Technische Universität München
Geodätisches Institut
Fachgebiet Geoinformationssysteme

Untersuchung geographischer Anfragesprachen auf der Basis relationaler und objektrelationaler Datenbankmanagementsysteme

Dipl.-Inform. Univ. Matthias Ziegler

Vollständiger Abdruck der von der Fakultät für Bauingenieur- und Vermessungswesen der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender: Univ.-Prof. Dr. phil. nat. Markus Rothacher

Prüfer der Dissertation:
1. Univ.-Prof. Dr.-Ing. Matthäus Schilcher
2. Univ.-Prof. Dr.-Ing. Liqiu Meng
3. Univ.-Prof. Dr. rer. nat., Dr. rer. nat. habil. Gunnar Teege, Universität der Bundeswehr München

Die Disseration wurde am 15.05.2001 bei der Technischen Universität München eingereicht und durch die Fakultät Bauingenieur- und Vermessungswesen am 30.11.2001 angenommen.

Dank

Mein besonderer Dank gilt Herrn Prof. Dr. Matthäus Schilcher. Er hat die Arbeit nicht nur in fachlicher Hinsicht begleitet, sondern vermochte auch die nötige Zuversicht zu vermitteln, ein solches Projekt gegenüber allen Schwierigkeiten zu einem positiven Ende zu bringen. Frau Prof. Dr. Liqiu Meng danke ich für die freundliche Übernahme des Zweitgutachtens und Herrn Prof. Dr. Bernhard Mitschang für die zahlreichen gedanklichen Anstöße, die ich zu der Arbeit von ihm erhalten habe. Danken möchte ich auch allen Kollegen am Fachgebiet Geoinformationssysteme und am Lehrstuhl Datenbanksysteme für die gute Zusammenarbeit, insbesondere Herrn Robert Roschlaub für die wiederholte Durchsicht der Arbeit und die hilfreichen Verbesserungsvorschläge.

Die Firmen Siemens Nixdorf Informationssysteme AG und SICAD Geomatics gewährten nicht nur ein Stipendium für die Dauer der Arbeit, sondern stellten auch Rechner, Arbeitsplatz und nicht zuletzt eine Arbeitsumgebung, in der ich viel lernen konnte, zur Verfügung. Mein Dank gebührt hier meinen Vorgesetzten Christian Singer, der den Anstoß zu der vorliegenden Arbeit gab, und Dr. Peter Ladstätter, der sie die letzten drei Jahre mit Vorschlägen und Anregungen unterstützte. Nicht vergessen werden dürfen an dieser Stelle auch alle Personen bei SICAD, die mir eine angenehme Arbeitsatmosphäre geboten und mit Rat und Tat zur Seite gestanden haben.

Die untersuchten Anwenderdaten und -anfragen stellten freundlicherweise die Kartographische Anstalt des Bayerischen Staatsministeriums für Ernährung, Landwirtschaft und Forsten und das Vermessungsamt der Stadt München zur Verfügung. Ich danke Herrn Georg Lother, Herrn Dr. Hans-Otto Schmidtbauer und Herrn Ernst Eidelsburger für die fruchtbare Zusammenarbeit.

Abschließend möchte ich auch ganz herzlich meiner Familie und meiner Frau Inga danken, die mich in dieser Zeit entbehrt und ertragen haben.

München, im April 2001

Matthias Ziegler

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1. Kurzfassung

Problemstellung:

Verschiedene Trends beeinflussen die Entwicklung von Geoinformationssystemen (GIS). Davon wirken sich zwei besonders auf die Beziehung zwischen Datenbanksystemen und den darauf aufbauenden Geoinformationssystemen aus:

- Zunehmende Integration:
  Bestehende Informationsstrukturen in Unternehmen und Verwaltungen beziehen zunehmend Geoinformationssysteme als eine Teilkomponente mit ein. Dies erfordert, die Geodaten stärker mit anderen alphanumerischen Daten zu integrieren, die heute überwiegend von relationalen Datenbankmanagementsystemen (RDBMS) verwaltet werden.
- Neue Anwendungen:
  Bisher erfaßten und aktualisierten isolierte Anwender geographische Daten mit proprietären Geoinformationssystemen. Nun gewinnen zusätzlich themenübergreifende Online-Analysen an Bedeutung, zum Beispiel in Echtzeit durchgeführte Anfragen für Location Based Services über Mobiltelefon. Zur Beantwortung solcher gemischt geographisch-alphanumerischer Anfragen ist der gleichzeitige Zugriff auf standardisiert gespeicherte Geometriedaten und damit verknüpfte Sachdaten notwendig.

Diese beiden Trends steigern die Bedeutung der Standard-Datenbanken und ihrer Integration in GIS. In RDBMS greift man auf die Daten durch eine das Ergebnis beschreibende (deklarative) Sprache zu, die Structured Query Language (SQL). In GIS setzt man dagegen prozedural und in mehreren interaktiven Schritten das Ergebnis zusammen. Die Bandbreite der Ergebnisse reicht dabei von einer Auswahl einzelner Sachsätze über die Darstellung graphischer Elemente bis zur Erstellung einer kompletten Karte mit Hintergrundbild und Legende.

Mit der zunehmenden Integration von Datenbanken in GIS wird versucht, GIS-Operationen in SQL-Anfragen auszudrücken. Dies erfordert, entweder das GIS oder das DBMS zu erweitern:

- Es gibt Versuche, GIS um an SQL angelehnte Anfragesprachen zu erweitern. Ihr Problem ist die Optimierung und Vearbeitung gemischter Anfragen auf Geometrien und Sachdaten, da die Geometrien in der Regel vom GIS außerhalb des RDBMS gespeichert und verarbeitet werden.
- Zahlreiche Beispiele experimenteller Systeme belegen die Versuche, DBMS zu erweitern. Sie konzentrieren sich häufig auf die Weiterentwicklung einer speziellen Verarbeitungs-, Optimierungs- oder Zugriffsmethode. Es mangelt an der Untersuchung mit aus der GIS-

Praxis kommenden Daten und Anfragen. Es bleibt, die Eignung der erweiterten DBMS für echte GIS-Anwendungen zu belegen, oder die Defizite aufzuzeigen.

Bisher konzentrieren sich Beiträge auf diesem Gebiet auf eine der beiden Möglichkeiten. Außerdem beziehen sie die Anforderungen realer Anwenderdaten und -anfragen nicht ausreichend ein, um GIS-Anwender wirkungsvoll bei der Entscheidung für einen Ansatz unterstützen zu können.

