anfragesprache ATSQL2 unterstützt und somit zeitliche Daten beschreiben, manipulieren und abfragen kann. Die objektrelationalen Konzepte (Geometrietypen etc.) von Oracle Spatial lassen sich aber nicht ohne weiteres in TimeDB nutzen. Da die Modifizierung der TimeDB Module innerhalb des kurzen Bearbeitungszeitraums dieser Bachelorarbeit nicht möglich war, wurde ein benutzerdefiniertes relationales topologisches 3D-Modell in Oracle implementiert. Die Implementierung eines benutzerdefinierten topologischen Modells wäre ohnehin erforderlich gewesen, da dreidimensionale topologische Modell in keinem DBMS Standard sind.

Mit dem Hintergrund einer raum-zeitlichen Visualisierung in einem GIS oder einer anderen Graphiksoftware muss darüber hinaus eine Datenbankanbindung sichergestellt werden. Eine Vielzahl von Frontends (ESRI, AUTODESK) nutzen derweil standardmäßig das relationale DBMS Oracle, sodass diese Softwareprodukte für erste Visualisierungen genutzt werden können. In Anwendungen, in denen keine standardmäßige Datenbankanbindung zur Verfügung steht, kann diese programmiert werden (z.B. in Python oder Java).

[1] Die Oracle Database weist einen globalen Datenbank-Marktanteil von 48,6 % auf [Oracle07, Stand: 2007]

4 Grundlagen raum-zeitlicher Modellierung

Die zu modellierenden Eigenschaften von Geodaten werden in die vier Kategorien Geometrie, Topologie, Thematik und Dynamik unterteilt [BF94]. Die meisten Datenbank- und Geoinformationssysteme bilden Geoinformationen allerdings nur statisch ab, sodass temporale Aspekte zumeist nur unzureichend oder gar nicht berücksichtigt werden. In vielen GIS-Anwendungen besteht allerdings der Bedarf die Änderungen von Geoobjekten über einen definierten Zeitraum zu berücksichtigen. Konventionelle Datenbanksysteme werden diesen Anforderungen allerdings nicht gerecht, sodass neue Datenbankmodelle zur Abbildung zeitabhängiger Daten entwickelt werden müssen. Die nachfolgenden Abschnitte sollen zunächst einen kurzen Überblick in die Modellierung von Geodaten schaffen. Da es zurzeit kein geeignetes Modell gibt, das sowohl räumliche als auch temporale Eigenschaften vereint, werden diese beiden Schwerpunkte getrennt analysiert.

4.1 Modellierung räumlicher Daten

4.1.1 Eigenschaften von Geoobjekten

Wie bereits erwähnt sind für Geoobjekte die Eigenschaften Geometrie, Topologie, Thematik und Dynamik kennzeichnend. Im Nachfolgenden wird zunächst nur auf die ersten drei Eigenschaften eingegangen. Die temporalen Eigenschaften von Geoobjekten werden in Abschnitt 4.2 näher untersucht.

4.1.1.1 Geometrische Eigenschaften

Die Geometrie eines Geoobjekts kann durch die Angabe von Lage und Ausdehnung bzw. Form des Geoobjekts in einem eindeutig definierten räumlichen Bezugssystem beschrieben werden. Zur Abbildung der geometrischen Eigenschaften von Geoobjekten werden das Vektor- und das Rastermodell unterschieden. Vektormodelle bauen auf Vektoren, d.h. gerichteten geraden Strecken, auf, die durch die Angabe eines Anfangs- und Endpunktes in einem definierten Koordinatensystem

bestimmt sind. Träger der geometrischen Information ist der Punkt, der sich durch die Angabe von Koordinaten eindeutig beschreiben lässt. Auf dieser Grundlage können komplexere Geometrien wie Polygone gebildet werden.

Rastermodelle nutzen nicht den Punkt als Träger der Koordinateninformation, sondern gleichförmige, quadratische oder rechteckige Rasterzellen, die in der digitalen Bildverarbeitung auch als Pixel bezeichnet werden. Im dreidimensionalen Raum werden gleichförmige, kubische Zellen, sog. Voxel, genutzt. Jeder Pixel wird in Form eines Arrays durch Zeilen- und Spaltenindex eindeutig definiert. [Bart05, Brink08, Lange02]

4.1.1.2 Topologische Eigenschaften

Die Topologie beschreibt die relative räumliche Beziehung von Geoobjekten zueinander, wobei von der Geometrie abstrahiert wird. Die topologische Sichtweise wird gerne am Beispiel der auf einem Luftballon eingezeichneten Straßenkarte beschrieben. Durch Hineinpumpen oder Ablassen von Luft verändert sich zwar die Geometrie der Elemente, die topologische Beziehung der Elemente zueinander bleibt hingegen unverändert. Zu den wichtigsten topologischen Konzepten zählen Nachbarschaften, Überlagerungen bzw. Überschneidungen oder Teilmengenbeziehungen. [Bart05, Brink08, Lange02]

4.1.1.3 Thematische Eigenschaften

Die Thematik eines Geoobjekts kann durch die Angabe von Sachattributen beschrieben werden. Dabei wird zwischen vier verschiedenen Skalenniveaus unterschieden:

- Nominalskala (z.B. Namen, Postleitzahlen)
- Ordinalskala (z.B. Ränge, Bewertungsstufen)
- Intervallskala (metrische Daten mit nicht eindeutigem Nullpunkt: z.B. °C)
- Verhältnisskala (metrische Daten mit eindeutigem Nullpunkt: z.B. Alter)

Zur Beschreibung verschiedener Thematiken von Geoobjekten in einem GIS wird das sog. Layerprinzip genutzt, bei dem die verschiedenen thematischen Eigenschaften der Geoobjekte in unterschiedlichen Ebenen gehalten werden. [Bart05, Brink08, Lange02]

4.1.2 Geometrische Datenmodelle

Zur Darstellung räumlicher Daten in einem 3D-GIS werden verschiedene räumliche Repräsentationen unterschieden, die in Tabelle 4.1 gegenübergestellt sind.

Tab. 4.1: Räumliche Repräsentationen in verschiedenen Dimensionen [Breu05]

	Kantenmodell	**Flächenmodell**	**Volumenmodell**
Dimension	1D	2D	3D
Räumliche Repräsentationen	Wireframe	Vektor-Rand-Repräsentation	Zellzerlegung Enumerationsverfahren Funktions-Randrepräsentation Sweep Repräsentation Parametrisierte Repräsentation Constructive Solid Geometry

Die dreidimensionalen räumlichen Repräsentationen können in vier verschiedene Repräsentationsarten mit differenzierten Ansätzen unterteilt werden, die in den nachfolgenden Abschnitten näher erläutert werden.

4.1.2.1 Vektorbasierte Repräsentationen

Bei vektorbasierten Repräsentationen wird grundsätzlich zwischen *Kanten*- bzw. Wireframe-Repräsentation und *Randrepräsentation* unterschieden. Im Kantenmodell werden Körper oder Oberflächen einzig aus Linien oder Kurven zusammengesetzt. Die Darstellung des Objekts wirkt auf den Nutzer wie ein Drahtmodell (engl. *wireframe*). Es können daher keine Informationen über Flächen oder Volumen ausgedrückt werden, sodass diese Repräsentationsform für ein 3D-GIS eher ungeeignet ist.

Die *Randrepräsentation* (*boundary representation*) ist eine Beschreibungsmethode, in der ein Objekt aus sich begrenzenden Geometrien besteht. Die zur Begrenzung eines Körpers genutzten topologischen Primitive sind Knoten, Kanten und Flächen. Diese Form der Repräsentation wird auch als *Vektor-Randrepräsentation* bezeichnet. Ferner gibt es eine weitere Repräsentationsart, die analytische Funktionen für die Beschreibung einer Fläche nutzen. Die als *Funktions-Randrepräsentation* bezeichnete Form wird in Abschnitt 4.1.2.3 näher betrachtet. [BF94, Breu05]

Vektor-Randrepräsentation (VRR)
Mithilfe der Vektor-Randrepräsentation können beliebige Polyeder beschrieben werden. Eine komplexe Geometrie kann hierarchisch in Form von Flächen, Kanten und Knoten aufgebaut werden, wobei die Koordinaten der Knoten als Träger der geometrischen Information dienen. Durch die explizite Separation von Geometrie und Topologie müssen bei Transformationen lediglich die Koordinaten geändert werden, da die Topologie infolge einer Transformation unverändert bleibt. Nachteil der Vektor-Randrepräsentation ist, dass keine topologischen Beziehungen und Informationen zum Volumeninhalt vorhanden sind. [Breu05]

4.1.2.2 Zerlegende Repräsentationen

Diese Form der Repräsentation zerlegt den Raum in sich nicht überlappende geometrische Primitive. Es werden prinzipiell zwei verschiedene Formen unterschieden: das Verfahren der *Zellzerlegung* und das *Enumerationsverfahren*.

Zellzerlegung (Cell-Decomposition)
Komplexe Objekte werden aus einfachen räumlichen Geometrien unterschiedlicher Form und Größe zusammengesetzt. Die einzelnen Zellen bestehen normalerweise aus einem Satz vordefinierter, parametrisierter Zelltypen (z.B. Würfel, Tetraeder, Zylinder etc.). Bei der Zusammensetzung der geometrischen Zellen ist eine Überschneidung nicht zulässig. Der Vorteil des Verfahrens besteht in einer genauen Annäherung beliebiger Körper. Als Nachteil wird die fehlende topologische Information der Primitive angesehen. [BF94, Breu05, Kada]

Enumerationsverfahren (Spatial Occupancy Enumeration)
Das *Enumerationsverfahren*, als Spezialfall der Zellzerlegung, unterteilt eine Geometrie in identische, sich nicht überlappende und eindeutig adressierbare Zellen. Als Primitive werden zumeist Würfel verwendet. Der Definitions- und Wertebereich ist abhängig von der Größe und Form der Raumzellen. Die Adressierbarkeit der Zellen hat den Vorteil, dass die geometrischen Primitive räumlich indexiert werden können, was zu einer Beschleunigung der Suche nach Geometrien führt. Des Weiteren lassen sich topologische Beziehungen leicht berechnen. Auf der anderen Seite lassen sich Geometrien durch die Festlegung von identischen Primitiven nur grob approximieren. [BF94, Breu05]

4.1.2.3 Analytische Repräsentationen

Analytische Repräsentationen nutzen Funktionen und Parameter der analytischen Geometrie zur Beschreibung der Oberfläche oder des Volumens von Körpern. Dabei werden drei verschiedene Ansätze unterschieden.

Funktions-Randrepräsentation (FRR)
Bei der *Funktions-Randrepräsentation* werden Körper nicht durch topologische Primitive, sondern durch analytische Funktionen beschrieben. Dadurch lassen sich geometrische Eigenschaften sehr schwer nachweisen. [Breu05]

Sweep-Repräsentation
Die *Sweep-Repräsentation* basiert auf dem Gedanken, einen Punkt, eine Kurve oder eine Fläche entlang einer definierten Achse zu verschieben oder zu drehen. Dabei wird zwischen Translations- und Rotationssweep sowie einer hybriden Darstellung der beiden Sweeps differenziert. Die Sweep-Modelle eignen sich v. a. zur Darstellung vordefinierter Freiformen. Nachteilig ist, dass der Rechenaufwand, der zur Berechnung geometrischer Operationen benötigt wird, sehr hoch ausfällt. [Breu05, Foley97, Hsu09]

Parametrisierte Repräsentation (Primitive Instancing)
Bei der in der englischen Literatur als *Primitive Instancing* bezeichneten Repräsentation wird die Geometrie eines Objekts durch eine feste Anzahl von Parametern

(z.B. Länge, Breite, Höhe, Tiefe, Radius) charakterisiert. Jedes Objekt kann so mithilfe dieser Parameter beschrieben werden. Ziel der parametrisierten Repräsentation ist es, komplexe Objekte aus simplen primitiven Formen zu konstruieren. Die geometrischen Primitive sind durch mathematische Gleichungen definiert, die den Körper beschreiben. Dadurch lassen sich einfach fest vergebene Geometriefamilien modellieren, da jedes Element aus einer festen Anzahl von Parametern beschrieben werden kann. Der Nachteil der Methode besteht darin, dass komplexe Geometrien nicht aus einfachen Primitiven zusammengesetzt werden können, sondern durch einen Satz von Parametern modelliert werden müssen. [BF94, Breu05, Foley97]

4.1.2.4 Constructive Solid Geometry (CSG)

Das CSG-Modell basiert auf der Grundlage, dass ein physikalisches Objekt in einen Satz von Basisprimitiven zerlegt werden kann, die durch Bool'sche Operationen (Mengenoperationen und Transformationen) miteinander kombiniert werden, um das Objekt zu formen. Ein Raumprimitiv besteht aus einem Satz von Oberflächen. Im Gegensatz zur Randrepräsentation werden Knoten, Kanten und Flächen nicht explizit gespeichert. Die Primitive werden mithilfe eines Algorithmus berechnet. Jedes Objekt ist in Form eines binären Baumes repräsentierbar, der aus Raumprimitiven und Operationen besteht. Die hybride CSG eignet sich aufgrund einer einfachen Modellierungssprache, die sich aus dem Konstruktionsbaum ableiten lässt, sehr gut zur Modellierung künstlicher Objekte. Freiformflächen, die zur Beschreibung natürlicher Objekte herangezogen werden, sind hingegen schwieriger zu beschreiben. [BF94, Breu05, Hsu09]

4.1.2.5 Vergleich der Repräsentationen

Im Folgenden werden die untersuchten räumlichen Repräsentationen miteinander verglichen und anhand ausgesuchter Kriterien für die Eignung als 3D-GIS untersucht. Die Auswahl der Kriterien erfolgte nach BREUNIG in [Breu05]. Das Ergebnis seiner Untersuchungen ist in Tabelle 4.2 dargestellt.

Tab. 4.2: Vergleich der 3D-Repräsentationen nach BREUNIG in [Breu05]

	Kantenmodell	Sweep-Repräsentation	Parametrisierte Repräsentation	CSG	VRR / FRR	Enumerations-verfahren	Zellzerlegung
Definitionsbereich	Keine ~[1]	Keine ~[2]	Keine ~[3]	Abh. ~[4]	Nur ~[5]	Alle ~[6]	Alle ~[6]
Gültigkeit - geometrische - fachliche	 Nein Nein	 Ja Nein	 Ja Ja	 Ja Nein	 Ja Nein	 Ja Ja	 Ja Nein
Nicht-Mehrdeutigkeit Eindeutigkeit	Nein Nein	Ja Nein	Ja Ja	Ja Nein	Ja Nein	Ja Ja	Ja Nein
Abgeschlossenheit	Nein	Ja	Ja	Ja	Ja	Ja	Ja
Effizienz	+	-	-	-	+ / -	+	+
Genauigkeit	-	++	++	++	- / ++	-	+
Speicherbedarf	Gering	Sehr gering	Sehr gering	Sehr gering	Hoch / gering	Sehr hoch	Sehr hoch

[1] Flächen und Volumina, [2] unregelmäßigen Objekte, [3] Freiformen, [4] von Primitiven,
[5] (Ober-)Flächen, keine Volumina, [6] approximierten Objekte

Das Kantenmodell ist für die Verwendung in einem 3D-GIS offenbar nicht geeignet. Größter Nachteil dieser Repräsentation ist, dass weder Flächen- noch Volumeninformationen über ein Objekt vorliegen. Der Definitionsbereich der Sweep-, der parametrisierten- und der CSG-Repräsentation ist ebenso beschränkt wie bei der Kantendarstellung. Die CSG eignet sich jedoch sehr gut zur Visualisierung einfacher 3D-Geometrien bei eingeschränkter Anzahl von Primitiven. Die Randpräsentationen eignen sich sehr gut zur Darstellung von (Ober-)flächen, aufgrund fehlender Volumeninformationen jedoch nicht zur Modellierung von Körpern. [Breu05]

Im nächsten Abschnitt werden verschiedene topologische Datenmodelle vorgestellt, die auf Basis einer Randflächenrepräsentation modelliert worden sind. Diese Form der Darstellung hat den Vorteil, dass bei raum-zeitlichen Veränderungen, denen Geoobjekte unterliegen können, lediglich der Punktdatenbestand geändert werden

muss. Die topologischen Beziehungen bleiben hingegen unverändert oder können bei fehlender Gültigkeit eines Punktes dynamisch aktualisiert werden.

4.1.3 Topologische Datenmodelle für 3D-Geodatenbanken

In einem GIS werden vier wesentliche Komponenten unterschieden: die Erfassung, die Verarbeitung, die Analyse sowie die Präsentation von Geodaten (EVAP, vgl. Abschnitt 2.1: Begriff Geoinformationssystem). Der effizienten Analyse raumbezogener Daten, insbesondere die räumlichen Ausprägungen von Geoobjekten und deren Beziehung zueinander, kommt eine besondere Bedeutung zu. Die Entwicklung topologischer Modelle verfolgt zwei wesentliche Ziele: zum einen die Performanz topologischer Abfragen und zum anderen die Erhaltung der Datenintegrität [Coors05]. Geometrische Modelle haben den Nachteil, dass gleiche Punkte redundant gehalten werden, sodass bei Änderungen des Punktdatenbestands alle redundanten Speicherungen dieses Punktes aktualisiert werden müssen. Bei topologischen Datenmodellen wird jeder Punkt hingegen nur einmal gespeichert. Des Weiteren können räumliche Beziehungen von Objekten zueinander direkt abgeleitet werden, während bei geometrischen Modellen aufwändige Schnittberechnungen durchgeführt werden müssen.

4.1.3.1 3D-FDS nach Molenaar

Die 3D Formal Data Structure (3D-FDS) ist eine Erweiterung des vektorbasierten Modells (Single Vector Value Map, svvm), das MOLENAAR [Mol90] für die Verwaltung thematischer, topologischer und zweidimensionaler geometrischer Daten entwickelte, um die dritte Dimension. Als Grundbestandteile für die Repräsentation geometrischer Primitive dienen die Elemente Knoten (Node), Kante (Arc) und Fläche (Face). Zur Modellierung der dritten Dimension wurden zusätzlich die Primitive Halbkante (Edge) und Körper (Body) eingeführt. Die Halbkanten haben dabei zwei Aufgaben. Zum einen definieren sie die Grenze zu einer Fläche und zum anderen dienen sie zur Orientierung der Flächen, um zwischen linken und rechten Körper zu unterscheiden. [Coors05, ZRS02] Abbildung 4.1 beschreibt die 3D-FDS nach MOLENAAR.

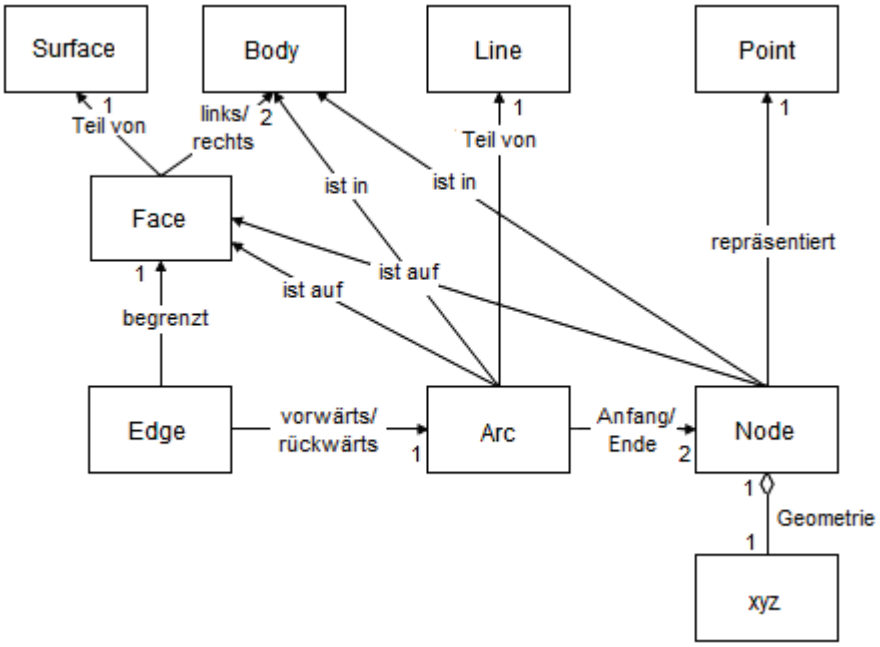

Abb. 4.1: Konzeptionelles Datenmodell 3D-FDS nach MOLENAAR [Mol90]

Zwischen diesen Primitiven bestehen verschiedene 1:n-Beziehungen. Dabei sind folgende topologische Zusammenhänge erkennbar:

- Jede Kante (Arc) besitzt genau einen Anfangs- und einen Endknoten (Node).
- Ein Knoten kann Bestandteil mehrerer Kanten sein.
- Eine Fläche (Face) kann maximal zwei Körper (Body) begrenzen, während ein Körper aus mehreren Flächen bestehen kann.
- Es bestehen Verweise zwischen Knoten und Kante zu der Fläche bzw. zum Körper, auf dem diese sich befinden.
- Flächen und Körper bestehen aus mehreren Knoten oder Kanten.

4.1.3.2 3D-GIS-Datenmodell nach Flick

Eine Erweiterung der 3D-FDS nach MOLENAAR wurde von FLICK [Flick99] konzipiert. Das Konzept des Datenmodells lässt sich am Einfachsten aus Abbildung 4.2 ableiten. Das Modell besteht aus vier Basisprimitiven (0- bis 3-Zellen) zur Verwaltung topologischer und geometrischer Beziehungen. Dabei entsprechen:

- 0-Zellen Points (Punkten)
- 1-Zellen Lines (Linien)
- 2-Zellen Surfaces (Oberflächen)
- 3-Zellen Bodies (Körper).

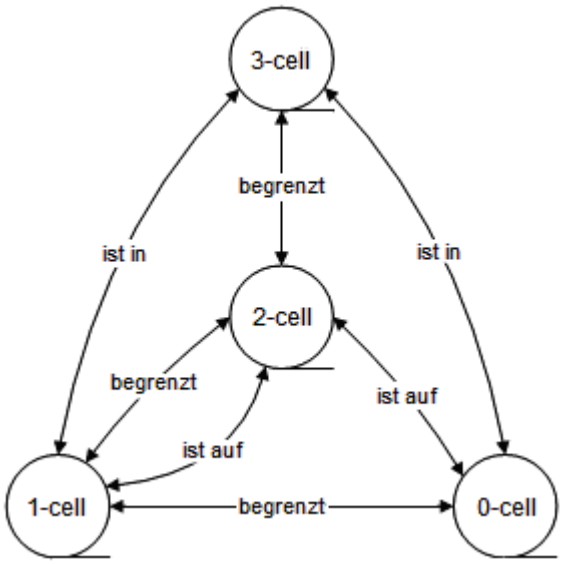

Abb. 4.2: 3D-GIS-Datenmodell nach FLICK [Flick99]

Die Beziehungen der Zellen zueinander („begrenzt“, „ist in“ oder „ist auf“ Beziehungen) werden ohne Einschränkung bidirektional gespeichert. Weiterhin gelten folgende Regeln:

- *n*-Zellen dürfen sich nicht selbst oder mit anderen *n*-Zellen schneiden (*n*=1,2,3).
- Weder 0- noch 1-Zelle haben gleichzeitig eine „ist in“ oder „ist auf“ Beziehung.

Die Zuordnung zwischen geometrischen und topologischen Primitiven sowie konkreten Geoobjekten wird über Assoziationen realisiert. Das Konzept ist sehr flexibel. So können einem Geoobjekt verschiedene Visualisierungsdaten über ein View-Konzept unabhängig von den geometrischen Daten zugeordnet werden. [Coors05]

4.1.3.3 Urban Data Model nach Coors

Das Urban Data Model (UDM) nach COORS stellt eine wesentliche Vereinfachung der 3D-FDS nach MOLENAAR dar. So wird zum einen auf die explizite Speicherung von Kanten verzichtet und zum anderen werden Flächen und Linien nur über Eckpunkte repräsentiert. Durch diese Einschränkung kann die Anzahl der zu speichernden Objekte erheblich reduziert werden. Abbildung 4.3 beschreibt das topologische Datenmodell in UDM. [Coors05, ZRS02]

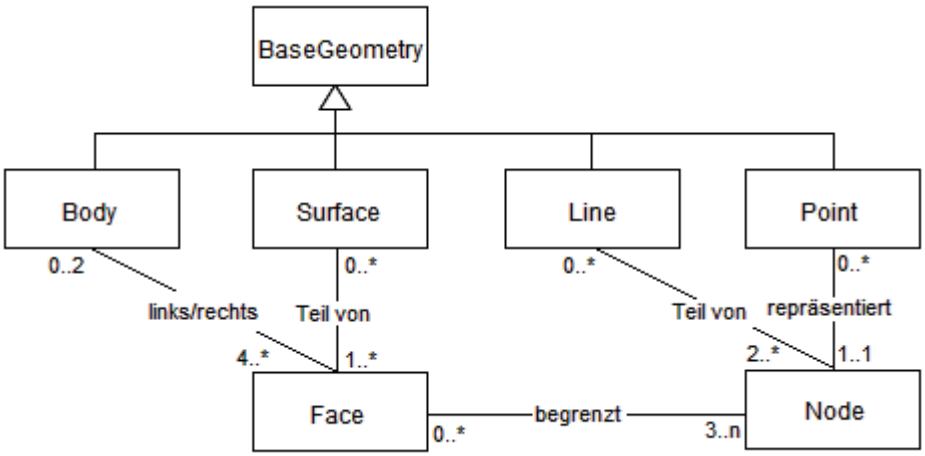

Abb. 4.3. Topologisches Datenmodell in UDM [Coors05]

Folgende topologische Zusammenhänge bestehen zwischen den einzelnen Primitiven:

- Ein Punkt (Point) wird genau durch einen Knoten (Node) repräsentiert.
- Eine Linienzug (Line) besteht minimal aus zwei Knoten.
- Eine Fläche (Face) wird mindestens durch drei Knoten begrenzt.
- Eine Fläche (Face) kann maximal zwei Körper (Body) begrenzen, während ein Körper aus minimal vier Flächen besteht.

4.1.3.4 Vergleich der topologischen Modelle

Nachfolgend sollen die wichtigsten Eigenschaften sowie Vor- und Nachteile der drei untersuchten Modelle tabellarisch zusammengefasst werden. Das Ergebnis ist in Tabelle 4.3 dargestellt.

Tab. 4.3: Vergleich der untersuchten topologischen Modelle [Coors05, WWSL, ZRS02]

	3D-FDS	**3D-GIS-Datenmodell**	**UDM**
Modellbasis	svvm	3D-FDS	Triangulation
Primitive	Node Arc Edge Face	0-cell 1-cell 2-cell 3-cell	Node Face (Triangle)
Vorteile	Einfache Verknüpfung topologischer Primitive mit thematischen Daten	Effiziente und flexible Bearbeitung topologischer Anfragen	Effiziente Speicherung polygonaler Objekte Vollständige Unterstützung topologischer Relationen Einfache Visualisierung 50% geringeres Datenvolumen als 3D-FDS
Nachteile	Komplexe räumliche Objekte schwer darstellbar Hohes Datenvolumen zur Speicherung von Polyedern	Komplexe räumliche Objekte schwer darstellbar Hohes Datenvolumen durch redundante Speicherung von Relationen	Dynamische Aktualisierung komplex Durch implizite Speicherung von Kanten können linienförmige Objekte nur über Strecken definiert werden

Aus den in Tabelle 4.3 aufgeführten Gründen erweist sich eine Implementierung des Urban Data Model nach COORS [Coors05] in eine relationale Datenbank als sinnvoll. Durch die Restriktion auf zwei Primitive beschränkt sich der Modellraum auf eine Randflächendarstellung von Polyedern. Dies hat aber den Vorteil, dass sich das Modell effizient in eine Tabellenstruktur überführen lässt und das Datenvolumen auf ein Minimum reduziert wird. Das geringere Datenvolumen hat wiederum eine schnellere Visualisierung der Geometrien zur Folge.

4.2 Modellierung zeitlicher Daten

4.2.1 Dynamik von Geoobjekten

Geoobjekte können sich hinsichtlich ihrer Thematik, Geometrie und Topologie im zeitlichen Verlauf verändern. Ein einfaches Beispiel für die thematische Veränderung im Zeitverlauf ist die Beobachtung der Temperaturen für ein ausgesuchtes Gebiet über einen definierten Zeitraum. Es ist davon auszugehen, dass sich die Temperaturen über einen bestimmten Zeitraum ändern, die Lage und die Ausdehnung des untersuchten Gebietes aber unverändert bleibt. Es besteht also lediglich eine *zeitliche Variabilität* hinsichtlich der Thematik.

Auf der anderen Seite können sich Geometrie und Topologie von Geoobjekten im Zeitverlauf ändern. Die als *raum-zeitliche Variabilität* bezeichnete Eigenschaft von Geoobjekten ist allerdings weit aus komplizierter zu erfassen und in bisher entwickelten Geoinformationssystemen nur schwer darstellbar. [Bart05, Lange02]

4.2.2 Grundbegriffe

4.2.2.1 Zeitdimensionen

In nicht-temporalen Datenbanken besteht die Möglichkeit die Zeit mithilfe eines speziellen Datentyps zu beschreiben. Es wird allerdings nur ein bestimmter Zustand der realen Welt beschrieben. Diese Form wird als *benutzerdefinierte Zeit* (engl. *user-defined time*) bezeichnet, da das beschriebene Attribut nur vom Benutzer gedeutet werden kann und vom DBMS lediglich als weiteres Attribut interpretiert wird. Es besteht demnach keine Möglichkeit der temporalen Auswertung von Informationen. Diese Restriktion führte zur Entwicklung von temporalen Datenmodellen, in denen zwei Dimensionen der Zeit von Bedeutung sind: zum einen die *Gültigkeitszeit* (engl. *valid time*) und zum anderen die *Transaktionszeit* (engl. *transaction time*). Als Gültigkeitszeit wird der Zeitraum bezeichnet, in dem ein Objekt der realen Welt den beschriebenen Zustand aufweist, unabhängig vom Aufzeichnungszeitpunkt in der Datenbank. Die Transaktionszeit beschreibt den Zeitraum, zu dem eine

Information der realen Welt in der Datenbank dokumentiert wurde. [JSS94, Ste98, ZCFS97]

Je nach Unterstützung dieser beiden Komponenten werden die folgenden temporalen Datenbanksysteme unterschieden [Ste98]:

- Schnappschuss Datenbank: Unterstützung einer benutzerdefinierten Zeit
- Historische Datenbank: Unterstützung der Gültigkeitszeit
- Rollback Datenbank: Unterstützung der Transaktionszeit
- Bitemporale Datenbank: Unterstützung von Gültigkeits- und Transaktionszeit

Für die Analyse von zeitabhängigen 3D-Geoobjekten ist es daher erforderlich, sowohl die Gültigkeits- als auch die Transaktionszeit zu unterstützen, um einerseits die Gültigkeit eines Objektes festzuhalten und andererseits den Zeitpunkt, zu dem das Objekt in der Datenbank abgelegt wurde, zu erfassen. Die beiden Grundtypen bilden die **temporale Historie** eines Objekts.

4.2.2.2 Darstellung der Zeit

In [ZCFS97] wurden verschiedene Möglichkeiten beschrieben, den Zeitverlauf darzustellen. Der zeitliche Verlauf lässt sich demnach als *linear*, *sublinear* oder *verzweigend* (*branching*) klassifizieren (**temporale Ordnung**). Dem linearen und sublinearen Modell liegt zugrunde, das die Zeit von der Vergangenheit bis zur Zukunft in einer linear geordneten Weise verläuft. Der Unterschied besteht lediglich darin, dass sich die Zeitintervalle in der linearen Ordnung nicht überlappen dürfen, im sublinearen Fall jedoch keine Einschränkung diesbezüglich vorliegt. Das verzweigende Modell wird bis zu einem bestimmten Zeitpunkt als linear betrachtet. Ist dieser Zeitpunkt erreicht, sind unterschiedliche Zeitlinien zulässig, mit denen verschiedene mögliche Ereignisse beschrieben werden können.

Die Zeit im linearen Modell kann durch drei verschiedene Dichten (*stetig*, *diskret* oder *dicht*) beschrieben werden. Stetige Modelle (*continuous model*) betrachten die Zeit als isomorph zu realen Zahlen, d.h. dass jede reale Zahl einem Zeitpunkt ent-

spricht und somit lückenlos darstellbar ist. Ein Zeitmodell wird als dicht (*dense model*) bezeichnet, wenn die Zeit isomorph zu rationalen Zahlen dargestellt wird, d.h. dass sich zwischen zwei Zeitpunkten ein weiterer Zeitpunkt finden lässt. Als diskret (*discret model*) wird ein Zeitmodell bezeichnet, wenn zwischen einem Zeitpunkt und seinem Nachfolger kein weiterer Zeitpunkt existiert. [Ste98]

Die Zeit kann auf einer Zeitachse durch eine beliebige Anzahl von Zeitpunkten fester Länge repräsentiert werden. Die kleinste, nicht zerlegbare Zeiteinheit auf der Zeitachse (Periode) wird auch als *Chronon* bezeichnet. Die Länge des Chronons bzw. die Dauer einer Periode wird durch die *Granularität* bestimmt (siehe Abbildung 4.4). Im normalen Sprachgebrauch spricht man allerdings eher von den zeitlichen Primitiven *Zeitpunkt* (*instant*) und *Zeitraum* (*time period* oder *period*), die die **temporale Struktur** bilden. Ein Zeitraum wird dabei durch zwei Zeitpunkte begrenzt.

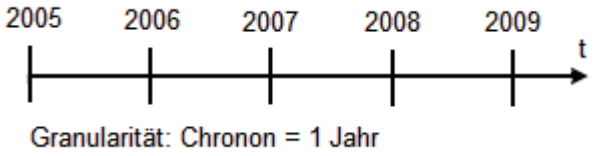

Abb. 4.4: Darstellung einer Granularität im linearen Zeitmodell

Ein *Kalender* wird durch die Einteilung in geeignete Abschnitte definiert und stellt bestimmte Granularitäten zur Verfügung. Ein Kalender bildet die **temporale Repräsentation**. Im Gregorianischen Kalender, der in der Norm ISO 8601 spezifiziert ist, werden die Granularitäten Jahre, Monate, Tage, Stunden, Sekunden sowie Bruchteile von Sekunden unterschieden.

4.2.3 TSQL2 Datenmodell

TSQL2 ist ein bitemporales konzeptionelles Datenmodell (BCDM), das sowohl die Gültigkeits- als auch die Transaktionszeit unterstützt. Für beide zeitlichen Dimensionen wird vorausgesetzt, dass es sich um lineares, diskretes und begrenztes Zeitmodell handelt. Des Weiteren wird angenommen, dass die beiden Zeitdimensionen absolut sind. Die Zeit wird durch die zeitlichen Primitive Zeitpunkt und Zeitraum beschrieben, wobei der Zeitraum durch zwei Zeitpunkte an beiden Enden begrenzt wird. Die Zeitinformationen werden in Form von Zeitstempeln gespeichert. Die tem-

poralen Intervalle können dabei nach rechts hin offen sein, sofern der Endzeitpunkt nicht bekannt ist. In diesen Fällen wird der künstliche Zeitpunkt forever genutzt, sodass das Objekt bis auf weiteres gültig ist. Des Weiteren wird der künstliche Zeitpunkt now unterstützt, bei dem die Gültigkeit eines Objekts stetig mit der aktuellen Systemzeit anwächst. [SBJS00]

Darüber hinaus unterstützt TSQL2 ALLEN's 13 Beziehungsoperatoren [Allen83], über die Zeitintervalle relativ miteinander in Beziehung gesetzt werden können. Dabei müssen sowohl Start (i_1)- und Endpunkt (i_2) gegeben sein, da die Beziehung der Intervalle über die Lage von Start- und Endpunkt zueinander aufgebaut wird. Die möglichen temporalen Beziehungen sind mathematisch in Tabelle 4.4 und graphisch in Abbildung 4.5 dargestellt.