Beitrag der Arbeit:

Die vorliegende Arbeit realisiert ein **GIS mit integrierter geographischer Anfragesprache**. Sie wählt dafür zwei unterschiedliche Ansätze und vergleicht sie aus Anwendersicht.

- Der erste Ansatz **definiert** die Anfragesprache GQL und **implementiert** sie in einem GIS, das auf einem RDBMS basiert. Dies beinhaltet die Implementierung der Anfrageverarbeitung und -optimierung der räumlichen Operatoren. Die volle Mächtigkeit der bisherigen GIS-Datentypen und -Funktionen steht dem Anwender für Anfragen zur Verfügung, die bestehenden Geodaten können unverändert beibehalten werden.
- Der zweite Ansatz **verwendet** ein ORDBMS mit einer Komponente, die einen Datentyp zur Speicherung räumlicher Daten bereitstellt, SQL um standardisierte räumliche Operatoren erweitert und räumliche Anfragen optimiert und verarbeitet. Die Arbeit **modelliert** bestehende Geodaten objektrelational, **transferiert** sie vom GIS/RDBMS zum ORDBMS und **integriert** das erweiterte Datenbanksystem in die GIS-Anwendung.

Die Arbeit **testet** beide Ansätze sowohl mit synthetischen Testdaten, wie dem Sequoia 2000 Benchmark, als auch mit realen Anwenderdaten und -anfragen, die aus dem laufenden Produktionsprozeß der beiden Projektpartner, dem Vermessungsamt der Stadt München und der Kartographischen Anstalt des Bayerischen Staatsministeriums für Ernährung, Landwirtschaft und Forsten, stammen. Die Verwendung realer Daten deckt verborgene Probleme auf und liefert wertvolle Erkenntnisse für die tatsächliche Einsetzbarkeit.

Die Arbeit **vergleicht** beide Ansätze, zeigt ihre Vor- und Nachteile und **erarbeitet**, welche Klassen von GIS-Problemen sich mit welchem Ansatz besser lösen lassen. Sie **zeigt**, wie Geodaten in ORDBMS geeignet modelliert werden können und **empfiehlt**, wie sich GIS weiterentwickeln können, um objektrelationale Datenbanken zu nutzen. Sie schließt mit einem Ausblick auf das zukünftige Verhältnis von Datenbanksystemen und Geoinformationssystemen.

2. Motivation und Einführung

Motivation

In den letzten Jahren hat sich die Forschung und die Technologie im Bereich Geoinformationssysteme in großen Schritten weiterentwickelt. Der Schwerpunkt der Forschung hat sich von Anfragesprachen sowie der Verarbeitung und Optimierung von Anfragen auf andere Bereiche verschoben und ausgedehnt. Davon unabhängig bleiben zum Teil die alten Probleme, wie zum Beispiel die Integration von Geoinformationssystemen (GIS) und Datenbanken, bestehen (siehe [SSD 95], Editorial). Es besteht eine Lücke zwischen den Ergebnissen der Forschung und ihrem Einsatz in realen Anwendungen in der Praxis.

Die vorliegende Arbeit will dazu beitragen, diese Lücke zu schließen, zu beschreiben, welche Probleme dabei auftreten und wie man sie lösen kann, indem sie zwei Ansätze zur *Integration einer geographischen Anfragesprache in ein GIS* realisiert, unter realen Einsatzbedingungen testet und die Ergebnisse aus der Sicht des GIS-Anwenders aufbereitet. Unter GIS-Anwender wird in dieser Arbeit ein Power-User verstanden, der die Programmiermöglichkeiten des GIS beherrscht, im Gegensatz zu einem End-User, der durch eine vorgefertigte Menüführung unterstützt aber auch eingeschränkt wird.

Forschungsfragen

Im Rahmen dieser Arbeit stellen sich drei Gruppen von Fragen:

1. Anwenderanforderungen:
   Welche Anforderungen hat der Benutzer bei der Entwicklung von GIS-Applikationen auf der Basis einer RDBMS-basierten Geodatenbank bezüglich der Anfragen? Wie geht der Anwender bei Suchanfragen typischerweise vor? Welche Operatoren braucht eine geographische Anfragesprache?
2. Datenbanksysteme als Basis:
   Welche Vor- und Nachteile hat die Implementierung einer Anfragesprache im GIS gegenüber der Verwendung der Anfragesprache einer Datenbankerweiterung? Wie eignen sich relationale und objektrelationale DBMS als Basis für die Entwicklung einer GIS-Applikation mit integrierter Anfragesprache?
3. Test und Ergebnisse:
   Wie kann man zu verläßlichen Aussagen über die Eignung der Sprache für reale Anwendungen kommen? Wie sind die Ergebnisse aus der Sicht des Anwenders zu bewerten?

Umfeld der Arbeit

Die Arbeit bezieht zur Beantwortung dieser Fragen Forschung, Anwender und Hersteller gleichermaßen ein. Sie erarbeitet die Anforderungen der Nutzer in Zusammenarbeit mit der Kartographischen Anstalt des Bayerischen Staatsministeriums für Ernährung, Landwirtschaft und Forsten und dem Vermessungsamt der Stadt München. Dabei dienen die Anwendungen SICAD-Forst und Digitale Stadtgrundkarte als Referenz. In die Entwicklung und den Test der realisierten Systeme fließen die Erfahrungen von SICAD Geomatics mit der Entwicklung von Geoinformationssystemen ein.