Tab. 4.4: Temporale Beziehungen nach ALLEN [Allen83]

Temporale Vergleichsoperatoren	**Definition über Intervallendpunkte**
i_1 before i_2	end (i_1) < begin (i_2)
i_1 after i_2	end (i_2) < begin (i_1)
i_1 during i_2	(begin(i_1) > begin(i_2) $\cap$ end(i_1) $\leq$ end(i_2)) $\cup$ (begin(i_1) $\geq$ begin(i_2) $\cap$ end(i_1) < end(i_2))
i_1 contains i_2	(begin(i_2) > begin(i_1) $\cap$ end(i_2) $\leq$ end(i_1)) $\cup$ (begin(i_2) $\geq$ begin(i_1) $\cap$ end(i_2) < end(i_1))
i_1 overlaps i_2	begin(i_1) < begin(i_2) $\cap$ end(i_1) > begin(i_2) $\cup$ end(i_1) < end(i_2)
i_1 overlapped_by i_2	begin(i_2) < begin(i_1) $\cap$ end(i_2) > begin(i_1) $\cup$ end(i_2) < end(i_1)
i_1 meets i_2	end(i_1) = begin(i_2)
i_1 met_by i_2	end(i_2) = begin(i_1)
i_1 starts i_2	begin(i_1) = begin(i_2) $\cap$ end(i_1) < end(i_2)
i_1 started_by i_2	begin(i_1) = begin(i_2) $\cap$ end(i_2) < end(i_1)
i_1 finishes i_2	begin(i_1) > begin(i_2) $\cap$ end(i_1) = end(i_2)
i_1 finished_by i_2	begin(i_2) > begin(i_1) $\cap$ end(i_1) = end(i_2)
i_1 equals i_2	begin(i_1) = begin(i_2) $\cap$ end(i_1) = end(i_2)

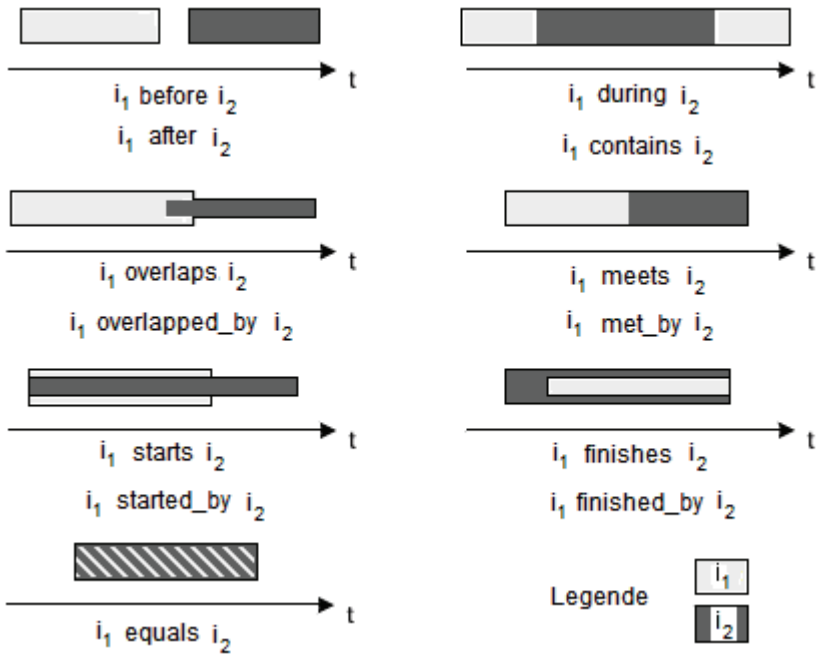

Abb. 4.5: Graphische Repräsentation der temporalen Beziehungen nach ALLEN

Des Weiteren werden verschiedene Kalender unterstützt, z.B. der Gregorianische Kalender oder der Mondkalender. Es besteht weiterhin die Möglichkeit Zeitpunkte in unterschiedlicher Weise darzustellen und zu speichern:

```
DATE '2009-09-05'
DATE 'September 5, 2009'
DATE '05/09/09'
```

Nicht zuletzt können verschiedene Granularitäten über die Operatoren SCALE und CAST spezifiziert werden:

```
SCALE (DATE '2009-09-05' AS YEAR) = '2009'
SCALE (DATE '2009-09-05' AS MONTH) = '2009-09'
SCALE (DATE '2009-09-05' AS DAY) = '2009-09-05'
```

4.2.4 Zeitmodell nach ISO 19108

Als Bestandteil der Normenfamilie ISO/TC 211 *Geographic Information* beschreibt die Norm ISO 19108 das temporale Schema von Geoinformationen. Das Schema

besteht aus zwei Paketen. Das Paket „Temporal Objects“ definiert temporale geometrische und topologische Primitive, während das Paket „Temporal Reference System“ Elemente für die Beschreibung temporaler Bezugssysteme bereitstellt. Gemäß ISO 19108 *Geographic Information – Temporal Schema* werden die nachstehenden zeitlichen Formen unterschieden [ISO19108]:

Aktualitätsstempel (engl. *time stamp*)
Es wird untersucht, ob ein Geoobjekt gültig ist, gültig war oder in der Zukunft eine Gültigkeit aufweist.

Ereignis (engl. *event*)
Ein Ereignis entspricht einem Punkt auf der Zeitachse, eine Aktion, die sich zu einem bestimmten Zeitpunkt ereignet. Dabei können sich sowohl Geometrie, Topologie und Thematik des Geoobjekts ändern.

Zustand (engl. *state*)
Ein Zustand entspricht einem Intervall auf der Zeitachse, in dem ein Objekt hinsichtlich seiner Eigenschaften unverändert bleibt.

Dynamisches Verhalten
Das dynamische Verhalten beschreibt die Veränderung eines Objekts mit fortschreitender Zeit.

4.2.4.1 Temporale Objekte

Die Zeit ist als Dimension - analog zu jeder anderen räumlichen Dimension - zu betrachten. So weist die Zeit ebenso eine Geometrie und Topologie auf wie der Raum selbst. Das Zeitmodell baut auf zwei temporalen Primitiven unterschiedlicher Dimension auf [ISO19108]:

Instant
TM_Instant bezeichnet ein 0-dimensionales temporales Primitiv, das einen *Zeitpunkt* repräsentiert. Der Zeitpunkt ist gleichbedeutend mit einem Punkt im Raum zu sehen.

Period

TM_Period ist ein 1-dimensionales temporales Primitiv, das einen *Zeitraum* repräsentiert. Der Zeitraum wird von zwei Zeitpunkten (BEGINNING und ENDING) begrenzt, wobei BEGINNING vor ENDING liegen muss.

Darüber hinaus werden Operationen zur Berechnung der relativen Ordnung zwischen zwei temporalen Primitiven spezifiziert [ISO19108]:

RelativePosition

Der Datentyp TM_RelativePosition gibt Operationen zur Berechnung der relativen Position zwischen zwei Zeitprimitiven vor. Darunter sind u. a. die in Abschnitt 4.2.3 beschriebenen Vergleichsoperatoren nach ALLEN enthalten.

Duration

Der Datentyp TM_Duration wird herangezogen, um die Dauer eines Zeitraums oder den zeitlichen Abstand zwischen zwei Zeitpunkten zu repräsentieren.

Beispiel: Eine Dauer von 4 Tagen, 7 Stunden und 20,8 Minuten wird in der Form P4DT7H20.8 dargestellt.

4.2.4.2 Temporale Bezugssysteme

Die Zeit kann nicht absolut gemessen werden, sondern nur relativ zu einem vordefinierten temporalen Bezugssystem. Die Norm ISO 8601 spezifiziert die Angabe des Gregorianischen Kalenders und der koordinierten Weltzeit UTC (Universal Time Coordinated) oder einer lokalen Zeit im 24h-Format.

Das Paket „Temporal Reference Systems“ umfasst drei verschiedene Typen von zeitbezogenen Bezugssystemen [ISO19108]:

Kalender und Uhrzeit (calendar and clock)

Kalender und Uhrzeit beruhen beide auf einer Intervallskala. Ein Kalender ist ein diskretes temporales Bezugssystem, das einen Zeitpunkt mit einer Genauigkeit von

einem Tag definieren kann. Für die vollständige Beschreibung einer temporalen Position für einen bestimmten Tag wird daher die Angabe einer Uhrzeit benötigt.

Zeitkoordinaten (Temporal coordinate systems)

Bei Zeitkoordinatensystemen werden temporale Angaben relativ zu einem definierten Zeitpunkt definiert. Das System besteht aus zwei Attributen: dem Koordinatenursprung in Form einer Datums- und Zeitangabe und einer definierten fortlaufenden (kontinuierlichen) Intervalllänge.

Zeitordnungen (Ordinal temporal reference systems)

Die Verwendung von Zeitordnungen ist in Anwendungen wie z.B. der Geologie oder Archäologie gefragt, in denen die relative Zuordnung von Zeitpunkten weit aus präziser angegeben werden kann als die Dauer eines Zeitabschnitts. Das Bezugssystem beruht auf einer Ordinalskala. In seiner einfachsten Form ist das System eine geordnete Reihenfolge von Ereignissen.

4.2.4.3 Zeit in GI-Objekten

Die zeitliche Dimension eines GI-Objekts kann auf unterschiedliche Art und Weise berücksichtigt werden [Bart02]:

- als *Zeitattribute* in Form einer zeitlichen Merkmalsbeschreibung für ein GI-Objekt
- als *Zeitfunktion* in Form einer Merkmalsänderung eines GI-Objekts als Funktion der Zeit
- als *Zeitrelation* in Form einer Beziehung zwischen Zeitpunkten und Zeiträumen
- als *Metadatenelement* in Form einer zeitlichen Gültigkeit.

5 Oracle Spatial 3D und TimeDB

Die Entwicklung eines 4D-GIS kann nur auf Grundlage eines vollständigen 3D-GIS erfolgen. Für eine effiziente Speicherung dreidimensionaler räumlicher Daten ist es daher zweckgemäß eine 3D-Geodatenbank einzusetzen, die die Verwaltung und Anfragen dreidimensionaler Geoobjekte unterstützt. Oracle Spatial erlaubt die Speicherung der dritten Dimension. Seit Version 11 ist es zudem möglich, Körper und Sammlungen von Körpern zu modellieren. In den nächsten Abschnitten werden zunächst grundsätzliche Konzepte zur Verwaltung von Geodaten in Oracle Spatial erläutert, die zur Implementierung eines 3D-Datenmodells (siehe Kapitel 6: Realisierung eines 4D-Datenmodells) benötigt werden. Dabei wird besonders auf die 3D-Funktionalitäten von Oracle Spatial eingegangen. Relationale Datenbanksysteme stellen zurzeit noch unzureichende Mittel zur Verwaltung temporaler Daten zur Verfügung. Es kann lediglich ein bestimmter Zustand der realen Welt beschrieben werden. Für die Auswertung zeitbehafteter Geoobjekte ist daher eine temporale Abfragesprache erforderlich. Die temporale Erweiterung in Form von SQL/Temporal ist derweil noch nicht in SQL3 implementiert, da es Unstimmigkeiten im ISO Komitee darüber gab, wie weit die temporale Unterstützung im SQL3-Sprachstandard gehen soll [Snod]. Den besten Ansatz zur Beschreibung temporaler Informationen bietet der Sprachvorschlag TSQL2 auf Basis von SQL2. Die Weiterentwicklung ATSQL2 wurde in TimeDB implementiert und soll nachfolgend als temporales Datenmodell für die relationale Datenbank Oracle genutzt werden.

5.1 Oracle Spatial 3D

Oracle Spatial bildet eine Teilkomponente der Oracle 11g Enterprise Edition und stellt ein SQL-Schema und Funktionen zur Verfügung, die die Speicherung, Erfassung, Aktualisierungen und Abfragen räumlicher Objekte in einer Oracle-Datenbank erleichtern [OSDG09]. Frühere Versionen haben lediglich 2D-Geometrietypen unterstützt. Seit Version 11g werden von Oracle Spatial zusätzliche 3D-Datentypen angeboten. Dadurch ergeben sich interessante Einsatzmöglichkeiten, v. a. in der Verwaltung von 3D-Stadtmodellen und Laser-scandaten.

5.1.1 Geometrieschema – SDO_GEOMETRY

Zur Repräsentation von Geometrien stellt Oracle Spatial ein Datenbankschema zur Verfügung, in denen Klassen, Methoden und Funktionen verwaltet werden. Die Klasse SDO_GEOMETRY, die durch das Schema MDSYS bereitgestellt wird, dient zur Beschreibung der Geometrie eines Objekts. [Brink08, OSDG09]

In der Klasse MDSYS.SDO_GEOMETRY werden fünf Attribute definiert.

```
CREATE TYPE MDSYS.SDO_GEOMETRY AS OBJECT (
SDO_GTYPE          NUMBER,                        -- Geometrietyp
SDO_SRID           NUMBER,                        -- räuml. Bezugssystem
SDO_POINT          MDSYS.SDO_POINT_TYPE,          -- Punktgeometrie
SDO_ELEM_INFO      MDSYS.SDO_ELEM_INFO_ARRAY,     -- Interpretation der KO
SDO_ORDINATES      MDSYS.SDO_ORDINATE_ARRAY       -- Koordinaten );
```

Die Bedeutung dieser Attribute soll anhand der in Abbildung 5.1 dargestellten dreidimensionalen Geometrie näher erläutert werden.

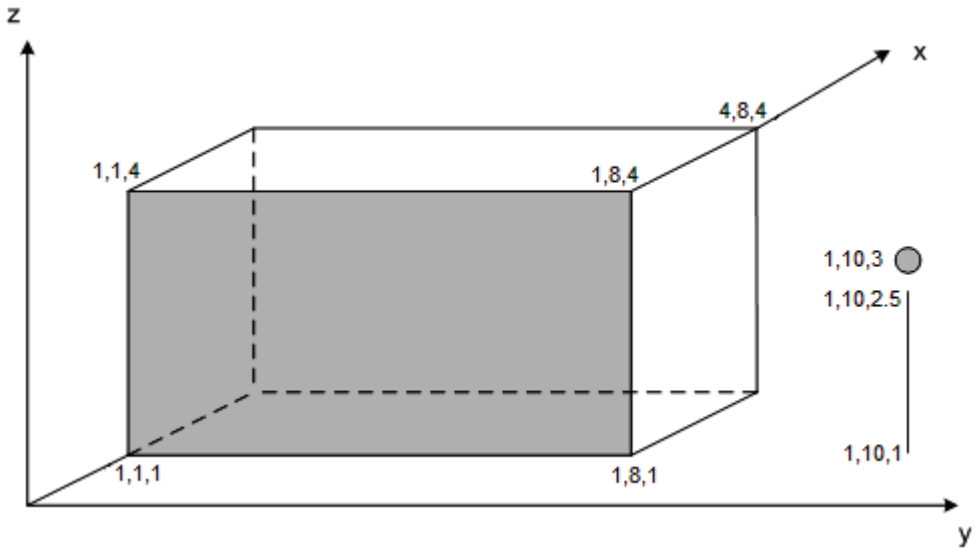

Abb. 5.1: Geometrien für die Tabelle "Object3D"

Dazu wird zunächst die Tabelle „Object3D“ angelegt.

```
CREATE TABLE Object3D (
id          INTEGER,                        -- ID
name        VARCHAR(255),                   -- Objektname
geometry    MDSYS.SDO_GEOMETRY,             -- Geometrie
CONSTRAINT pk_object PRIMARY KEY(id)        -- Primärschlüssel );
```

5.1.1.1 Geometrietyp SDO_GTYPE

Das Attribut SDO_GTYPE beschreibt den Geometrietyp durch Angabe einer vierstelligen Zahl im Format *DLTT*. *D* bestimmt dabei die Dimension der Koordinaten, die Werte zwischen 2 und 4 beinhalten kann. *L* spezifiziert das lineare Referenzsystem (LRS). Als Default-Wert wird *Null* angegeben, d.h. kein Bezugssystem festgelegt. *TT* bestimmt den Geometrietyp. [OSDG09] Dabei werden seit Version 11g neun Typen unterstützt. Tabelle 5.1 beschreibt die verfügbaren dreidimensionalen Geometrietypen. Die Werte 01 bis 07 lassen sich auch auf kleinere Dimensionen anwenden. Die Werte 08 bis 09 werden seit Version 11g explizit zur Speicherung dreidimensionaler Körper verwendet.

Tab. 5.1: Übersicht der dreidimensionalen Geometrietypen [OSDG09]

Wert	Geometrietyp	Beschreibung
3L01	POINT	3D-Punkt
3L02	LINE oder CURVE	3D-Linienzug, bestehend aus Geraden und/ oder Bögen
3L03	POLYGON oder SURFACE	3D-Fläche; ggf. mit Löchern
3L04	COLLECTION	Sammlung unterschiedlicher 3D-Geometrien
3L05	MULTIPOINT	Sammlung von 3D-Punkten
3L06	MULTILINE oder MULTICURVE	Sammlung von 3D-Linienzügen, bestehend aus Geraden oder Bögen
3L07	MULTIPOLYGON oder MULTISURFACE	Sammlung von planaren 3D-Polygonen und 3D-Flächen
3L08	SOLID	Körper
3L09	MULTISOLID	Sammlung von Körpern

5.1.1.2 Räumliches Bezugssystem SDO_SRID

Das Attribut SDO_SRID wird zur Identifizierung eines räumlichen Bezugssystems genutzt. Der Wert muss nicht gesetzt werden, wenn kein Koordinatensystem mit der Geometrie verknüpft werden soll. Unterstützt werden lokale, geographische, geozentrische kartesische und projizierte Koordinatensysteme. Die räumlichen Bezugssysteme werden im Informationsschema SDO_COORD_REF_SYS beschrieben. [OSDG09] Die Tabelle besteht aus den in Tabelle 5.2 dargestellten Attributen.

Tab. 5.2: Tabelle "SDO_COORD_REF_SYS"

Name	Typ	Beschreibung
SRID	NUMBER(38)	EPSG-Schlüssel[2]
COORD_REF_SYS_NAME	VARCHAR(80)	Name des Bezugssystems
COORD_REF_SYS_KIND	VARCHAR(24)	Kategorie des Bezugssystems
COORD_SYS_ID	NUMBER(10)	Fremdschlüssel auf SDO_COORD_SYS
GEOG_CRS_DATUM_ID	NUMBER(10)	ID des geodätischen Datums
LEGACY_CODE	NUMBER(10)	Evtl. alter Oracle-Schlüssel
LEGACY-WKTEXT	VARCHAR2(2046)	Evtl. der WKT

5.1.1.3 Attribut SDO_POINT

Das Attribut SDO_POINT dient zur effizienten Beschreibung einzelner Punkte (vom Geometrietyp 3L01). Das Attribut wird über den Datentyp SDO_POINT_TYPE definiert, der aus einem Koordinatentripel besteht.

```
CREATE TYPE MDSYS.SDO_POINT_TYPE AS OBJECT (
X       NUMBER,      -- x-Koordinate
Y       NUMBER,      -- y-Koordinate
Z       NUMBER       -- z-Koordinate              );
```

[2] Die entsprechenden EPSG-Schlüssel finden sich unter: http://epsg.org

Bei der expliziten Speicherung von Punkten ist darauf zu achten, dass die Attribute SDO_ELEM_INFO und SDO_ORDINATES nicht gesetzt sind, da die Werte in SDO_POINT sonst ignoriert werden.

Mithilfe der erlangten Erkenntnisse kann nachfolgend der in Abbildung 5.1 dargestellte 3D-Punkt mit den Koordinaten (*1,10,3*) in die Tabelle „Object3D“ eingefügt werden.

```
-- 3D-Punkt einfügen
INSERT INTO Object3D (id,name,geometry)
VALUES (1, '3D-Punkt', MDSYS.SDO_GEOMETRY(3001,NULL,
        MDSYS.SDO_POINT_TYPE(1,10,3),NULL,NULL));
COMMIT;
```

5.1.1.4 Attribut SDO_ELEM_INFO

Das Attribut SDO_ELEM_INFO beschreibt, wie die Ordinatenwerte, die in SDO_ORDINATES gespeichert werden, zu interpretieren sind. Das Attribut ist definiert als Zahlenfeld wechselnder Länge (VARRAY[3] (1048576) of NUMBER). Das Attribut besteht aus einem Triplet, das wie folgt interpretiert wird [OSDG09]:

- SDO_STARTING_OFFSET

  Gibt die Position in SDO_ORDINATES an, ab der eine Teilgeometrie beginnt (erster Feldeintrag = 1).

- SDO_ETYPE

  Gibt die Art der Teilgeometrie an. Die SDO_ETYPE-Werte 1, 2, 1003 und 2003 sind *einfache Elemente*, d.h. dass sie durch einen einzelnen Tripel-Eintrag in SDO_ELEM_INFO definiert werden. Die erste Zahl in den Werten 1003 und 2003 gibt an, ob ein Ring *außen* (1) oder *innen* (2) liegt.

  Die SDO_ETYPE-Werte 4, 1005, 2005, 1006, 2006 und 1007 werden als *zusammengesetzte Elemente* bezeichnet, da sie zumindest aus einem Haupt-

[3] Ein VARRAY ist ein geordneter Satz von Datenelementen vom selben Datentyp [KGB07]

tripel und einer Reihe mehrerer Tripel bestehen. Die SDO_ETYPE-Werte und deren Bedeutung sind in Tabelle 5.3 zusammengefasst.

Tab. 5.3: Übersicht über die Elementtypen [OSDG09]

SDO_ETYPE	Bedeutung
1	Punkt
2	Linie
1003	Äußerer Ring; Angabe der Koordinaten linksherum
2003	Innerer Ring; Angabe der Koordinaten rechtsherum
4	Zusammengesetzte Linie
1005	Zusammengesetzter äußerer Ring
2005	Zusammengesetzter innerer Ring
1006	Zusammengesetzte äußere Fläche
2006	Zusammengesetzte innere Fläche
1007	Körper

- SDO_INTERPRETATION

  Spezifiziert die Teilgeometrie näher und ist abhängig vom SDO_ETYPE-Wert. Mögliche Kombinationen für Punkte, Linien und Flächen sind in Tabelle 5.4 dargestellt.

Tab. 5.4: Werte und Bedeutung der Werte in SDO_ELEM_INFO [OSDG09]

SDO_ETYPE	SDO_INTERPRETATION	Bedeutung
1	1	Einzelner Punkt
1	$n > 1$	Sammlung von n Punkten
2	1	Geradliniger Linienzug
2	2	Linienzug aus Kreisbögen
1003 2003	1	Polygon aus geradlinigen Liniensegmenten
1003 2003	2	Polygon aus Kreisbögen
1003 2003	3	Rechteck, beschrieben durch 2 Punkte (links unten und rechts oben)

1003 2003	4	Kreis, beschrieben durch 3 unterschiedliche Punkte auf dem Umkreis
4	$n > 1$	Zusammengesetzter Linienzug, bestehend aus Linienzügen und/ oder Kreisbögen
1005 2005	$n > 1$	Zusammengesetztes Polygon
1006 2006	$n > 1$	Oberfläche, bestehend aus einem oder mehreren Polygonen
1007	$n > 1$	Körper bestehend aus mehreren Oberflächen

5.1.1.5 Attribut SDO_ORDINATES

Das Attribut SDO_ORDINATES speichert alle Koordinaten, die die Grenze eines Geoobjekts bilden. Es ist definiert als Zahlenfeld wechselnder Länge (VARRAY (1048576) of NUMBER). Das Feld muss immer mit dem Attribut SDO_ELEM_INFO angegeben werden, da die Koordinaten sonst falsch interpretiert werden. Ein Polygon aus zweidimensionalen Punkten wird in der Form $x_1, y_1, x_2, y_2, x_3, y_3, x_1, y_1$ angegeben. Da ein Polygon geschlossen ist, muss zum Abschluss immer die Anfangskoordinate angegeben werden. Eine beliebige dreidimensionale Geometrie wird in der Form $x_1, y_1, z_1, x_2, y_2, z_2$ usw. deklariert. [Brink08, OSDG09]

Mit den Erkenntnissen der vorangegangenen Abschnitte können nun die in Abbildung 5.1 dargestellten Geometrien als SDO_GEOMETRY Objekttyp gespeichert werden. Zunächst werden der dreidimensionale Linienzug und die grau hinterlegte Fläche in die Datenbank überführt.

```
-- 3D-Linienzug einfügen
INSERT INTO Object3D (id,name,geometry)
VALUES (2, '3D-Linienzug',
        MDSYS.SDO_GEOMETRY(3002,NULL, NULL,
        sdo_elem_info_array (1,2,1),
        sdo_ordinate_array (1,10,1, 1,10,2.5)));
```

```
-- Graues Polygon als Rechteck
INSERT INTO Object3D (id,name,geometry)
VALUES (3, 'Graues 3D-Polygon',
        MDSYS.SDO_GEOMETRY(3003,NULL, NULL,
        sdo_elem_info_array (1,1003,3),
        sdo_ordinate_array (1,1,1, 1,8,4)));
COMMIT;
```

Alternativ kann das graue Polygon auch über ein einfaches Polygon definiert werden. Dazu müssen alle begrenzenden Koordinaten im Gegenuhrzeigersinn angegeben werden.

```
        MDSYS.SDO_GEOMETRY(3003,NULL, NULL,
        sdo_elem_info_array (1,1003,1),
        sdo_ordinate_array (1,1,1, 1,8,1, 1,8,4, 1,1,4, 1,1,1))
```

Die Beschreibung des Quaders über sechs Polygone ist sehr aufwändig. Für solche Körper ist es daher sinnvoll den Geometrietyp 3008 (SOLID) zu verwenden und über die diagonal gegenüberliegenden Eckpunkte zu beschreiben.

```
 -- Hauskörper als Quader über diagonale Eckpunkte einfügen
INSERT INTO Object3D (id,name,geometry)
VALUES (4, 'Hauskörper',
        MDSYS.SDO_GEOMETRY(3008,NULL, NULL,
        sdo_elem_info_array (1,1007,3),
        sdo_ordinate_array (1,1,1, 4,8,4)));
COMMIT;
```

5.1.2 Räumliche Indexierung

Ein Vorteil einer Geodatenbank gegenüber anderen Datenbanksystemen besteht in der Möglichkeit, räumliche Daten effizient über räumliche Indexe abzufragen. Räumliche Indexe dienen zur Beschleunigung des Datenzugriffs auf große Mengen räumlicher Daten. Grundsätzlich werden verschiedene Indexstrukturen[4] unterschieden. Dazu zählen u. a. Quadtrees, Gridfiles und R-Bäume. Da Oracle ab Version

[4] Einen Überblick über die aufgeführten Indexstrukturen bietet [Brink08, S. 172 ff.]

11 nur noch das R-Tree-Verfahren unterstützt, soll nachfolgend detaillierter auf diese Indexstruktur eingegangen werden.

5.1.2.1 Metadaten

Bevor Geometriedaten indexiert werden können, muss der Eintrag in der Metadatensicht USER_SDO_GEOM_METADATA aktualisiert werden. Exemplarisch wird die Geometrietabelle „Object3D“ aus Abschnitt 5.1.1 in die Metadatensicht übernommen.

```
-- Update der Sicht USER_SDO_GEOM_METADATA
INSERT INTO user_sdo_geom_metadata (
        TABLE_NAME,
        COLUMN_NAME,
        DIMINFO,
        SRID)

VALUES (
        'Object3D',
        'geometry',
        SDO_DIM_ARRAY (                    -- 4x10x4 Gitternetz
        SDO_DIM_ELEMENT ('X', 0, 4, 0.005), -- kleinster Wert, größter Wert, Toleranz
        SDO_DIM_ELEMENT ('Y', 0, 10, 0.005),
        SDO_DIM_ELEMENT ('Z', 0, 4, 0.005)

),
        NULL -- SRID
);
```

5.1.2.2 R-Tree Index

Der *R-Baum* basiert auf den theoretischen Untersuchungen von Antonin Guttman (1984). Der R-Baum ist eine Rechteckstruktur, d.h. dass für den räumlichen Zugriff ein minimal umschließendes achsenparalleles Rechteck (MBR)[5] zur Beschreibung der Geometrien genutzt wird (siehe Abb. 5.2). Zunächst werden für alle Geometrieobjekte die MBR gebildet.

[5] Oracle nutzt den Begriff Minimal Bounding Rectangle (MBR) anstelle des sonst üblicherweise genutzten Begriffs Bounding Box (BB)

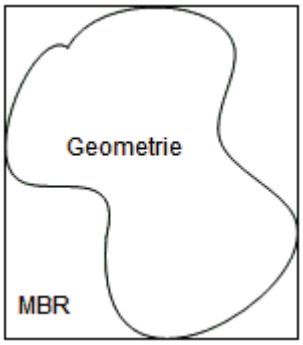

Abb. 5.2: Objektapproximation als MBR

Im Anschluss werden die approximierten Geometrien gemäß Abbildung 5.3 in einem hierarchisch aufgebauten balancierten Baum gruppiert.

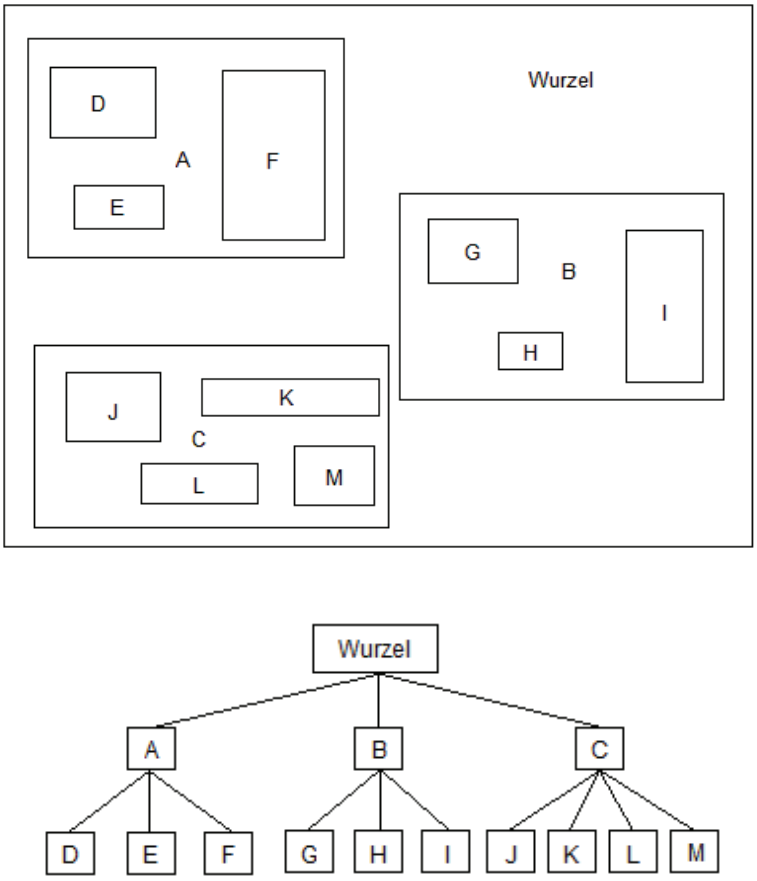

Abb. 5.3: Prinzip eines R-Baums

Ohne räumliche Indexierung müsste bei der Abfrage nach einer bestimmten Geometrie der gesamte Objektraum durchsucht werden. Durch die Bildung von Hierarchiestufen kann die Anzahl der zu prüfenden Geometrien stark eingegrenzt werden.

5.1.2.3 Indexierung in Oracle 11g

Ein räumlicher Index nach dem R-Tree-Verfahren wird in Oracle mit der folgenden SQL-Anweisung erzeugt.

```
CREATE INDEX points3d_sidx on Object3D (geometry)
INDEXTYPE IS mdsys.spatial_index
PARAMETERS ('sdo_indx_dims=3');
```

Der generierte Index bezieht sich auf die in der Tabelle “Object3D” gespeicherten Spalte geometry vom Typ SDO_GEOMETRY. Für dreidimensionale Objekte ist der Zusatz sdo_indx_dims in der Parametereinstellung von Bedeutung, um einen räumlichen Index explizit für dreidimensionale Geometrien zu erzeugen.

5.1.3 Räumliche Anfragebearbeitungen

Räumliche Anfragen werden in Oracle nach dem Prinzip der mehrstufigen Anfragebearbeitung durch den Einsatz von Filtern verarbeitet (siehe Abb. 5.4).

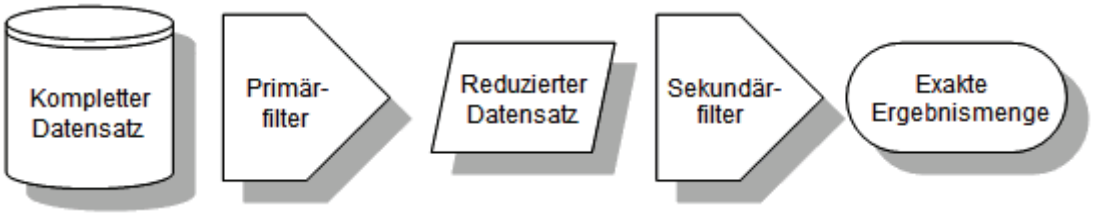

Abb. 5.4: Zweistufige Anfragebearbeitung in Oracle Spatial (vgl. [OSDG09])

Ziel einer solchen Anfragebearbeitung ist es, eine Untermenge des gesuchten Datensatzes bereits mithilfe des Primärfilters zu extrahieren. Dazu werden schnell ausführbare Algorithmen verwendet. Die exakte Ergebnismenge kann infolge aufwändigerer Berechnungen auf den reduzierten Datensatz mithilfe des Sekundärfilters zurückgegeben werden. [OSDG09]

Zu den wichtigsten räumlichen Abfragen mit 3D-Funktionalität in Oracle 11g zählen SDO_FILTER, SDO_RELATE, SDO_WITHIN_DISTANCE und SDO_NN. Die räumlichen Operatoren sollen im Nachfolgenden näher erläutert werden.

5.1.3.1 SDO_FILTER

Der Operator SDO_FILTER gibt alle Geometrien als Ergebnismenge zurück, die eine Überlappung mit ein oder mehrerer Anfragegeometrien aufweist. Der Operator weist die folgende Form auf:

```
SDO_FILTER (
        geometry1     IN     SDO_GEOMETRY,     -- Geometrieattribut
        geometry2     IN     SDO_GEOMETRY,     -- Anfragegeometrie
        params        IN     VARCHAR2          -- Parameter
)
```

Der Parameter geometry1 spezifiziert das geometrische Attribut in einer Tabelle. Die vorliegende Geometrie vom Typ SDO_GEOMETRY muss räumlich indexiert sein. Der Parameter geometry2 beschreibt die Anfragegeometrie. Dabei kann die Geometrie aus einer anderen Tabelle stammen oder in einer Variablen gespeichert sein. Ein räumlicher Index muss nicht vorliegen. Der Nachteil dieser Methode besteht darin, dass für die Filterung der Geometrien lediglich der Primärfilter genutzt wird. Da der Primärfilter lediglich die Objektapproximationen, also das eine Geometrie minimal umschließende Rechteck, nutzt, ist es möglich, dass Geometrien in die Resultatmenge gelangen, deren exakte Geometrie nicht innerhalb des Abfragefensters liegt. [Brink08, OSDG09]

5.1.3.2 SDO_RELATE

Im Gegensatz zu SDO_FILTER nutzt der Operator SDO_RELATE sowohl den Primär- als auch den Sekundärfilter, sodass eine exakte topologische Schnittmenge als Resultat zurückgegeben werden kann [OSDG09]. Tabelle 5.5 beschreibt alle in Oracle Spatial enthaltenen topologischen Beziehungen, die auf Grundlage des 9-Intersection-Modells nach EGENHOFER definiert sind.

Tab. 5.5: Topologische Beziehungen in Oracle Spatial

Oracle Spatial	Bedeutung
DISJOINT	9IM: Disjoint
CONTAINS	9IM: Contains
COVERS	9IM: Covers
EQUAL	9IM: Equal
TOUCH	9IM: Meets
INSIDE	9IM: Inside
COVEREDBY	9IM: Covered By
OVERLAPBDYINTERSECT	9IM: Overlaps
ON	Prüft, ob ein Linienzug sich vollständig auf dem Rand eines Polygons befindet
OVERLAPDBYDISJOINT	Prüft, ob ein Linienzug in einem Polygon beginnt, aber außerhalb endet
ANYINTERACT	Gibt topologische Schnittmenge zurück

Der Operator hat die folgende Syntax:

```
SDO_RELATE (
        geometry1     IN     SDO_GEOMETRY,     -- Geometrieattribut
        geometry2     IN     SDO_GEOMETRY,     -- Anfragegeometrie
        params        IN     VARCHAR2          -- Parameter
)
```

Die Form ist ähnlich zu dem im Abschnitt 5.1.3.1 vorgestellten Operator SDO_FILTER. Der Unterschied besteht lediglich darin, dass die topologischen Untersuchungen über das Schlüsselwort mask in den Parametereinstellungen eingeleitet werden. Im Release 1 von Oracle 11g können topologische Beziehungen bisher nur auf zweidimensional indexierte Geometrien angewendet werden. Für dreidimensionale Geometrien können aber topologische Schnittbeziehungen mithilfe des Operators SDO_ANYINTERACT geprüft werden.