Überblick

Kapitel 3 führt in den Bereich der Datenbank- und Geoinformationssysteme ein. Es schildert die Besonderheiten von Geodaten, stellt grundlegende GIS-Architekturen und Arten von DBMS vor und skizziert, wie DBMS in GIS integriert werden können.
Kapitel 4 legt die theoretischen Grundlagen der Arbeit im Bereich von Anfragesprachen für geographische Informationssysteme. Es definiert grundlegende Begriffe, verweist auf bisher entwickelte Anfragesprachen und SQL-Spracherweiterungen und geht kurz auf die Verarbeitung und Optimierung von Anfragen sowie die Standardisierung ein. Es führt Beispiele existierender experimenteller und kommerzieller Systeme an, diskutiert ihre Defizite und begründet ihre Eignung für weitere Untersuchungen.
Kapitel 5 faßt die Anforderungen an die geographische Anfragesprache GQL zusammen, definiert darauf aufbauend ihre Eigenschaften, Datentypen und am OGC ausgerichteten Operatoren und beschreibt ihre Implementierung auf der Basis des GIS SICAD.
Kapitel 6 testet die Einsatzfähigkeit dieser Anfragesprache mit synthetischen und realen Anwenderdaten und -anfragen und faßt zusammen, für welche Klassen von Anfragen GQL gut und für welche es weniger gut geeignet ist. Es leitet mit der Aufzählung der Nachteile dieses Ansatzes zum nächsten Kapitel über.
Kapitel 7 untersucht, wie die geographische Anfragesprache eines erweiterten DBMS genutzt werden kann. Es stellt vor, wie die Geodaten objektrelational modelliert und vom GIS/ RDBMS in das ORDBMS transferiert werden. Außerdem gibt es einen Überblick über das Design und die Implementierung des Gesamtsystems.
Kapitel 8 testet das auf einem erweiterten DBMS basierte GIS mit den in *Kapitel 6* für den ersten Ansatz beschriebenen Tests.
Kapitel 9 vergleicht die beiden Ansätze, bewertet die Testergebnisse und folgert, welche Analyseaufgaben durch welchen Ansatz besser bewältigt werden können.
Kapitel 10 faßt die Ergebnisse der Arbeit zusammen und gibt einen Ausblick auf weitere Forschungsmöglichkeiten und die Verwendung der Ergebnisse.

3. Datenbank- und Geoinformationssysteme

Ausgehend von einer Definition von Geodaten und einer Abgrenzung ihrer Eigenschaften von denen anderer Daten (3.1) beschreibt dieses Kapitel, wie Geoinformationssysteme Geodaten speichern (3.2). Anschließend geht es auf den Einsatz von Datenbanken für die Speicherung von Geodaten näher ein (3.3) und erklärt, wie man diese mit Geoinformationssystemen zu einem Gesamtsystem integriert (3.4).

3.1 Geodaten

Der Begriff Geodaten wird in der Literatur nicht einheitlich verwendet. Der folgende Abschnitt erklärt darum die in diesem Zusammenhang verwendeten Begriffe und leitet daraus die im Rahmen dieser Arbeit verwendete Definition des Begriffs Geodaten ab (s. Tab. 1).

<table>
<tr><td rowspan="2">reale Welt</td><td colspan="3">primäres Modell</td><td>sekundäres Modell</td><td>tertiäres Modell</td></tr>
<tr><td colspan="2">Objekt-ebene</td><td>Modell-ebene</td><td>Darstellungs-ebene</td><td>Denk-ebene</td></tr>
<tr><td rowspan="2">reale Objekte</td><td>räumliche Daten</td><td>Geo-Objekte</td><td>Ontologie</td><td>Dienste (Services)</td><td rowspan="2">Benut-zer</td></tr>
<tr><td>Geometrie</td><td>Geometrie + Sachdaten</td><td>Geo-Objekte im Gesamtzusam-menhang (z. B. mit Topologie)</td><td>Graphik (Feature)</td></tr>
<tr><td>Haus, Straße, Baum</td><td>Fläche, Linie, Punkt</td><td>Hausnr., Straßen-name, Baum-katasternr.</td><td>Häuser an einer Allee</td><td>skizzenhafte Bildschirmgraphik (Topogramm), kartographisch ausgestaltete Darstellung auf Papier, 3D-Darstellung</td><td>z. B. Stadt-planer</td></tr>
<tr><td colspan="4">Geo-Objekte im primären Modell repräsentieren reale Objekte in der realen Welt</td><td colspan="2">Graphische Features im sekundären Modell präsentieren Geo-Objekte im primä-ren Modell</td></tr>
</table>

Tab. 1: Geodaten als Modell der realen Welt nach [Meng 02]

Geoinformationen beinhalten vier Komponenten: Geometrie, Sachdaten, Topologie und Graphik. Die **Geometrie** bildet die räumliche Lage und Ausdehnung von Objekten der realen Welt, wie z. B. Bäume, Straßen und Häuser, näherungsweise als Punkte, Linien und Flächen ab. Fügt man zur Geometrie zusätzlich alphanumerische Informationen in Form von **Sachdaten** hinzu, wie z. B. Hausnummer, Straßenname oder Baumkatasternummer, entstehen **Geo-Objekte**. Bringt man die einzelnen Geo-Objekte und ihre Beziehungen zueinander, z. B. ihre **Topologie**, in einen Gesamtzusammenhang, **Ontologie**, so entsteht ein **primäres Modell** als Abbild der realen Welt. Die Geo-Objekte repräsentieren Objekte der realen Welt im Modell.

Die Arbeit verwendet **Geodaten** als allgemeinen Oberbegriff für gespeicherte Daten mit einem räumlichen Bezug, d.h. für gespeicherte Geo-Objekte des primären Modells. Ob die Geo-Objekte schon in einen Gesamtzusammenhang eingeordnet wurden, wird dabei nicht berücksichtigt.

Das primäre Modell wird zur Darstellung in ein **sekundäres Modell** überführt, z. B. als vereinfachte Bildschirmgraphik (Topogramm), als kartographisch ausgestaltete Karte auf Papier oder als dreidimensional berechnete Darstellung in einer Virtual-Reality-Umgebung. Die Sachdaten prägen sich dabei in der Farbe oder anderen Merkmalen der Graphik aus. Die Graphik muß nicht identisch mit der Geometrie sein. Durch Generalisierung werden z. B. Häuser verschoben oder Straßen breiter gezeichnet. Bildschirmgraphik wird oft nicht gespeichert, sondern zur Darstellungszeit berechnet.

Ein graphisch dargestelltes Geo-Objekt nennen wir ein **Feature**. Die graphische Darstellung präsentiert die Geo-Objekte. Es kann verschiedene Präsentationen (Features) für ein und dasselbe Geo-Objekt geben.

Der Benutzer, z. B. ein Stadtplaner, hat bei der Betrachtung der graphischen Darstellung ein mentales Modell in seinem Kopf, wir sprechen hier vom **tertiären Modell**.

Im Gegensatz zu herkömmlichen Daten weisen Geodaten eine Reihe zusätzlicher **Eigenschaften** auf, die im folgenden kurz vorgestellt werden.