5.1.3.3 SDO_WITHIN_DISTANCE

Der räumliche Operator SDO_WITHIN_DISTANCE gibt als Resultatmenge alle Geometrien zurück, die innerhalb eines definierten Umkreises der Abfragegeometrie liegen. Der Operator wird wie folgt dargestellt:

```
SDO_WITHIN_DISTANCE (
        geometry1    IN    SDO_GEOMETRY    -- Geometrieattribut einer Tabelle
        ageom        IN    SDO_GEOMETRY    -- Anfragegeometrie
        params       IN    VARCHAR2        -- Parameter
)
```

Im Parameter geometry1 wird das geometrische Attribut einer Tabelle beschrieben. Im Parameter ageom wird die Anfragegeometrie spezifiziert. Der Parameter params enthält weitere Einstellungen zur Anfrage (z.B. die Distanz des Puffers um die Anfragegeometrie). [Brink08, OSDG09]

5.1.4 Import räumlicher Daten

Bei Geodaten handelt es sich zumeist um sehr große Datenmengen. Die einfachste Möglichkeit Geometrien in die Datenbank zu überführen, ist die Nutzung der Standard SQL Syntax, auch als *Transactional Insert* bezeichnet. Diese Methode weist allerdings einige Nachteile auf. Zum einen ist die Formulierung einer entsprechenden SQL-Anweisung sehr aufwändig und die Gefahr einer fehlerbehafteten Syntax sehr hoch. Zum anderen hat das INSERT-Statement von Oracle eine Beschränkung von 999 Argumenten [OSDG09], sodass sich diese Methode eher für kleinere Datensätze eignet.

Um ein räumliches Datenmodell effektiv mit großen Datenmengen zu füllen, wird daher die so genannte *Bulk Loading*[6]-Methode von Oracle zur Verfügung gestellt. Zum Import wird das Werkzeug SQL*Loader benötigt, das den Import größerer Datenmengen aus ASCII-Dateien (z.B.: *.txt oder *.xyz) unterstützt. Die Spalten werden durch ein beliebiges Trennzeichen voneinander getrennt. Nachkommastellen

[6] Bulk Loading: „Massenladen“

von Zahlenwerten werden, anders als in SQL Statements, durch ein Komma getrennt. SQL*Loader benötigt für den Import eine Kontrolldatei (*.ctl), in der Name und Ort der Importdatei, die Zieltabelle inklusive der Attribute in Oracle und das Trennzeichen zwischen den Spalten definiert ist.

5.1.4.1 Import von Punktdaten

Im Nachfolgenden sollen exemplarisch einige Punktdatensätze mithilfe der Bulk Loading-Methode in Oracle importiert werden. Dazu muss zunächst die Ladekontrolldatei definiert werden.

```
LOAD DATA INFILE 'E:\Daten\3D_punkte.xyz'        -- Importdatei
TRUNCATE INTO TABLE Punkte3D                     -- Zieltabelle
FIELDS TERMINATED BY ';'                         -- Trennzeichen: Semikolon
TRAILING NULLCOLS        (                       -- letztes Attribut der Kontrolldatei
                                                 wird auf NULL gesetzt
    id,
    geometry COLUMN OBJECT     (
        sdo_gtype,
        sdo_srid,
        sdo_point COLUMN OBJECT (x, y, z)
                                )
            )
```

Eine Importdatei, die zu der Kontrolldatei passt, könnte wie folgt aussehen:

```
1;3001;4326;1;1;1
2;3001;4326;1;8;1
3;3001;4326;4;8;1
4;3001;4326;1;4;1
5;3001;4326;1;1;4
6;3001;4326;1;8;4
7;3001;4326;4;8;4
8;3001;4326;1;4;4
```

Die importierten Punkte entsprechen allen dargestellten Punkten des in Abbildung 5.1 dargestellten Quaders mit Bezug auf das WGS84-Referenzsystem (SRID: 4326).

Der SQL*Loader wird über das Programm SQLLDR im Kommandozeilenfenster des Betriebssystems gestartet.

```
sqlldr userid=<Benutzername>/<Passwort>@<Dienstname>
control=E:\Daten\3d_punkte.ctl
```

5.1.4.2 Import von Flächen und Körpern

Der Import von Flächen und Körpern soll wiederum anhand des Quaders aus Abbildung 5.1 erläutert werden. Anders als beim Import von Punkten müssen nun zusätzlich die Attribute SDO_ELEM_INFO und SDO_ORDINATES in der Kontrolldatei gesetzt werden.

```
LOAD DATA INFILE 'E:\Daten\3D_polygone.xyz'
TRUNCATE
CONTINUEIF NEXT (1:1) = '#'                          -- Fortführungszeichen
INTO TABLE Polygone3D
FIELDS TERMINATED BY ';'
TRAILING NULLCOLS (
        id,
        name,
        geometry COLUMN OBJECT (
                sdo_gtype,
                sdo_srid,
                sdo_elem_info VARRAY TERMINATED BY '|' (elements),
                sdo_ordinates VARRAY TERMINATED BY 'end'  (ordinates) ))
```

Eine entsprechende Importdatei könnte wie folgt aussehen:

```
1;Hausfront;3003; 4326; 1;1003;1|1;1;1; 1;8;1;
#1;8;4; 1;1;4; 1;1;1 end
2;Dach;3003; 4326; 1;1003;1| 1;1;4; 1;8;4;
#4;8;4; 4;1;4; 1;1;4 end
3;Hauskörper;3008; 4326; 1;1007;3|1;1;1; 4;8;4 end
```

Die Datei enthält einfache dreidimensionale Polygone und einen Körper, der über die Diagonalen des Quaders bestimmt wurde.

5.2 TimeDB

TimeDB unterstützt eine temporale Version von SQL, die als ATSQL2 bezeichnet wird. TimeDB ermöglicht durch die Übersetzung von temporalen SQL Statements in Standard SQL Statements die Erweiterung von nicht-temporalen relationalen DBMS um temporale Funktionalitäten. Das in Java geschriebene Programm hat den Vorteil, dass es lediglich als Frontend auf ein bereits bestehendes Datenbanksystem aufsetzt, sodass aufwendige Modifikationen der Datenbank entfallen.

5.2.1 Merkmale von TimeDB

TimeDB ist ein bitemporales DBMS, d.h. dass sowohl Gültigkeits- als auch Transaktionszeiten unterstützt werden. Es basiert auf der temporalen Sprache ATSQL2 als Weiterentwicklung von TSQL2 (vgl. Abschnitt 4.2.3: TSQL2 Datenmodell), die neben temporalen Abfragen auch Datenmanipulationen, Datendefinitionen sowie die Formulierung von temporalen Integritätsbedingungen zulässt. [Ste98]

5.2.2 Architektur des TimeDB DBMS

Abbildung 5.5 beschreibt die Architektur des temporalen DBMS TimeDB. Die Module Parser, Translator und Evaluator rufen Operationen, die in den Modulen Scanner, Checker und Coalescing implementiert sind, auf. Der Scanner liest ein ATSQL2 Statement ein, generiert eine Zeichenkette und übergibt diese an den Parser, der das Statement auf Gültigkeit überprüft. Bei Syntaxfehlern wird die Übersetzung des Statements abgebrochen. Ist das ATSQL2 Statement syntaktisch korrekt, wird ein sog. Parse Tree generiert und an den Translator übergeben. Der Translator prüft zunächst, ob alle im Statement angegebenen Tabellen und Attribute existieren. Im Falle einer temporalen Anfrage wird überprüft, ob die Tabellen vom richtigen Typ (validtime, transactiontime, bitemporal) sind. [Ste98]

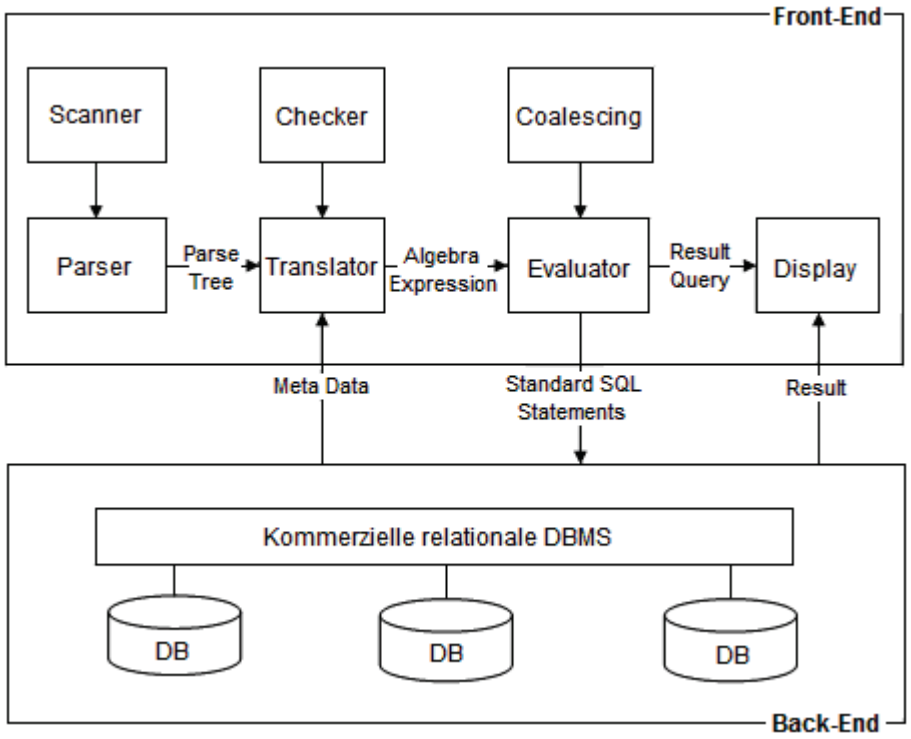

Abb. 5.5: Architektur von TimeDB [Ste98]

Der Checker stellt für diese Anfragen die benötigten Operationen zur Verfügung. Bei syntaktisch korrekten Anfragen wird das Statement in einen algebraischen Ausdruck übersetzt und an den Evaluator gesendet, der den Ausdruck auf dem kommerziellen relationalen DBMS ausführt. Die Anfrageergebnisse werden nachfolgend im Display angezeigt. [Ste98]

5.2.3 Temporale Ausdrücke und Vergleichsoperatoren

In TimeDB werden sowohl absolute als auch relative temporale Primitive unterstützt (siehe Tabelle 5.6).

Tab. 5.6: Temporale Primitive in TimeDB und deren Bedeutung

Klassen	Temporale Primitive	Bedeutung
Relativ	span	Zeitspanne, z.B. 30 Tage
Absolut	event (instant) time interval (period)	Zeitpunkt, z.B. 06/09/2009 Zeitraum, z.B. 06/09/2009 – 25/09/2009

Darüber hinaus werden für die temporalen Primitive Arithmetische und Vergleichsoperatoren angeboten, die in Tabelle 5.7 zusammengefasst sind.

Tab. 5.7: Temporale Primitive und zulässige Operationen in TimeDB

Temporale Primitive	**Zulässige Arithmetische Operatoren**	**Zulässige Vergleichsoperatoren**
span	span + span = span span – span = span span * number = span span / number = span	= < , > <= , >= <>
event	event + span = event event – span = event	precedes equals
time interval		precedes overlaps meets contains equals

5.2.4 Kalender

In TimeDB wird ein einfacher minimaler Kalender unterstützt. Der Kalender beginnt mit Jahr 1 und endet in der Zukunft mit dem Jahr 9999 (forever Operator). Jeder Monat hat 30 Tage und jeder Tag 24 Stunden (von 0 bis 23). Die kleinste darstellbare Zeiteinheit ist die Sekunde.

6 Realisierung eines 4D-Datenmodells

Die Modellierung der Phänomene der realen Welt ist eine der zentralsten und schwierigsten Aufgaben bei der Entwicklung eines GIS. Modelle werden benötigt, um natürliche oder künstliche Objekte durch eine Reduzierung der Komplexität und Abstrahierung der Ausgangsmenge zu repräsentieren. Dieses Kapitel baut auf den Erkenntnissen der vorangegangen Abschnitte auf. Zunächst werden zwei 3D-Modelle vorgestellt und die jeweiligen Vor- und Nachteile der Modelle herausgearbeitet. Anschließend wird das temporale Modell beschrieben und mit dem topologischen Modell verknüpft. Somit handelt es sich beim entwickelten 4D-Modell um ein temporales dreidimensionales topologisches Datenmodell, das auf Basis der beiden Datenbanktechnologien Oracle und TimeDB implementiert wurde.

6.1 Dreidimensionales geometrisches Datenmodell

Abbildung 6.1 beschreibt das dreidimensionale geometrische Datenmodell in einem ER-Modell.

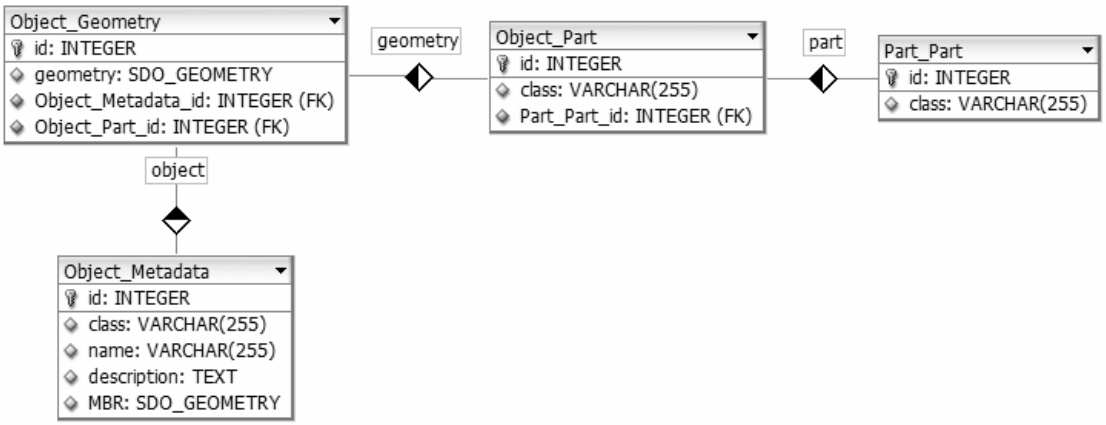

Abb. 6.1: Geometrisches 3D-Datenmodell in ER-Notation

Das Modell besteht aus vier Relationen. Drei dieser Relationen dienen lediglich zur thematischen Beschreibung eines 3D-Geoobjekts. In der Tabelle „Object_Metadata“ werden Metadaten wie Art, Name oder eine Beschreibung des 3D-Objekts verwaltet (siehe Tabelle 6.1). Optional kann ein MBR gespeichert werden,

um beispielsweise ein gesamtes Objekt für eine effiziente räumliche Suche abzufragen.

Tab. 6.1: Tabelle 'Object_Metadata' im geometrischen Modell

Attribut	Bedeutung
id (PK)	Eindeutiger Objekt-Identifikationsschlüssel
class	Objektart, z.B. Haus
name	Objektname (optional)
description	Objektbeschreibung (optional)
MBR	Minimal umschließendes Rechteck als SDO_GEOMETRY (optional)

In der Relation „Object_Part“ (siehe Tab. 6.2) werden unterschiedliche Objekteigenschaften verwaltet. Ein Objekt der Klasse Gebäude kann beispielsweise aus den Objektteilen Dach, Fassade oder Mauer bestehen. Eine Spezialisierung der in dieser Relation gespeicherten Eigenschaften bietet die Tabelle „Part_Part“ (siehe Tab. 6.3), in der die Attribute um spezielle Eigenschaften erweitert werden. Es ist zum Beispiel möglich, eine spezielle Dachform (Spitzdach etc.) zu spezifizieren. Die beiden Tabellen sind über einen Fremdschlüssel miteinander verknüpft.

Tab. 6.2: Tabelle 'Object_Part' im geometrischen Modell

Attribut	Bedeutung
id (PK)	Eindeutiger Objektteil-Identifikationsschlüssel
class	Objektteile: Dach, Fassade, Mauer etc.
part_part_id (FK)	Fremdschlüssel aus ‚Part_Part’

Tab. 6.3: Tabelle 'Part_Part' im geometrischen Modell

Attribut	Bedeutung
id (PK)	Eindeutiger Objektteil-Identifikationsschlüssel
class	Spezialisierung der in ‚Object_Part’ gehaltenen Objektteile z.B. für Dach: Spitzdach oder Flachdach

Der Kern dieses Datenmodells liegt in der Relation „Object_Geometry“ (siehe Tab. 6.4). In dieser Tabelle werden alle 3D-Geometrien der 3D-Objekte gespeichert und räumlich indexiert. Des Weiteren werden über die Fremdschlüssel part_id und metadata_id die Verbindungen zu den thematischen Eigenschaften der Objekte aufgebaut.

Tab. 6.4: Tabelle ‚Object_Geometry’ im geometrischen Modell

Attribut	**Bedeutung**
id (PK)	Eindeutiger Geometrie-Identifikationsschlüssel
part_id (FK)	Fremdschlüssel aus ‚Object_Part’
metadata_id (FK)	Fremdschlüssel aus ‚Object_Metadata’
geometry	Indexierte 3D-Geometrie als SDO_GEOMETRY

Ferner ist es möglich, weitere geometrische Datenmodelle speziell für unregelmäßige Oberflächen (TIN) und 3D-Punktwolken in Oracle 11g zu implementieren. Mögliche Datenmodelle werden in [Brink08, S. 427-434] beschrieben.

Vor- und Nachteile des geometrischen Datenmodells

Die Vorteile des geometrischen Lösungsansatzes liegen v. a. in der Simplizität des Datenmodells. Die Verwaltung der geometrischen Eigenschaften von 3D-Geoobjekten erfolgt in einer einzigen Tabelle. Komplexe Geometrien können dadurch sehr effizient in einer einzigen Spalte gespeichert werden. Des Weiteren können die 3D-Funktionalitäten von Oracle Spatial ausgenutzt werden. Ferner bietet Oracle erste Ansätze topologischer Funktionen. Im zweidimensionalen Raum wird das 9-Intersection-Modell nach EGENHOFER unterstützt. Im dreidimensionalen Raum lassen sich zumindest die nächstgelegenen Nachbarn (Nearest Neighbor) ermitteln und topologische Schnittbeziehungen prüfen (vgl. Abschnitt 5.1.3: Räumliche Anfragebearbeitungen). Ein weiterer Vorteil ist die sehr schnelle Visualisierung der Geometrien. Die Einfachheit dieses Ansatzes führt allerdings auch den größten Nachteil mit sich. Die Koordinaten werden redundant gehalten, sodass in komplexen Modellen jeder Punkt circa sechsmal gespeichert werden muss [Coors04]. Die Veränderung eines Koordinatentripels (x_0, y_0, z_0) in einer Geometrie hat die Modifi-

kation aller anderen Geometrien mit diesem Koordinatentripel (x_0, y_0, z_0) zur Folge. Der Aufwand, der durch die Änderung aller Geometrien entsteht, ist dadurch sehr hoch. Des Weiteren hat die redundante Speicherung der Koordinaten ein hohes Datenvolumen und infolge dessen einen hohen Speicherbedarf zur Folge. Das Modell ist daher eher für kleinere Datensätze mit wenigen Änderungen geeignet. Die aufgeführten Probleme führten zur Entwicklung eines topologischen Ansatzes.