1. Hohe Kosten für Erfassung und Pflege
   Im Gesamtsystem aus Geodaten, Hardware und Software haben die Geodaten, insbesondere die Erfassung und Pflege der Geometrie, den höchsten Anteil an den Kosten. Die Lebenszeit der Geodaten überdauert die der Hard- und Software um ein Vielfaches. Daraus ergibt sich ihre hohe Bedeutung [Schilcher 97].

2. Große Datenmengen
   Die in GIS verarbeiteten Daten erreichen die Größenordnung von Gigabyte bis Terabyte. Und die in GIS gehaltene Datenmenge steigt weiter an, da die Erfassung automatisiert und auf weitere Gebiete ausgedehnt wird, und da die Detailgenauigkeit steigt.

3. Zeitliche und räumliche Dimension
   Geo-Objekte haben eine Lage und eine Ausdehnung (spatial location and extent) im kontinuierlichen 2- oder 3-dimensionalen Raum. Der Zeitpunkt ihrer Erfassung oder die Dauer ihrer Gültigkeit fügen eine weitere, die zeitliche, Dimension hinzu (temporal location and extent). Innerhalb dieser Dimensionen existiert keine eindeutige lineare Ordnung [Gaede 96]. Dies erschwert, die Daten zu indizieren und effizient zu suchen.

4. Komplexität und lange Transaktionen
   Die Geometrie von Geodaten wird in komplexen Strukturen abgelegt, die hierarchisch aufeinander aufbauen. Eine Fläche wird z. B. aus den sie begrenzenden Linien zusammengesetzt, diese wiederum aus ihren Stützpunkten.
   Diese Daten zu verändern ist oft ein Prozeß, der nicht in Sekunden abgearbeitet ist, sondern sich über Tage oder Wochen erstrecken kann [Egenhofer & Frank 87]. Dies macht ein Konzept zur Behandlung von langen Transaktionen notwendig.

5. Varianz der Objektgrößen
   Die Geo-Objekte der einzelnen Objektklassen unterscheiden sich stark in ihrer Größe. Ein Punkt läßt sich durch Angabe seiner Koordinaten darstellen. Ein Polygon benötigt dagegen mehrere, mindestens drei, Stützpunkte. Die Anzahl dieser Stützpunkte ist nach oben theoretisch unbegrenzt. Dazu kommen Rasterdaten, bei denen ein Objekt, ein Rasterbild, mehrere Gigabyte beanspruchen kann.

6. Gekrümmte Oberfläche der Erde
   Geodaten beziehen sich im Unterschied zu anderen räumlichen Daten, etwa Konstruktionszeichnungen aus dem Computer Aided Design oder Molekülmodellen in der Chemie, auf die gekrümmte Oberfläche der Erde [Goodchild 90][Egenhofer & Frank 89a]. Diese kann nicht ohne Verzerrungen auf die Ebene der Karte abgebildet werden. Im Gegensatz zu einer Karte kann ein GIS die Daten eines dreidimensionalen Raumes auch ohne Anwendung einer Kartenprojektion speichern.

7. Modellbedingte Abstraktion
   Die geographische Vielfältigkeit ist unendlich komplex. Daten in Geodatenbanken sind notwendigerweise eine Abstraktion der Realität. Eine gewisse Ungenauigkeit in der räumlichen Auflösung, der thematischen und geographischen Abbildung [Goodchild 89] ist dabei gewünscht, um sich auf das Wesentliche zu konzentrieren.

8. Ungenauigkeit der Daten
   Im Gegensatz zu der gewünschten modellbedingten Abstraktion der Daten stehen zu vermeidende Ungenauigkeiten. Sie entstehen bei der Erfassung der Lage mit Meßgeräten, bei der Digitalisierung oder Rasterung der Objekte sowie durch die begrenzt genaue numerische Darstellung im Rechner.

Zu den Besonderheiten von Geodaten siehe außerdem [Gould et. al. 96], [Anselin 89] und weitere Literatur[1].

1. [Smith et. al. 87] gibt eine gute Einführung in GIS, ihre Geschichte, Komponenten und Anforderungen.
   [Bill & Fritsch 94] gibt einen Überblick und grundlegende Definitionen.
   [Abel 96] führt in die GIS-Thematik ein und berücksichtigt dabei insbesondere Datenbankthemen.
   [Wellar 95] beschreibt, was aus Daten Information macht und grenzt GIS von IS ab.
   [Kemp 90] führt eine Reihe besonderer Eigenschaften räumlicher Daten an.
   [Gradwell 90a] leitet Forderungen an Datenbanken aus den Besonderheiten von Geodaten ab.

3.2 Geoinformationssysteme

Geoinformationssysteme sind spezialisierte Informationssysteme zur Bearbeitung von Geodaten. Ihre Aufgabe ist die Erfassung, Modellierung, Speicherung, Veränderung, Analyse, Wiedergewinnung und Präsentation von Geodaten [Worboys 95]. Im Rahmen dieser Arbeit ist von besonderem Interesse, wie GIS ihre Daten speichern und analysieren. Zur Speicherung existieren heute verschiedene historisch gewachsene Ansätze nebeneinander (s. Abb. 1):

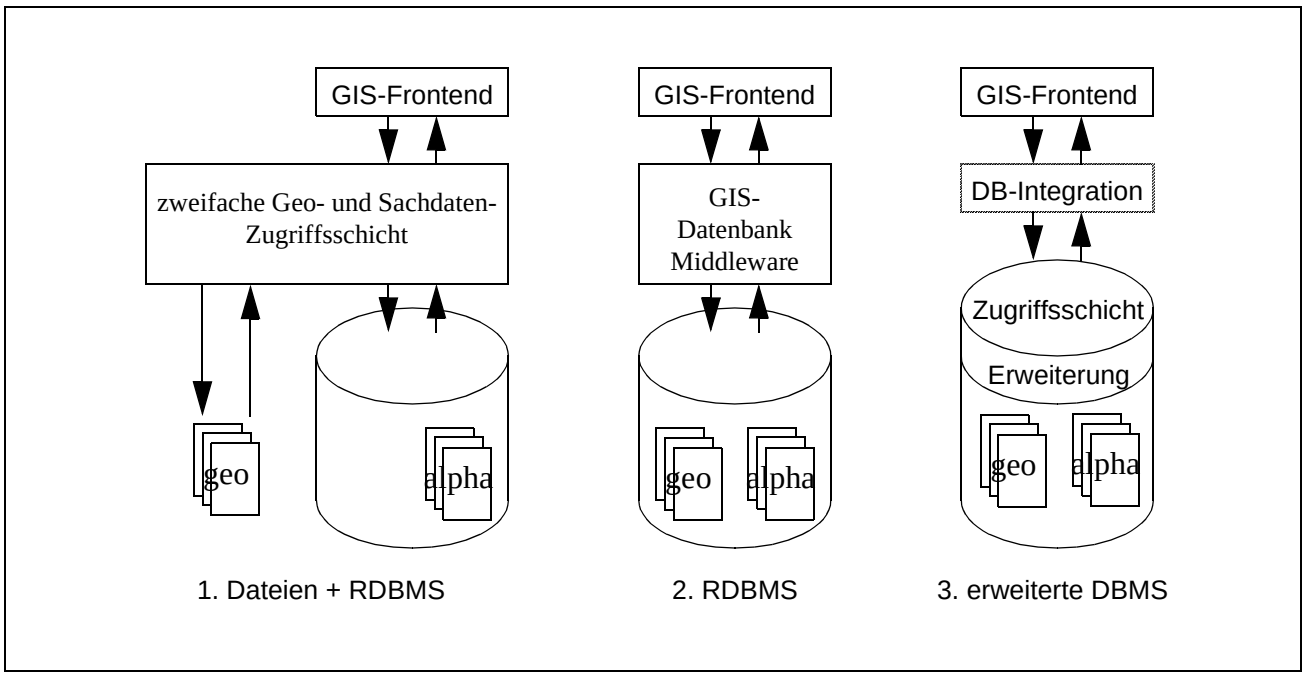


Abb. 1: Architekturen von Geoinformationssystemen

In den Pioniertagen der GIS gibt es keine für die **Speicherung** von GIS-Daten geeigneten Datenbanksysteme. Sie speichern darum ihre Geodaten in proprietären Formaten auf der Basis des Dateisystems. Mit zunehmender Funktionalität und Verbreitung werden die standardisierten RDBMS als Basis für GIS interessant. Kein GIS verzichtet heute darauf, zumindest die Sachdaten in ihnen abzulegen (1). Doch auch für die Geometrie und Topologie gibt es Ansätze auf der Basis der von RDBMS angebotenen Large Objects (Blob), die es ermöglichen ausschließlich auf DBMS aufzusetzen (2). Diese Ansätze münden in erweiterten relationalen und objektorientierten Systemen, die Geodaten nicht nur speichern, sondern auch analysieren können (3).

Aus den verschiedenen Architekturen folgen auch verschiedene Möglichkeiten zur **Analyse** von Geodaten. Beim ersten Ansatz kann eine Analyse nur im Frontend erfolgen. Bei dem zweiten Ansatz können Sachdaten schon in der Datenbank bearbeitet werden. Erst im dritten Ansatz können aber auch Geodaten innerhalb des DBMS analysiert werden.

3.3 Datenbanksysteme

Moderne Geoinformationssysteme speichern die Geodaten unter Verwendung eines Datenbanksystems (DBS). Ein DBS besteht aus einem Datenbankmanagementsystem (DBMS) und der Datenbank. Die Datenbank ist die Sammlung der für das GIS relevanten Daten. Verschiedene GIS-Architekturen setzen unterschiedliche DBMS ein:

1. Proprietäre Datenbankmanagementsysteme
   Proprietäre DBMS werden für eine ganz bestimmte Applikation entwickelt, auf deren Bedürfnisse sie zugeschnitten sind. Dies ist bei frühen GIS-Systemen (SICAD/BS2000 GDB, INFO von ARC/INFO) die Regel und ist auch bei einigen heute noch der Fall. Erst mit dem Aufkommen leistungsfähigerer DBMS verlagerte sich auch die Speicherung der Geodaten in Standarddatenbanken.

2. Relationale Datenbankmanagementsysteme
   RDBMS sind zusammen mit der Anfragesprache SQL heute der Standard im Bereich der Verwaltung von Unternehmensdaten. Alle gängigen GIS bieten die Möglichkeit der Verknüpfung von Geodaten mit im RDBMS gespeicherten Sachdaten über einen dem Geo-Objekt zugeordneten eindeutigen Schlüssel an. Allerdings legen bisher nur wenige GIS die Geometriedaten in RDBMS ab. Denn für die komplexen geographischen Daten reichen die Modellierungsmöglichkeiten des Relationenmodells nicht aus [Frank 84a] [Härder & Reuter 85]. Geometrie wird darum weiterhin in einem proprietären Format, in sogenannten Binary Large Objects (Blob), gespeichert. Die fehlende Zugriffsfunktionalität muß in einer Zwischenschicht (Middleware) implementiert werden.

3. Objektorientierte Datenbankmanagementsysteme
   OODBMS besitzen im Bereich der Modellierung komplexer Daten Vorteile gegenüber RDBMS [Egenhofer & Frank 89b]. Doch trotz der großen in sie gelegten Erwartungen konnten sie ihre Nischenstellung gemessen am Marktanteil nicht verlassen. Dies kann an der Anforderung der Integration mit anderen in RDMBS gehaltenen Unternehmensdaten liegen.

4. Objektrelationale Datenbankmanagementsysteme
   Aus der Kritik an den Mängeln der RDBMS und der Begeisterung über die Möglichkeiten der OODBMS ergeben sich eine Reihe von Versuchen, RDBMS um objektorientierte Konzepte zu erweitern. Beispiele sind Exodus [Carey et. al. 86a] [Carey et. al. 86b] [Günther 87] [Carey et. al. 88] [Carey et. al. 88], Starburst [Schwarz et. al. 86] [Hasan & Pirahesh 88] [Chang & Schek 89] [Haas et. al. 89] [Haas et. al. 90] und Postgres [Stonebraker & Rowe 86] [Rowe & Stonebraker 87] [Stonebraker et. al. 90], letzteres dient auch als Basis für Versuche zur Speicherung von Geodaten [Oosterom & Vijlbrief 91] [Hoop & Oosterom 92].

Aus diesen Ansätzen gehen die objektrelationalen DBMS (ORDBMS) hervor. Beispiele sind hier das aus Postgres abgeleitete Illustra, die Universal Server von Informix und Oracle oder IBM DB2.