6.2 Dreidimensionales topologisches Datenmodell

Die Entwicklung des dreidimensionalen topologischen Datenmodells orientierte sich stark an dem aus Abschnitt 4.1.3.3 vorgestellten Datenmodell UDM nach COORS. Das Datenmodell wurde in ein relationales Datenbankmodell überführt (siehe Abb. 6.2).

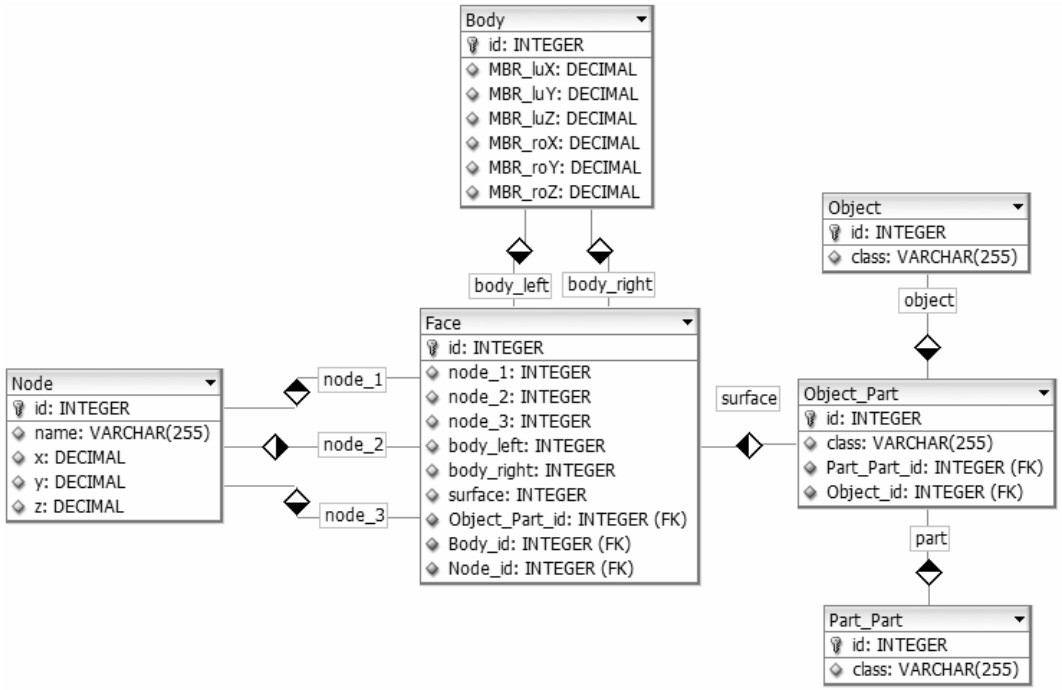

Abb. 6.2: Topologisches 3D-Datenmodell in ER-Notation (vgl. [Coors05])

Im konzeptionellen Modell besteht zwischen einer Fläche (Face) und den zugrunde liegenden Knoten (Node) eine n:m-Beziehung, d.h. dass eine Fläche als konvexes Polygon aus mehreren Knoten bestehen und ein Knoten auf der anderen Seite zu mehreren Flächen gehören kann. Zur vereinfachten Speicherung der Daten in ei-

nem relationalen Datenbankmodell wurde daher die zusätzliche Einschränkung eingeführt, dass eine Fläche nur ein planares Dreieck sein darf.

Dem topologischen Modell liegen sechs Relationen zugrunde. In der Tabelle „Node“ werden die Knoten als Koordinatentripel der Form *x*, *y*, *z* gespeichert. In der Relation „Face“ werden die eine Fläche begrenzenden Knoten node_1, node_2 und node_3 angegeben. Die Orientierung der Fläche erfolgt über die Reihenfolge der Knoten. Die Knoten müssen entgegen dem Uhrzeigersinn angegeben werden, damit die Flächennormale nach außen zeigt. Des Weiteren wird zwischen linkem und rechtem Körper (Body) unterschieden, da eine Fläche ein oder zwei Körper begrenzen kann. Falls eine Fläche lediglich einen Körper begrenzt, muss die linke Seite der Fläche, die die Richtung der Flächennormalen angibt, auf den leeren Raum (-1) verweisen. Des Weiteren wird in der Tabelle „Body“ ein minimal umgebendes Rechteck (MBR) für eine effiziente räumliche Suche gespeichert. [Coors04, Coors05]

Zur Speicherung der thematischen Eigenschaften eines 3D-Geoobjekts werden die drei Relationen „Object“ (Tab. 6.5), „Part_Part“ (Tab. 6.6) und „Object_Part“ (Tab. 6.7) herangezogen. Die Tabelle „Object_Part“ stellt dabei den Kern des thematischen Modells dar, in der die Objektteile eines Objekts sowie die Objektart und die speziellen Objektteile über die Fremdschlüssel part_part_id und object_id gespeichert sind.

Tab. 6.5: Tabelle 'Object’ im topologischen Modell

Attribut	Bedeutung
id (PK)	Eindeutiger Objekt-Identifikationsschlüssel
class	Objektart, z.B. Haus

Tab. 6.6: Tabelle 'Part_Part' im topologischen Modell

Attribut	Bedeutung
id (PK)	Eindeutiger Objektteil-Identifikationsschlüssel
class	Spezialisierung der in ‚Object_Part’ gehaltenen Objektteile z.B. für Dach: Spitzdach oder Flachdach

Tab. 6.7: Tabelle 'Object_Part' im topologischen Modell

Attribut	Bedeutung
id (PK)	Eindeutiger Objektteil-Identifikationsschlüssel
class	Objektteile: Dach, Fassade, Mauer etc.
part_part_id (FK)	Fremdschlüssel aus ‚Part_Part'
object_id (FK)	Fremdschlüssel aus ‚Object'

Vor- und Nachteile des topologischen Datenmodells

Größter Vorteil des topologischen Datenmodells gegenüber dem vorgestellten geometrischen Modell ist die redundanzfreie Speicherung der Punktkoordinaten. Jedes Koordinatentripel muss lediglich einmal gespeichert werden. Daraus ergibt sich eine einfachere Erhaltung der Datenkonsistenz bei einer Modifikation des Datenbestands. Zudem weist das topologische Modell eine sehr hohe Effizienz bei topologischen Anfragen auf. Ein weiterer Vorteil des hier vorgestellten topologischen Modells gegenüber anderen topologischen Modellen (z.B. 3D-FDS) besteht in der Reduzierung des Datenvolumens durch die implizite Speicherung von Kanten.

Der Nachteil des UDM als relationales Modell ist, dass lediglich planare Dreiecke als Flächen gespeichert werden können und komplexe Geometrien somit auf ein Minimum reduziert werden müssen. Ein Nachteil gegenüber dem geometrischen Modell ist die Speicherung der topologischen Primitive Node und Face in separaten Relationen und die damit verbundene Verknüpfung über Fremdschlüssel. Das Datenmodell weist gegenüber dem geometrischen Datenmodell dadurch eine höhere Komplexität auf.

6.3 Temporales Datenmodell

6.3.1 Temporales Schema

Die Entwicklung des temporalen Datenmodells orientierte sich zum einen an dem in Abschnitt 4.2.3 vorgestellten TSQL2 Datenmodell und zum anderen an den Möglichkeiten, die der Sprachvorschlag ATSQL2 in Form von TimeDB bot. Das Zeitmo-

dell unterscheidet zwischen Zeitinformation, Zeitverlauf, Historie sowie temporaler Repräsentation.

6.3.2 Temporale Struktur

Bei der Repräsentation der temporalen Struktur wird zwischen drei temporalen Primitiven unterschieden (siehe Abb. 6.3), die entweder *absolut* (*anchored*) oder *relativ* (*unanchored*) dargestellt werden können. Die temporale Domäne wird als diskret angenommen. Des Weiteren wird zwischen *determinierten* (*deterministic*) und *nicht-determinierten* (*indeterministic*) temporalen Primitiven unterscheiden, d.h. dass die temporale Struktur entweder exakt oder ungenau beschrieben werden kann.

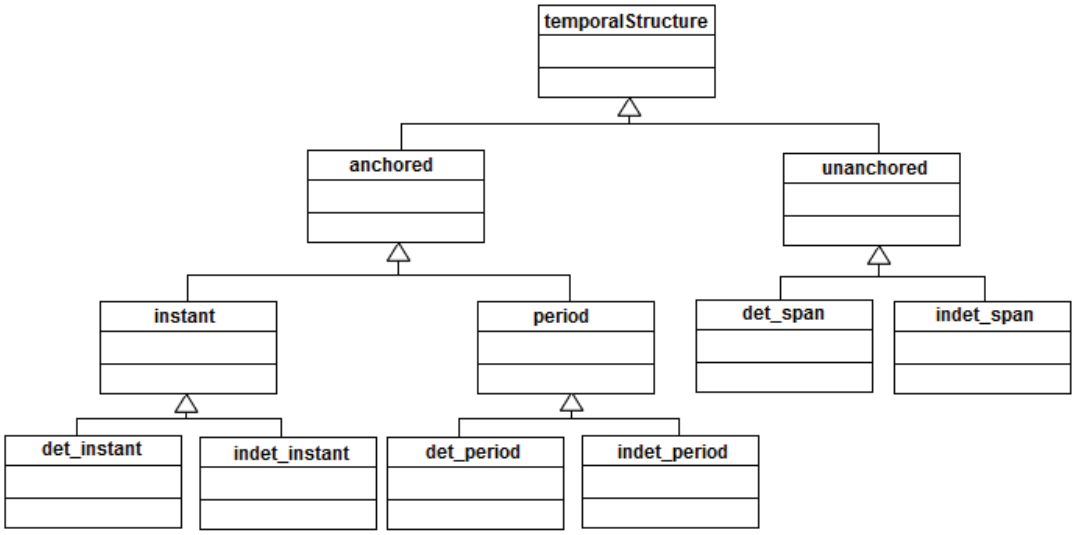

Abb. 6.3: Temporale Strukturen in UML-Notation (vgl. [ZK01])

Absolute temporale Primitive können in Form eines Zeitpunktes *i* (instant) und eines Zeitraums *p* (period) mit $i_A < i_E$ repräsentiert werden. Das temporale Primitiv span beschreibt eine relative Zeitinformation durch Angabe einer Zeitspanne.

Für die Auswertung zeitabhängiger Primitive lassen sich die in Abschnitt 5.2.3 vorgestellten Arithmetischen und Vergleichsoperatoren von TimeDB nutzen.

Die temporalen Primitive lassen sich zu einer temporalen Ordnung zusammenfassen, in der die Zeit linear oder sublinear verlaufen kann. In der sublinearen Klasse können sich die Zeitintervalle überlappen, sodass diese nur zur Beschreibung von nicht-determinierten zeitlichen Zusammenhängen herangezogen wird. In der linearen Ordnung ist eine Überlappung nicht möglich (siehe Abb. 6.4).

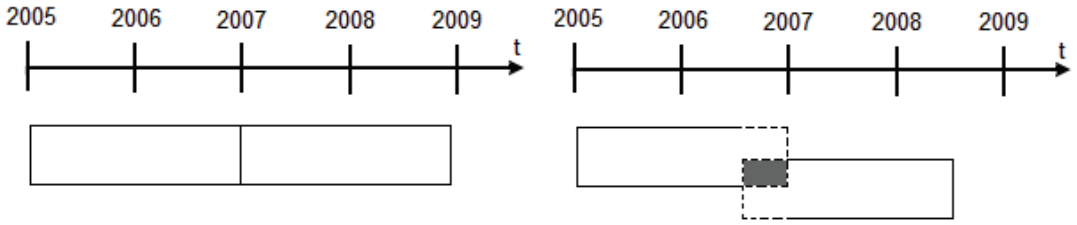

Abb. 6.4: Beispiele für eine lineare (links) und sublineare (rechts) Ordnung

6.3.3 Temporale Bezugssysteme

Die Zeit kann als Dimension nur relativ zu einem vordefinierten Bezugssystem gemessen werden (vgl. Abschnitt: 4.2.4.2: Temporale Bezugssysteme). In TSQL2 werden grundsätzlich verschiedene Kalender (z.B. Gregorianischer Kalender oder Mondkalender) unterstützt. Für die meisten Anwendungsgebiete reicht es allerdings aus, den in TimeDB implementierten minimalen Kalender mit einer zeitlichen Genauigkeit von einer Sekunde zu nutzen.

6.3.4 Temporale Historie

Die temporale Historie beschreibt die Entwicklung eines Objekts der realen Welt über die Zeit. Hierfür ist es erforderlich sowohl die Gültigkeitszeit als auch die Transaktionszeit in der Datenbank festzuhalten, um zum einen das physikalische Entstehen eines Geoobjekts und dessen Lebensdauer und zum anderen das Erfassungsdatum in der Datenbank beschreiben zu können. Das vorgestellte temporale Modell unterstützt beide zeitlichen Dimensionen.

6.4 Zusammenführen der Modelle

Abbildung 6.5 beschreibt das Zusammenspiel der räumlichen, thematischen sowie temporalen Eigenschaften eines Geoobjekts. Dazu wurde das in Abschnitt vorgestellte topologische dreidimensionale Datenmodell mit dem temporalen Datenmodell gekoppelt. Zur Beschreibung der Gültigkeits- und Transaktionszeit wurden die bitemporalen Komponenten jeweils mit den Relationen zur Beschreibung der räumlichen („Node“) und thematischen („Object_Part“) Informationen in Beziehung gesetzt. Die in Abbildung 6.5 dargestellten Relationen „Valid“ und „Transaction“ dienen dabei aber nur zur Orientierung. Intern werden sowohl Gültigkeits- als auch Transaktionszeit als Attribute der Tabellen „Node“ und „Object_Part“ gespeichert. Die Verbindungen zu den jeweiligen Tabellen werden über die Schlüsselwörter validtime und transactiontime hergestellt. Das nachfolgende Beispiel verdeutlicht, wie historische Tabellen angelegt, entsprechende Datensätze eingefügt und Daten hinsichtlich temporaler Kriterien abgefragt werden.

Beispiel

```
CREATE TABLE Node
(id INTEGER, name VARCHAR(255), x DECIMAL, y DECIMAL, z DECIMAL)
AS VALIDTIME;

validtime period [date '17.09.2009'- forever)
insert into Node values (1, '1', 0.0,10.0,5.0);

validtime period [date '17.09.2009'- date '19.09.2010')
insert into Node values (2, '2', 10.0, 0.0, 5.0);

validtime select * from Node where name = '1';
```

id	name	x	y	z	validtime
1	1	0.0	10.0	5.0	[17.09.2009-forever)

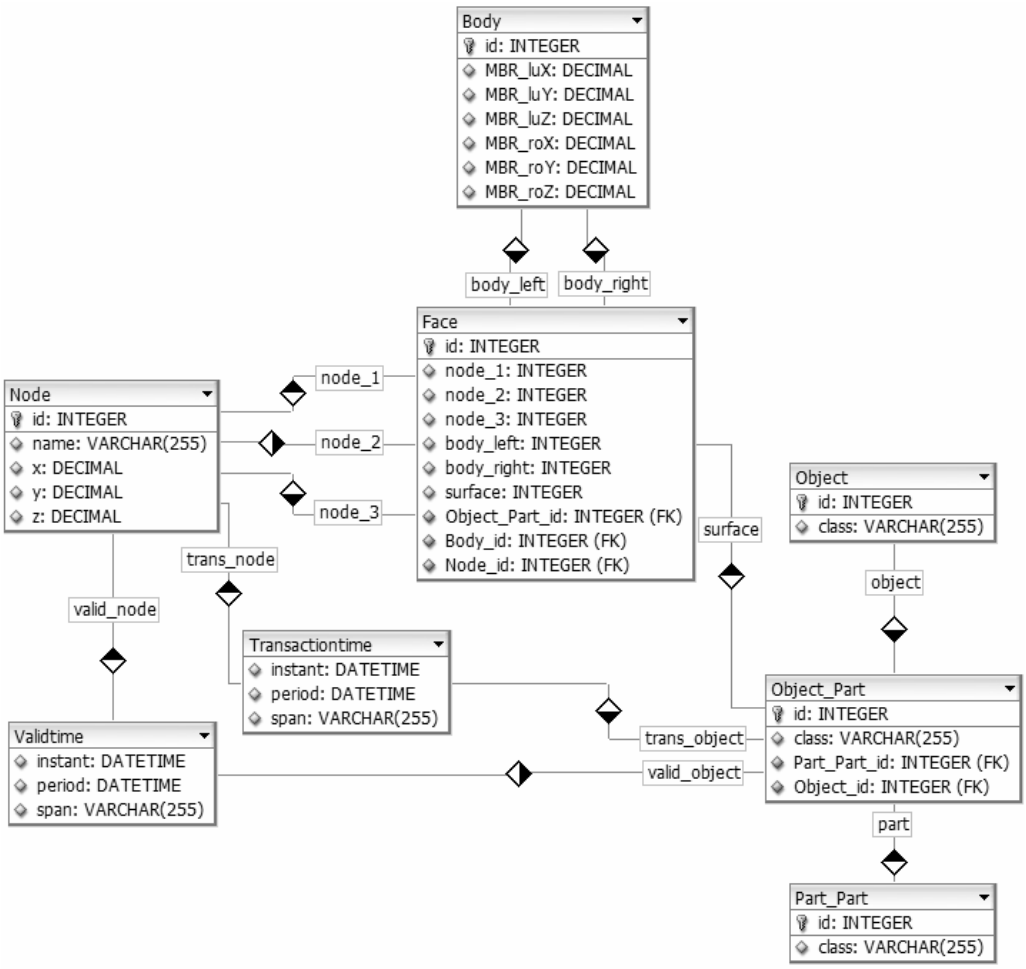

Abb. 6.5: Topologisches 4D-Datenmodell in ER-Notation

Bei der Beschreibung der zeitlichen Gültigkeit eines Geoobjekts können folgende Zustände eintreten:

- Räumliche Veränderung des Geoobjekts im Zeitverlauf
- Thematische Veränderung des Geoobjekts im Zeitverlauf
- Räumliche und thematische Veränderung des Geoobjekts im Zeitverlauf

6.4.1 Raum-zeitliche Variabilität eines Geoobjekts

Ein Geoobjekt kann seine geometrischen und topologischen Eigenschaften im temporalen Verlauf verändern. Dies wird auch als raum-zeitliche Variabilität bezeichnet. Im vorgestellten topologischen Modell ist ein Knoten (*Node*) der Träger der räumlichen Information. Über die Knoten werden die entsprechenden Flächen und Körper

zusammengesetzt. Aufgrund dessen müssen lediglich die Knoten mit der temporalen Information versehen werden. Dabei werden folgende Zustände unterschieden:

1. Vorhandene Knoten bleiben unverändert
2. Vorhandene Knoten verändern ihre Position
3. Hinzufügen neuer Knoten
4. Vorhandene Knoten verlieren ihre Gültigkeit

Im ersten Fall muss lediglich die zeitliche Gültigkeit des Geoobjekts aktualisiert werden. Wenn bereits vorhandene Knoten ihre Position verändern (2), so verändert sich auch die geometrische Form des Geoobjekts. Das Objekt setzt sich somit aus den bereits vorhandenen unveränderten und den veränderten Knoten zusammen. Beim Hinzufügen und Entfernen von Knoten (3 und 4) verändern sich die topologischen Zusammenhänge im jeweiligen Objekt. Diese Zustandsänderungen haben zur Folge, dass jeweils neue Geoobjekte gebildet werden müssen. Eine Kombination aus geometrischen und topologischen Veränderungen ist ebenso möglich (siehe Abb. 6.6).

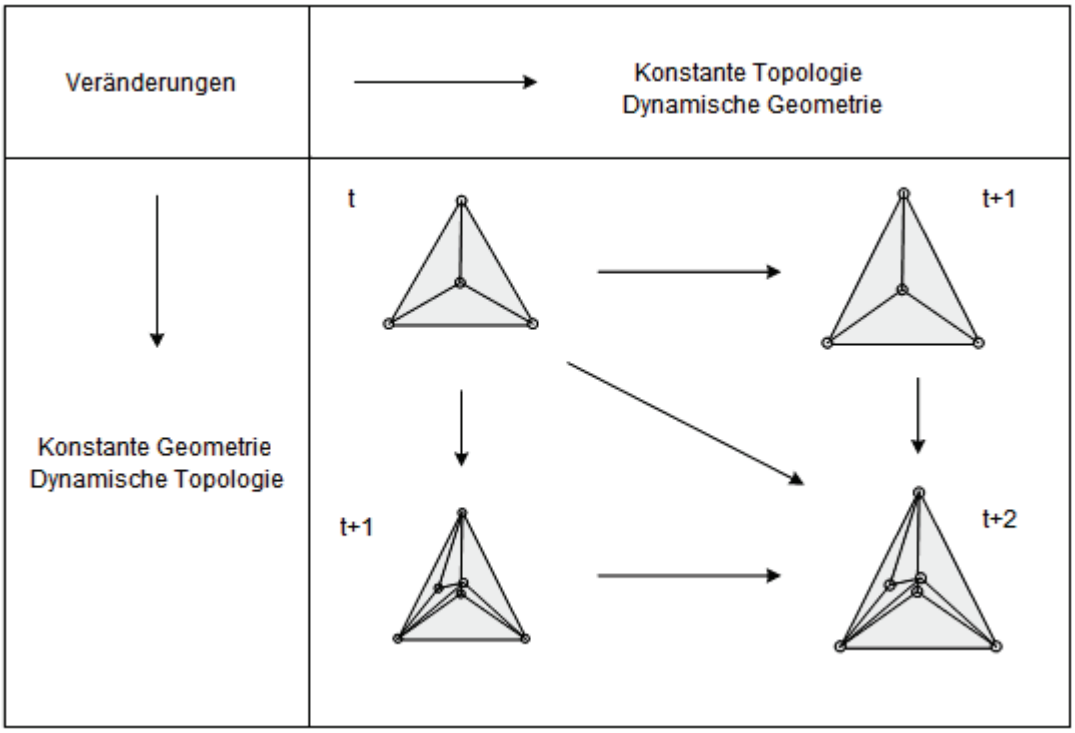

Abb. 6.6: Beispiel für eine raum-zeitliche Variabilität von Geoobjekten (vgl. [Breu01])

6.4.2 Thematisch-zeitliche Variabilität eines Geoobjekts

Bei thematischen Veränderungen von Geoobjekten müssen lediglich die sachlichen Attribute bzw. die Metadaten aktualisiert werden, da die räumlichen Eigenschaften unverändert bleiben. Aufgrund dessen sind im vorgestellten Modell sowohl Gültigkeits- als auch Transaktionszeit mit den Metadaten der Objekte in Beziehung gesetzt worden. Ein Beispiel für eine solche thematisch-zeitliche Variabilität ist die Änderung einer Nutzungsart für ein Gebäude ausgehend von einer linearen Ordnung (siehe Abb. 6.7).

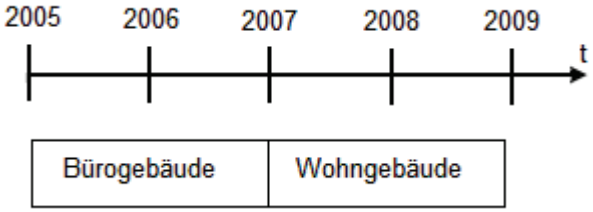

Abb. 6.7: Beispiel für eine thematisch-zeitliche Variabilität von Geoobjekten

Eine Kopplung von raum-zeitlicher und thematisch-zeitlicher Variabilität ist ebenso denkbar.

6.5 Kritische Beurteilung des vorgestellten Modells

Das vorgestellte relationale 4D-Datenmodell auf Basis von Oracle und TimeDB eignet sich sehr gut zur temporalen Beschreibung von 3D-Geoobjekten. Es lassen sich alle Zustandsänderungen eines Geoobjekts beschreiben. Raum-zeitliche Analysen sind ebenso möglich, lassen sich aber bei aufwändigen Analysen nur über komplexe Anfragen herleiten. Das in Abschnitt vorgestellte geometrische Modell und die damit verbundenen objektrelationalen Konzepte von Oracle (vgl. Abschnitt 5.1: Oracle Spatial 3D) lassen sich aber nicht ohne weiteres mit der temporalen Datenbank TimeDB und deren temporalen Strukturen verknüpfen. Daher bedarf es weiterführender Arbeiten auf dem Gebiet der spatio-temporalen Datenbankentwicklung. In den letzten Jahren wurden unterschiedliche Ansätze in verschiedenen Forschungsprojekten verfolgt. Eine Möglichkeit zur Integration der räumlichen Datentypen, Zugriffsmethoden sowie Funktionen und Operatoren von Oracle Spatial in Ti-

meDB wurde in [CRS06] vorgeschlagen und prototypisch umgesetzt. Für die Realisierung der spatio-temporalen Datenbank wurden alle Kernmodule (vgl. Abschnitt 5.2.2: Architektur des TimeDB DBMS) modifiziert und so angepasst, dass die objektrelationalen Konzepte von Oracle Spatial unterstützt werden. Andere Arbeiten betreffen die Integration von räumlichen und temporalen Operatoren für die effiziente Bearbeitung raum-zeitlicher Anfragen in einer einheitlichen Anfragesprache. Ein Vorschlag zur Implementierung der temporalen Anfragesprache TSQL2 in SQL3 wurde dem ISO SQL3 Komitee von [SBJS00] bereits im Jahr 2000 unterbreitet, bisher aber nicht verwirklicht. Eine weitere prototypische Umsetzung zur Integration von räumlichen und temporalen Beziehungsoperatoren in SQL3 auf Basis eines relationalen Datenmodells findet sich in [Lee02]. LEE implementierte u. a. drei Makros zur Beschreibung von räumlichen sowie temporalen Aspekten (siehe Tab. 6.8).

Tab. 6.8: Spatio-Temporale Operatoren in [Lee02]

Operator	Beispielnutzung
Spatial	- a.*spatial* disjoints b.*spatial* - b.*spatial* contains a.*spatial* - a.*spatial* overlaps b.*spatial*
Valid	- a.*valid* precedes Timestamp 'Nov-31-1998' - a.*valid* meets b.*valid*
Transaction	- a.*transaction* contains PERIOD 'Jan-01-1998, Nov-31-1998'

Darüber hinaus werden multidimensionale Indexierungsverfahren für die effiziente Anfragebearbeitung in spatio-temporalen Datenbanksystemen benötigt. Erste Ansätze zur spatio-temporalen Indexierung von Geodaten werden u. a. in [TNS99], [SJLL00] und [MNBS05] beschrieben.

Es ist zu beachten, dass es derzeit keine Gesamtlösung oder gar eine kommerzielle Lösung gibt, die alle Anforderungen einer spatio-temporalen Datenbank erfüllt. Bis eine zufrieden stellende Lösung für jeden gleichermaßen zur Verfügung steht, bedarf es noch einiger Studien und Untersuchungen in diesem Bereich.

7 Visualisierung

Die Darstellung der zeitlichen Veränderung von Objekten der realen Welt ist eine der wesentlichsten Aufgaben eines 4D-GIS, um raum-zeitliche Variabilitäten von Geoobjekten für den Nutzer zu veranschaulichen. Aufgrund dessen sollen in diesem Kapitel Möglichkeiten der Visualisierung temporaler 3D-Objekte aufgezeigt werden. Dabei werden sowohl Geoinformationssysteme als auch Game-Engines bezüglich ihrer Datenbankanbindung an Oracle, der Verarbeitung drei- bzw. vierdimensionaler Daten und der Visualisierung zeitabhängiger Prozesse analysiert und in die Betrachtung mit einbezogen.

7.1 Geoinformationssysteme

7.1.1 ESRI ArcGIS 9.3

ArcGIS ist die Bezeichnung für verschiedene aufeinander abgestimmte GIS-Softwareprodukte der Firma ESRI. Als Marktführer im Bereich GIS bietet ArcGIS eine Vielzahl von Möglichkeiten zur Visualisierung zeitabhängiger 3D-Geodaten. In diesem Abschnitt wird Bezug auf die Bachelorarbeit „Visualisierung raum-zeitlicher Dynamik“ [GS09] genommen, in der GRUNER und SCHUMACHER aufgezeigt haben, dass es mithilfe der vorhandenen Desktoperweiterungen von ArcGIS 9.3 möglich ist, die raum-zeitliche Variabilität eines Geoobjekts zu berücksichtigen.

7.1.1.1 Datenbankanbindung

ArcGIS unterstützt verschiedene Geodatenmodelle. Dabei wird grundsätzlich zwischen dateibasierter und datenbankbasierter Verwaltung unterschieden. Die dateibasierte Verwaltung (z.B. Shapefiles und Coverages) führt allerdings eine Reihe von Nachteilen mit sich (vgl. Kapitel 3: Datenbanksysteme). Für die effiziente Verwaltung großer Datenmengen erlaubt ArcGIS daher die Nutzung von externen rela-

tionalen DBMS wie Oracle. Der Zugriff erfolgt über den ArcSDE[7]-Geodatenserver, der es ermöglicht das Spatial-Format von Oracle zu übersetzen. ArcSDE ist Bestandteil der ArcGIS-Server Version und liegt in drei verschiedenen Lizenzen vor. Für die Nutzung von Oracle ist die Enterprise Edition erforderlich. Aufgrund fehlender Ressourcen der Hochschule konnte die Datenbankverbindung aber nicht getestet werden.

7.1.1.2 Verarbeitung und Visualisierung von 3D/4D-Daten

ArcGIS bietet eine Vielzahl von Möglichkeiten zur Verarbeitung dreidimensionaler Datenbestände. Mithilfe des 3D-Analyst können dreidimensionale Daten analysiert und dargestellt werden. Des Weiteren lassen sich TIN's und Volumenkörper berechnen und analysieren. Die Verwaltung temporaler Aspekte erfolgt durch die Einführung eines zusätzlichen Attributs Zeit, in der ein benutzerdefinierter Zeitstempel für das jeweilige Objekt abgelegt ist. Dadurch lassen sich aber weder Gültigkeits- noch Transaktionszeit des Objekts beschreiben und keine temporalen Beziehungen (vgl. Abschnitt 4.2.3: TSQL2 Datenmodell) aufbauen und analysieren.

Allerdings bietet ArcGIS sehr gute Ansätze zur Visualisierung von raum-zeitlichen Veränderungen. So können mithilfe von Time-Layer-Animationen (Animation Tool) die Modifikationen von Punktkoordinaten durch Einblenden von Verschiebungsvektoren graphisch repräsentiert werden. Die Darstellung von sich verändernden Volumenkörpern kann durch arithmetische Operationen (Addition, Subtraktion) von Teilkörpern erfolgen. Eine ausführliche Beschreibung zur graphischen Darstellung der raum-zeitlichen Variabilität von Geoobjekten findet sich in [GS09]. ArcGIS bietet einen ersten Ansatz zur Visualisierung von zeitabhängigen 3D-Daten, auch wenn es aufgrund fehlender temporaler Strukturen nicht als 4D-GIS bezeichnet werden kann.

[7] Spatial Data Engine (SDE)

7.1.2 Autodesk AutoCAD Map 3D 2010

AutoCAD Map 3D ist ein Softwareprodukt der Firma Autodesk, das auf der neuesten Version von AutoCAD basiert und um eine Reihe GIS-spezifischer Funktionen und Werkzeuge erweitert wurde. Map 3D vereinigt demnach alle Vorteile eines CAD- und eines GIS-Programms. Der Zusatz „3D“ im Produktnamen lässt zudem auf eine vollständige 3D-Funktionalität schließen, sodass eine Untersuchung der Software auf 4D-Funktionalitäten zweckgemäß wäre.

7.1.2.1 Datenbankanbindung

AutoCAD Map 3D nutzt für den Zugriff auf externe relationale Datenbanken die unter LGPL stehende FDO[8] Datenzugriffstechnologie. FDO stellt eine API zur Verfügung, mit der räumliche Daten manipuliert, beschrieben und analysiert werden können [FDO07]. Abbildung 7.1 beschreibt die Architektur der FDO-Technologie.

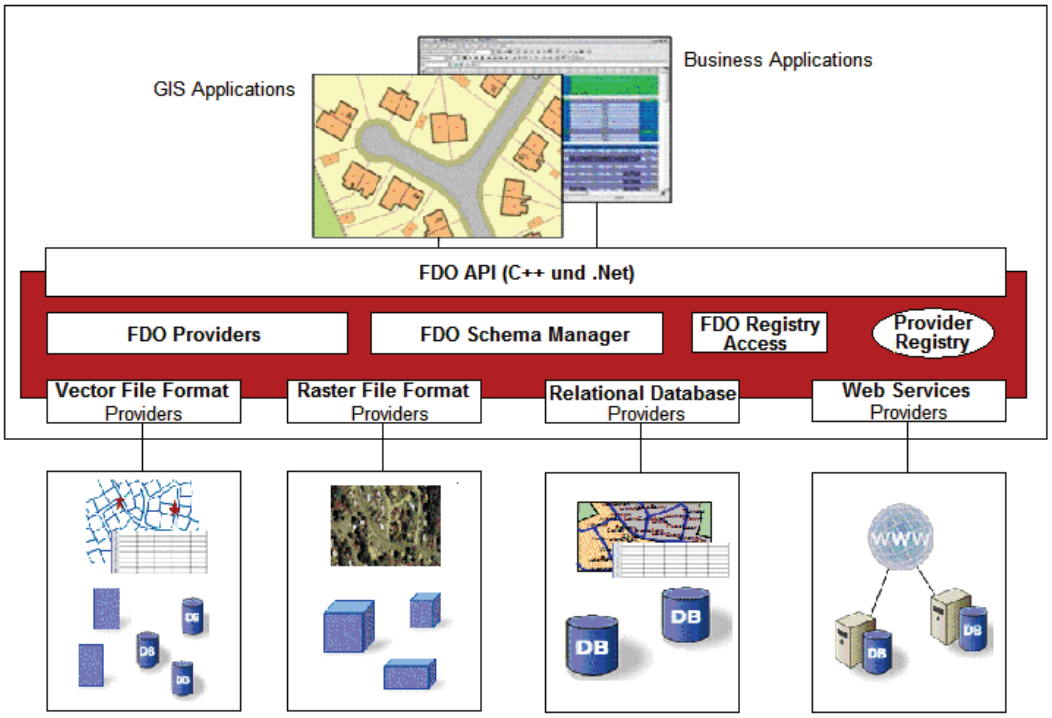

Abb. 7.1: FDO-Architektur [FDO07]

[8] Feature Data Object; API verfügbar unter: http://fdo.osgeo.org

7.1.2.2 Verarbeitung und Visualisierung von 3D/4D-Daten

Der Vorteil der FDO-Technologie ist, dass sich dreidimensionale Geodaten direkt aus der Datenbank visualisieren lassen. Die dargestellten Geometrien und in der Tabelle gespeicherten Sachdaten lassen sich ohne weiteres editieren und in die Datenbank zurück schreiben. Dreidimensionale Geometrien lassen sich allerdings nur im zweidimensionalen Raum editieren. Volumenkörper (SOLID und MULTISOLID) lassen sich zwar in AutoCAD erstellen, können aber nicht in der Datenbank gespeichert werden. Die in der Datenbank gespeicherten Volumenkörper werden nicht unterstützt. AutoCAD bietet sehr gute Möglichkeiten zur Auswertung und Visualisierung dreidimensionaler Geometrien. Allerdings bietet die Software keine Möglichkeit der Animation. Temporale Strukturen lassen sich zur graphischen Repräsentation nicht berücksichtigen, sodass AutoCAD als Basis für ein 4D-GIS nicht in Frage kommt.

7.2 Game-Engines

Die Möglichkeiten, die Game-Engines bei der Visualisierung zeitabhängiger Daten bieten, sind natürlich weit aus größer als bei klassischen Geoinformationssystemen. Die Motivation zum Einsatz von Game-Engines liegt u.a. in Anwendungen begründet, in denen photorealistische Darstellungen von 3D-Objekten gefordert sind.