3.4 Integration von Datenbanksystemen in Geoinformationssysteme

Bei den dargestellten Datenbankmanagementsystemen stellt sich die Frage nach der Integration in das Gesamtsystem. Die proprietären Systeme sind mit dem GIS zu einem Programmpaket zusammengebunden, zwischen GIS- und DBS-Funktionalität wird nicht explizit unterschieden, beide sind als Funktionen im selben Programm realisiert. Bei den anderen drei Fällen dagegen baut das GIS auf einer externen Datenbank auf, deren Funktionalität über Schnittstellen genutzt werden kann.

Es gibt verschiedene Schnittstellen zu DBS, die jeweils ihre Vor- und Nachteile haben:

1. Schnittstelle für den Anwendungsentwickler (Application Programming Interface, API)
   Über das API kann der Anwendungsentwickler Funktionen aus einer Programmbibliothek (library) aufrufen, welche die Funktionalität des DBMS nach außen zur Verfügung stellen. Diese Funktionen sind auf einer systemnahen Ebene angesiedelt. Sie lassen damit einerseits dem Anwendungsprogrammierer weitreichende Möglichkeiten in das System einzugreifen und bieten gute Performance. Andererseits erfordern sie aber auch eine genaue Kenntnis des Systems und der internen Organisation der Daten. Für jede Abfrage ist die Entwicklung eines Programmes notwendig, in dem prozedural die einzelnen Schritte zur Lösung beschrieben sind. Eine einmal erstellte Lösung muß für Änderungen neu übersetzt werden.
2. Anfragesprachen
   Über eine Anfragesprache kann der Anwender ad-hoc Anfragen formulieren und vom System bearbeiten lassen, ohne dazu programmieren zu müssen. Die Anfragen werden deklarativ beschrieben, das System wählt einen von mehreren Lösungswegen, führt diesen aus und liefert das Ergebnis zurück. Die Funktionen der Sprache sind auf der Ebene des Datenmodells angesiedelt und anwendernah. Dadurch ist nur die Kenntnis der Modellierung, aber nicht der internen Bearbeitung der Daten nötig. Die Performance ist von der Leistungsfähigkeit des Optimierers und der Anfrageverarbeitung der Datenbank abhängig. Anfragesprachen können aber auch in prozedurale Programme eingebunden werden.

Von den genannten Schnittstellen bieten nur Anfragesprachen die Möglichkeit, sowohl in direkter Interaktion mit dem System Anfragen einzugeben, als auch Anfragen fest in Programme zu integrieren. Auch aufgrund ihrer Flexibilität und Einfachheit nehmen sie eine führende Stellung bei der Integration von DBMS in Applikationen ein.

4. Geographische Anfragesprachen

Viele Faktoren beeinflussen die Entwicklung einer geographischen Anfragesprache und ihre Integration in ein Gesamtsystem aus GIS und Datenbanksystem. Dazu gehören die Ergebnisse der Forschung, die Anforderungen der Anwender und die Entwicklungsziele der GIS-Hersteller. Diese drei Bereiche und ihre Beziehungen zueinander zeigt Abbildung 2.

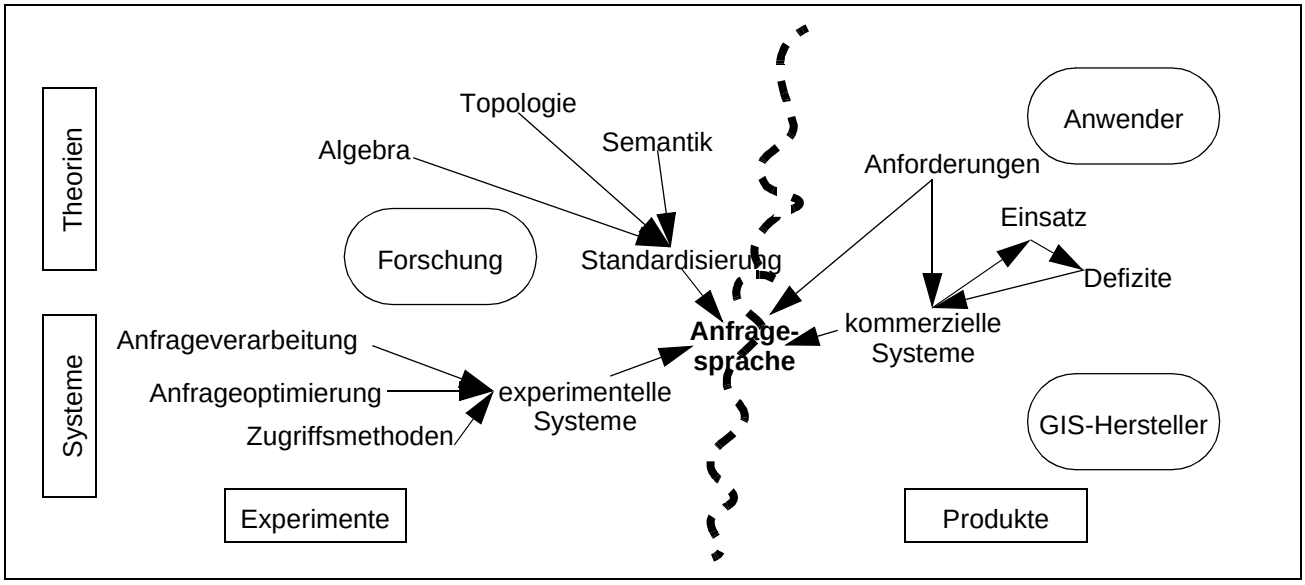


Abb. 2: Entwicklung einer Anfragesprachen für GIS im Umfeld der Einflüsse aus Forschung, Anwendung und Industrie

Auf der einen Seite befindet sich die **Forschung**. Sie stellt die Grundlagen für die Entwicklung einer Anfragesprache bereit (4.1). Dazu gehören die theoretischen GIS-Grundlagen wie die Definition einer Algebra und die Beschreibung topologischer Beziehungen zwischen Geo-Objekten. Sie definiert die Semantik der Sprache und fließt mit ihren Ergebnissen in die aktuelle Standardisierung ein. Weiterhin gehören dazu die DBMS-Grundlagen für die Abarbeitung geographischer Anfragen (4.2), wie Anfrageverarbeitung (4.2.1), Anfrageoptimierung (4.2.2) und räumliche Zugriffsmethoden (4.2.3). Die Fortschritte in diesen Bereichen dokumentieren Beispiele existierender Systeme (4.3), wie z. B. eine Reihe von experimentellen Systemen (4.3.1).

Auf der anderen Seite befinden sich die **Anwender** und die **Hersteller** von Geoinformationssystemen. Die Anwender verwenden kommerzielle GIS (4.3.2), welche die GIS-Hersteller nach ihren Anforderungen erstellen. Im täglichen Einsatz auftretende Defizite fließen als Anforderungen in die Produkte zurück und lassen so spezialisierte Lösungen entstehen, häufig ohne die Ergebnisse aus der Forschung aufzugreifen.

Eine in ein kommerzielles GIS integrierte Anfragesprache muß die Grenze zwischen diesen beiden Bereichen überwinden und ihren Anforderungen gleichermaßen genügen: Sie greift auf die in der Standardisierung gebündelten theoretischen Anforderungen zurück, führt die

Erfahrungen mit experimentellen Systemen in kommerzielle Produkte über und stellt ein für die Anwender sinnvoll nutzbares Werkzeug zur Verfügung.

4.1 Grundlagen und Entwicklungen

Eine **Anfragesprache**[1] (Query Language, QL) ist eine High-Level Computersprache zur Abfrage von Daten in einem Datenbanksystem [Samet 81]. Sie erlaubt interaktiv und im direkten Zugriff (on-line) nicht vorgefertigte (ad-hoc) Anfragen einzugeben. [Samet 81] unterscheidet Anfragesprachen von anderen Systemen anhand den in Tabelle 2 gezeigten Eigenschaften:

Eigenschaft	Anfragesprachen	andere Systeme
Einsatzbedingungen	on-line, nicht vordefiniert, ad-hoc	Batch, vordefiniert, oft wiederholt
Art der Nutzer	unerfahren mit Datenbanken	Spezialisten oder Applikationsentwickler
Art der Sprache	deklarativ, Nutzer geben an, **was** sie wollen, häufig in einem Befehl	prozedural, Nutzer geben an, **wie** die Aufgabe ausgeführt werden soll, häufig in großen Programmpaketen
Dateneingabe und -veränderung	eingeschränkt	unbeschränkt
Komplexität von Berechnungen, Zugriff und Ausgabeformaten	eingeschränkt	nur durch die Implementierung eingeschränkt
Menge von gleichzeitig darstellbaren Daten	eingeschränkt auf wenige Zeilen oder Datensätze	große Mengen von Daten können ausgegeben werden
Performance	Antwort- und Eingabezeit sind wichtiger als die Effizienz zur Laufzeit	Effizienz der Laufzeit ist wichtig

Tab. 2: Charakteristische Eigenschaften von Anfragesprachen nach [Samet 81]

Anfragesprachen kann man weiterhin nach der Art der Interaktion mit dem Benutzer, nach der Art der Operationen oder nach der Art der bearbeiteten Daten unterscheiden:

1. Nach der **Art der Interaktion** versteht man unter Anfragesprachen im weiteren Sinne auch mit einem Textmenü auswahlbasierte oder graphische Systeme, und solche, die durch Eingabe mathematischer Formeln oder mit natürlicher Sprache bedient werden. Anfragesprachen im engeren Sinne sind textuell, d. h. sie verfügen ähnlich einer Programmiersprache über einen begrenzten Wortschatz an Schlüsselworten, die nach bestimmten syntaktischen Regeln zu einer Anfrage zusammengesetzt werden können [Vassiliou & Jarke 84] [Jarke & Vassiliou 85].

1. Die Begriffe Anfragesprache und Abfragesprache werden gleichbedeutend verwendet.

Graphische Anfragesprachen ermöglichen dem Anwender, eine Anfrage durch eine Zeichnung auszudrücken, die dann vom System ausgewertet und in eine systemspezifische Form übersetzt wird.

Ein Beispiel einer graphischen Anfragesprache ist **CIGALES** (Cartographical Interface Generating an Adapted Language for Extensible Systems) [Mainguenaud & Portier 90] [Calcinelli & Mainguenaud 91]. CIGALES ist ein graphisches auf Icons basierendes Anfragesystem, eine Art graphisches Query-by-Example (QBE) [Zloof 75][2]. Es setzt auf dem erweiterten RDBMS Sabrina [Gardarin et. al. 87] auf. Es erlaubt die graphische Formulierung von GIS-Anfragen, die dann in eine für das zu Grunde liegende System geeignete Darstellung übersetzt werden. Die Sprache dient der direkten und nutzerfreundlichen Kommunikation mit dem Endanwender. Sie wurde nicht für die Integration in ein prozedurales Gesamtsystem geschaffen und führt keine globale Optimierung durch, was in schwacher Performance resultiert. Ein weiteres Beispiel einer graphischen Anfragesprache ist **Sketch!** [Meyer 92]. Eine Übersicht früherer Sprachen in Bilddatenbanken wie z. B. Query-by-Pictorial-Example (QPE) gibt [Chang & Fu 81], für eine neuere Entwicklung siehe [Lee & Chin 95].

Ein Beispiel für **textuelle Anfragesprachen** ist die Structured Query Language[3] (SQL) der heute vorherrschenden RDBMS.

2. Nach der **Art der Operationen** kann man zwischen Anfragesprachen unterscheiden, die sich nur auf die eigentliche Abfrage von Daten konzentrieren oder die auch die Veränderung (Data Manipulation Language, DML) und Definition der Daten (Data Definition Language, DDL) erlauben. Forschungsarbeiten konzentrieren sich häufig nur auf die Abfrage, während kommerzielle Sprachen wie SQL den vollen Funktionsumfang unterstützen.

3. Nach der **Art der Daten** kann man Anfragesprachen unterscheiden, die sich auf die Abfrage bestimmter Daten spezialisiert haben. Häufig werden sie durch speziell angepaßte Datenstrukturen unterstützt.

   Graph-Anfragesprachen erlauben Anfragen auf Graphen [Amann & Scholl 92]. Die Graphen bestehen aus durch Kanten verbundenen Knoten. Graphen finden in GIS z. B. für die Verwaltung von Netzwerken von Straßen oder Versorgungsleitungen Anwendung. Beispiele solcher Sprachen sind **G** [Cruz et. al. 87] und **GraphDB** [Güting 94a]. Obwohl einige GIS-Probleme mit Graphen gelöst werden können, decken sie nur einen Teilbereich ab. Sie reichen deshalb nicht als alleinige geographische Abfragesprache aus.