7.2.1 Blender

Blender ist eine unter GPL stehende Software zur Erstellung von 3D-Graphiken. Sie enthält Funktionen zur Modellierung dreidimensionaler Körper sowie Werkzeuge zur Texturierung. Weiterhin lassen sich mit der Software aufwändige Animationen generieren.

7.2.1.1 Datenbankanbindung

Eine standardmäßige Datenbankschnittstelle zu relationalen Datenbanken wie in den zuvor beschriebenen Geoinformationssystemen ist in Blender nicht vorhanden. Allerdings lässt sich eine entsprechende Datenbankverbindung durch die in Blender verwendete Skriptsprache Python programmieren.

7.2.1.2 Verarbeitung und Visualisierung von 3D/4D-Daten

Blender verfügt als Game-Engine von Hause aus natürlich nicht über die Fähigkeiten bei der Analyse und Auswertung räumlicher Daten. Allerdings lässt sich der Funktionsumfang durch entsprechende Programme erheblich erweitern und auf die Anforderungen für ein 4D-GIS anpassen. So sind z.B. in Python programmierte Pakete zur Beschreibung von topologischen Beziehungen (9-Intersection-Modell) oder Kollisionsprüfungen verfügbar. Darüber hinaus kann Blender als Game-Engine seine Stärken im Bereich Animation und Visualisierung ausspielen. Für die Animation werden zumeist Key-Frame-Animationen verwendet, d.h. dass zu festgelegten Zeitpunkten die Positionen des zu animierenden Objekts gespeichert werden. Diese Methode eignet sich daher sehr gut zur Visualisierung raum-zeitlicher Dynamik. Eine entsprechende erste Umsetzung auf Basis eines objektorientierten Ansatzes findet sich in [Hill09].

7.2.2 Autodesk 3ds Max 2009

3ds Max ist eine kommerzielle Software der Firma Autodesk zur Erstellung von 3D-Graphiken und Animationen. Die Software findet v. a. im Bereich der Computerspiel- und Filmindustrie sowie in Architektur- und Designbüros seine Anwendung.

7.2.2.1 Datenbankanbindung

3ds Max verfügt, wie Blender, über keine standardmäßige Datenbankschnittstelle zu Oracle oder anderen relationalen DBMS. Allerdings ist es möglich, entsprechen-

de Datenbankverbindungen in der objektorientierten Programmiersprache C# zu entwickeln. Hinreichende Klassen stehen in C# bereits zur Verfügung.
Eine Alternative zur Eigenentwicklung würde die Implementierung der in Abschnitt 7.1.2.1 vorgestellten FDO Datenzugriffstechnologie in 3ds Max darstellen. In Softwarelösungen im Bereich GIS (AutoCAD Map 3D) und 3D-Stadtmodelle (Autodesk LandXplorer) ist diese bereits verfügbar.

7.2.2.2 Verarbeitung und Visualisierung von 3D/4D-Daten

Die Analyse und Auswertung raumbezogener Daten gestaltet sich schwierig, da die Software nicht als Informationssystem sondern als Modellierungs- und Animationswerkzeug konzipiert wurde. Durch entsprechende anwendungsspezifische Programmierungen kann der Funktionsumfang aber auch bei 3ds Max erweitert werden, was aber zu einem erheblichen Mehraufwand führt. Die Stärken der Software sind aber auch in diesem Fall in der photorealistischen Modellierung und Visualisierung von Objekten der realen Welt sowie deren Animation zu sehen. So lässt sich die Dynamik von 3D-Objekten sehr gut mithilfe von Key-Frame-Animationen visualisieren.

7.3 Zusammenfassung

Der Einsatz von Geoinformationssystemen und Game-Engines für die Visualisierung zeitabhängiger Prozesse hängt stark vom spezifischen Anwendungsgebiet ab. GIS eignen sich sehr gut zur Analyse und Auswertung räumlicher Daten. Darüber hinaus bieten viele GIS standardmäßige Datenbankschnittstellen zu relationalen Datenbankmanagementsystemen. Für Anwendungen, in denen photorealistische Darstellungen gefordert sind, stellen vorhandene GIS allerdings keine zufrieden stellende 3D-Graphik zur Verfügung. In diesen Fällen wäre der Einsatz von Game-Engines eine Alternative, sofern auf etwaige Analysefunktionen verzichtet werden kann. Darüber hinaus wäre der zu betreibende Programmieraufwand relativ hoch, um Game-Engines um zwingend erforderliche Analysefunktionen zu erweitern.

8 Diskussion

Die Entwicklung eines 4D-GIS, das alle Anforderungen und Aspekte von der Datenhaltung, über die Analyse bis hin zur Visualisierung gleichermaßen umfasst und berücksichtigt, ist eine interessante, aber vor allem anspruchsvolle Aufgabe für die weitere Forschung. Der Hauptschwerpunkt der vorliegenden Arbeit wurde auf die effiziente Speicherung raum-zeitlicher Daten gelegt. Da relationale und objektrelationale Datenbankmanagementsysteme noch unzureichende Mittel zur Verwaltung temporaler Daten besitzen, wurde das vorgestellte 4D-Datenmodell auf Basis zweier Datenbanktechnologien entwickelt: TimeDB und Oracle. Das vorgestellte relationale topologische 4D-Datenmodell bietet dabei einen Ansatz zur Verwaltung zeitabhängiger 3D-Geodaten. Darüber hinaus wird aufgezeigt, wie sich räumliche und thematische Eigenschaften von Geoobjekten mit temporalen Aspekten miteinander in Beziehung setzen lassen. Es lassen sich alle zeitlichen Zustandsänderungen von Geoobjekten beschreiben. Aufwändige Raum-Zeit-Analysen lassen sich hingegen nur über komplexe Anfragen herleiten. Es bedarf daher weiterführender Arbeiten im Bereich der spatio-temporalen Datenbankentwicklung. Dabei können verschiedene Ansätze verfolgt werden.

Eine mögliche Weiterentwicklung betrifft die Integration der räumlichen Datentypen, Zugriffsmethoden sowie Funktionen und Operatoren von Oracle Spatial (vgl. Abschnitt 5.1: Oracle Spatial 3D) mit der temporalen Datenbank TimeDB. Mit der Realisierung eines spatio-temporalen DBMS auf Basis von Oracle Spatial und TimeDB wäre auch die Entwicklung einer geeigneten raum-zeitlichen Abfragesprache für die Analyse spatio-temporaler Daten umgesetzt. Ein anderer Ansatz wäre die Erweiterung des Sprachstandards SQL3 um temporale Funktionen und Operatoren. Der Vorteil dieser Lösung ist v. a. in der flexiblen Einsatzfähigkeit in verschiedenen DBMS zu sehen. Eine entsprechende Standardisierung in Form vom SQL/Temporal ist bereits seit mehreren Jahren angedacht. Darüber hinaus werden multidimensionale Indexierungsverfahren benötigt, um leistungsfähige Zugriffe auf große Datenmengen sicherzustellen. Derartige Ansätze könnten in entsprechenden Forschungsprojekten an der Hochschule bearbeitet werden.

Neben einer effizienten Datenhaltung und der Durchführung raum-zeitlicher Analysen ist die Visualisierung der zeitlichen Veränderungen von 3D-Objekten ein zentraler Aspekt bei der Entwicklung eines 4D-GIS. Entsprechende Möglichkeiten wurden bereits in Kapitel 7 diskutiert. Eine Alternative würde eine Eigenentwicklung darstellen, in der sich die Anfrageergebnisse direkt in einem 3D-Fenster visualisieren lassen. In ersten prototypischen Implementierungen könnte bei der graphischen Darstellung zunächst auf statische Repräsentationen zurückgegriffen werden.

Es bedarf somit noch einiger Untersuchungen und Studien für die Entwicklung eines flexibel und effizient einsetzbaren 4D-GIS, das alle in Abschnitt 2.2 definierten Anforderungen gleichermaßen abbildet. Mittelfristig ist ein entsprechendes 4D-GIS nach Meinung des Autors aber nicht absehbar.

Abkürzungsverzeichnis

Numerisch

3D-FDS	3D Formal Data Structure; Topologisches Modell nach MOLENAAR
9IM	9-Intersection-Modell

A

API	Application Programming Interface
ATSQL2	Applied Temporal SQL2; angewandte temporale Erweiterung des SQL-92 Standards

B

BCDM	Bitemporal Conceptual Data Model

C

CAD	Computer-Aided Design; rechnergestützter Entwurf
CSG	Constructive Solid Geometry

D

DB	Datenbank
DBMS	Datenbankmanagementsystem
DBS	Datenbanksystem
DDL	Data Definition Language; Datendefinitionssprache von SQL
DML	Data Manipulation Language; Datenmanipulationssprache von SQL

E

EPSG	European Petroleum Survey Group; EPSG-Schlüssel dienen der Identifikation räumlicher Bezugssysteme
ESRI	Environmental Systems Research Institute Inc.; Sitz: Redlands, CA
EVAP	Eingabe, Verarbeitung, Analyse, Präsentation (4 Komponenten eines GIS)

F

FDO	Feature Data Object; Datenzugriffstechnologie unter LGLP

G

GIS	Geoinformationssystem

I

IQL	Interactive Query Language; Interaktive Abfragesprache von SQL
ISO	International Organization for Standardization

L

(L)GPL	GNU (Lesser) General Public Licence

M

MBR	Minimal Bounding Rectangle; minimal umgebendes Rechteck

P

Pixel	Kunstwort, zusammengesetzt aus Picture und Element

S

SDE	Spatial Data Engine; Geodaten-Server von ESRI
SDO	Spatial Data Option
SQL	Structured Query Language
SRID	Spatial Reference Identifier; ID eines räumlichen Bezugssystems
SVVM	Single Vector Value Map

T

TGIS	Temporales Geoinformationssystem
TIN	Triangulated Irregular Network
TSQL2	Temporal SQL; temporale Erweiterung des SQL-92 Standards

U

UDM	Urban Data Model; Topologisches Datenmodell nach COORS
UML	Unified Modelling Language
UTC	Universal Time Coordinated; koordinierte Weltzeit

Literaturverzeichnis

[Allen83] Maintaining Knowledge about Temporal Intervals. James F. Allen. Communications of the ACM. 1983

[Bart05] Geoinformatik – Modelle, Strukturen, Funktionen, 4. Auflage. Norbert Bartelme. Springer Verlag, Berlin, Heidelberg. 2005
[ISBN: 3-540-20254-4]

[BF94] Grundlagen der Geoinformationssysteme – Hardware, Software und Daten. Band 1, 2. Auflage. Ralf Bill, Dieter Fritsch. Herbert Wichmann Verlag, Heidelberg. 1994

[Breu01] On the Way to Component-Based 3D/4D Geoinformation Systems. Martin Breunig. Springer Verlag, Berlin, Heidelberg, New York. 2001
[ISBN: 3-540-67806-9]

[Breu05] Räumliche Repräsentationen. Martin Breunig. In: 3D-Geoinformationssysteme – Grundlagen und Anwendungen. Volker Coors, Alexander Zipf. Herbert Wichmann Verlag, Heidelberg. 2005
[ISBN: 3-87907-411-9]

[Brink08] Geodatenbanksysteme in Theorie und Praxis – Einführung in objekt-relationale Geodatenbanken unter besonderer Berücksichtigung von Oracle Spatial, 2. Auflage. Thomas Brinkhoff. Herbert Wichmann Verlag, Heidelberg. 2008 [ISBN: 978-3-87907-472-3]

[BZ01] Lexikon der Geoinformatik. Ralf Bill, Marco Zehner. Herbert Wichmann Verlag, Heidelberg. 2001 [ISBN: 3-87907-364-3]

[Coors04] 3D-Infrastrukuren. Volker Coors. Seminar GIS & Internet, Fachhochschule Stuttgart. 2004

[Coors05] Topologische Modelle für 3D-Geodatenbanken. Volker Coors. In: 3D-Geoinformationssysteme – Grundlagen und Anwendungen. Volker Coors, Alexander Zipf. Herbert Wichmann Verlag, Heidelberg. 2005 [ISBN: 3-87907-411-9]

[CRS06] A Spatio-Temporal Database System based on TimeDB and Oracle. A. Carvalho, C. Ribeiro, A. Sousa. In: International Federation for Information Processing. Volume 205. Research and Pracitcal Issues of Enterprise Information Systems. 2006

[CZ05] 3D-Geoinformationssysteme – Grundlagen und Anwendungen. Volker Coors, Alexander Zipf. Herbert Wichmann Verlag, Heidelberg. 2005 [ISBN: 3-87907-411-9]

[FDO07] FDO Data Access Technology – Project Home, OSGeo. Webseite zum FDO-Projekt. 2007
Online unter: http://fdo.osgeo.org (Letzter Zugriff: 08.09.2009)

[Flick99] Konzeption eines adaptiven Frameworks für 3D-GIS. Sascha Flick. Dissertation zur Erlangung des akademischen Grades eines Doktor-Ingenieurs (Dr.-Ing.). Fachbereich Informatik, Technische Universität Darmstadt. 1999

[Foley97] Computer Graphics: Principles and Practise. James D. Foley. Addison Wesley Verlag. 1997

[GS09] Visualisierung raum-zeitlicher Dynamik. Matthias Gruner, Benjamin Schumacher. Bachelorarbeit Geoinformatik, Hochschule Neubrandenburg. 2009

[Güt94] An Introduction to Spatial Database Systems. Ralf Hartmut Güting. Praktische Informatik IV, FernUniversität Hagen. 1994

[Hill09] Photorealistisches 4D-GIS im Altbergbau. Tobias Hillmann. In: Angewandete Geoinformatik 2009 – Beiträge zum 21. AGIT Symposium Salzburg. Josef Strobl, Thomas Blaschke, Gerald Griesebner. Wichmann Verlag. 2009

[Hsu09] Constructive Solid Geometry and Sweep Representation. Yeh-Liang Hsu. Yuan Ze University (China). 2009

[ISO19108] International Organization for Standardization (ISO): ISO 19108:2002 – Geographic Information – Temporal Schema. 2002

[JSS94] The TSQL2 Data Model. Christian S. Jensen, Richard T. Snodgrass, Michael D. Soo. In: The TSQL2 Temporal Query Language. Richard T. Snodgrass. Kluwer Academic Publishers. 1995

[Kada] Generalisation of 3D Building Models by Cell Decomposition and Primitive Instancing. Martin Kada. Institut für Photogrammetrie, Universität Stuttgart

[KGB07] Pro Oracle Spatial for Oracle Database 11g. Ravi Kothuri, Albert Godfrind, Euro Beinat. Apress. 2007 [ISBN: 978-1-59059-899-3]

[Lange02] Geoinformatik in Theorie und Praxis. Norbert de Lange. Springer Verlag, Berlin, Heidelberg, New York. 2002 [ISBN: 3-540-43286-8]

[Lee02] Integrating Spatial and Temporal Relationship Operators into SQL3 for Historical Data Management. Jong-Yun Lee. ETRI Journal, Volume 24, Number 3. 2002

[MNBS05] RDBMS Support for Efficient Indexing of Historical Spatio-Temporal Point Data. Daniel Mallett, Mario A. Nascimento, Viorica Botea, Joerg Sander. A TimeCenter Technical Report. 2005

[Mol90] A Formal Data Structure for Three Dimensional Vector Maps. Martien Molenaar. Proceedings of the 4th International Symposium on Spatial Data Handling. 1990

[Oracle07] Oracle Database – Number One Database. Juli 2008
Online unter: http://www.oracle.com/database/number-one-database.html (Letzter Zugriff: 10.08.2009)

[OS01] Time-Integrative Geographic Information Systems. Thomas Ott, Frank Swiaczny. Springer Verlag, Berlin, Heidelberg, New York. 2001 [ISBN: 3-540-41016-3]

[OSDG09] Oracle® Spatial Developer's Guide 11g Release 1. 2009
Online unter:
http://www.oracle.com/pls/db111/to_pdf?pathname=appdev.111/b28400.pdf (Letzter Zugriff: 03.08.2009)

[Sauer98] Relationale Datenbanken: Theorie und Praxis, 4. Auflage. Hermann Sauer. Addison Wesley Longman Verlag GmbH, Bonn. 1998
[ISBN: 3-8273-1381-3]

[SBJS00] Transitioning Temporal Support in TSQL2 to SQL3. Richard T. Snodgrass, Michael H. Böhlen, Christian S. Jensen, Andreas Steiner. In: Temporal Database Management. Christian S. Jensen. Dissertation zum Erlangen des Titels "Doctor of Technical Science", Tucson, Arizona (USA). 1999

[SJLL00] Indexing the Positions of Continuously Moving Objects. S. Saltenis, C. Jensen, S.T. Leutenegger, M.A. Lopez. Proceedings ACM SIGMOD International Conference on Management of Data. Dallas, Texas (USA). 2000

[Snod] TSQL2 and SQL3 Interactions. Richard T. Snodgrass. Webseite des Department of Computer Sciences, University of Arizona (USA)

Online unter: http://www.cs.arizona.edu/~rts/sql3.html
(Letzter Zugriff: 02.09.2009)

[Snod95] The TSQL2 Temporal Query Language. Richard T. Snodgrass, Kluwer Academic Publishers. 1995

[Ste98] A Generalisation Approach to Temporal Data Models and their Implementations. Andreas Steiner. Dissertation zum Erlangen des Titels “Doctor of Technical Science”, ETH Zürich (Schweiz). 1998

[TSN99] On the Generation of Spatiotemporal Datasets. Yannis Theodoridis, Jefferson R. O. Silva, Mario A. Nascimento. A TimeCenter Technical Report. 1999

[WWSL] On 3D GIS Spatial Modeling. Wang Yanbing, Wu Lixin, Shi Wenzhong, Liu Xiaomeng. ISPRS Workshop on Updating Geospatial Databases with Imagery & The 5th ISPRS Workshop on DMGISs, Peking (China), Hongkong (China)

[ZCFS97] Advanced Database Systems. Carlo Zanioli, Stefano Ceri, Christos Faloutsos, Richard T. Snodgrass. Morgan Kaufman Publishers, San Francisco, Kalifornien (USA). 1997

[ZK01] Ein objektorientierter Framework für temporale 3D-Geodaten. Alexander Zipf, Sven Krüger. AGIT 2001, Symposium für Angewandte Geographische Informationsverarbeitung, Salzburg (Österreich). 2001

[ZRS02] Topology for 3D Spatial Objects. Sikya Zlatanova, Alias Abdul Rahman, Wenzhong Shi. Delft (Niederlande), Johor (Malaysia), Hongkong (China). 2002

Abbildungsverzeichnis

Tabellenverzeichnis