---

2. Weitere Beispiele QBE-basierter Anfragesprachen sind Query-by-Pictorial Example (QPE) [Chang & Fu 80], GEOBASE [Barrera & Buchmann 81] [Radhakrishan et. al. 81] oder PICQUERY [Joseph & Cardenas 88]. Auch Hypermaps [Boursier & Mainguenaud 92] lassen sich hier einordnen.
3. Auch Standard Query Language. Über die Entwicklung von SQL siehe [Vossen 94] und [Worboys 95].

Eine **Räumliche Anfragesprache** (Spatial Query Language, SpQL) ermöglicht Anfragen auf Daten mit einem räumlichen Bezug. Typische Anwendungsbereiche sind CAD-CAM, VLSI-Design, Robotersteuerung, historische Datenbanken, Architektur, räumliche Wahrnehmung und autonome Navigation, Wegverfolgung, Umweltschutz, medizinische Bilddaten und Geoinformationssysteme. Die Sprachen weisen Gemeinsamkeiten in den zu bearbeitenden geometrischen Problemen auf, es gibt aber auch Unterschiede, die sich aus den Besonderheiten der jeweiligen Daten und ihrer Anwendung herleiten. Zu den Besonderheiten von Geodaten siehe Kapitel 3.1.

Eine **Geographische Anfragesprache** (Geographical Query Language, GQL) ist eine Spezialisierung räumlicher Anfragesprachen für geographische Daten. Unterschiede liegen z. B. in der Größe des behandelten Gebiets (ein Mensch kann sich darin bewegen (Umwelt, Natur) oder kann es in der Hand halten (Chip, Bauteil)), in der Art der behandelten Daten (künstliche und beliebig genaue Daten bei CAD und generalisierte und aus Meßwerten der realen Welt abstrahierte Daten bei GIS) oder in der Art der anzuwendenden Funktionen (Herausfiltern von ganzen Bildern mit bestimmten Eigenschaften in der Medizin, Herausfinden von Elementen mit bestimmten Eigenschaften aus einer Karte mit GIS).

Eine geographische Anfragesprache, die sich von SQL ableitet, läßt sich, wie SQL auch, unter textuellen Sprachen (Punkt 1) einordnen. Die geographische Anfragesprache erweitert in diesem Fall SQL um Operatoren und Funktionen, die ihr zur Bearbeitung räumlicher Daten fehlen.

Die drei vorgestellten Einteilungen sind nicht notwendigerweise getrennt, sondern können sich auch überschneiden. Graphische Anfragesprachen sind z. B. im Bereich der Geoinformationssysteme ein vielversprechender Ansatz, die Schnittstelle zwischen Nutzer und System zu verbessern. Als Schnittstelle zwischen Systemen, z. B. zwischen dem Geoinformationssystem und dem Datenbanksystem, kommen aber weiterhin textuelle Sprachen zum Einsatz. Sie sind sowohl für Applikationsentwickler als auch für nicht völlig unerfahrene Nutzer geeignet.

Gemäß dieser Einteilungen beschäftigt sich die vorliegende Arbeit mit der Integration von GIS und DBMS durch eine textuelle Anfragesprache, die sich an den aktuellen Sprachstandard SQL anlehnt. Sie ist rein auf Anfragen ausgerichtet und spezialisiert sich auf geographische Anfragen als einen Teilbereich räumlicher Anfragen.

Die für diesen Teilbereich wichtigen theoretischen Grundlagen (4.1.1) stellt der folgende Abschnitt vor.

4.1.1 Theoretische Grundlagen

Die theoretischen Grundlagen bilden das Fundament der Anfragesprache. Sie legen sowohl die Bedeutung der einzelnen Operatoren der Sprache fest, als auch, wie sie sich in den Gesamtansatz einfügen. Für die Entwicklung einer räumlichen Anfragesprache von Bedeutung sind die Erweiterung der Relationenalgebra mit räumlichen Algebren, die Topologie und das Dimensionally Extended 9 Intersection Model. Auf sie wird in diesem Abschnitt kurz eingegangen.

Erweiterungen der Relationenalgebra

Die **Relationenalgebra** ist eine Menge von Operatoren, die Relationen bearbeiten [Codd 72]. Zu den Operatoren gehören Mengenoperatoren (Vereinigung, Differenz, Durchschnitt, symmetrische Differenz) und spezielle Operatoren der Relationenalgebra (Restriktion, Projektion, Verbund, Division). Sie kann für eine lineare Interndarstellung einer Anfrage verwendet werden. Ein Ausdruck der Relationenalgebra beschreibt einen Algorithmus zur Konstruktion der Ergebnisrelation [Mitschang 95]. Die Algebra bildet das Fundament der Sprache.

Systeme räumlicher Datentypen, genannt **räumliche Algebra**, erfassen die fundamentalen Abstraktionen für Punkte, Linien und Flächen, die Beziehungen zwischen ihnen und die Operationen auf ihnen [Güting 94c]. Ein Beispiel ist hier die ROSE-Algebra [Güting & Schneider 93a] [Güting & Schneider 93b] [Güting & Schneider 95] [Güting et. al. 95].

Um eine Basis für die Bearbeitung von räumlichen Objekten in DBMS zu schaffen, ist es notwendig relationale und räumliche Algebra zu integrieren und so ein gemeinsames Modell der Bearbeitung von Geometrien in der Datenbank zu schaffen. Ansätze dafür sind:

- Die **Geo-Relational Algebra** (GRAL) [Güting 88a] [Güting 88b] [Güting 88c] z. B. erweitert die relationale Algebra um geometrische Datentypen und Operatoren. Die Implementierung eines Prototyps ist unter [Güting 89] beschrieben.
- In der Fortsetzung seiner Arbeit definieren [Scholl & Voisard 89] theoretisch eine Sprache zur Manipulation von Karten, die auf der Complex Object Algebra von [Abiteboul & Beeri 88] beruht. Als Basis einer Implementierung könnte ein OODBMS dienen, worauf aber nicht näher eingegangen wird.
- Laurini erweitert die relationale Algebra zur **Peano Tupel Algebra** [Laurini 87] [Laurini & Milleret 87] und untersucht ihren Einsatz in Kombination mit Computational Geometry Algorithmen [Laurini & Milleret 89].
- [Worboys et. al. 90a] [Worboys et. al. 90b] beschreiben verschiedene Ansätze, welche Operatoren eine Sprache enthalten sollte und definieren eine formale Sprache.

Topologie

Zwischen Geo-Objekten bestehen eine Vielzahl von topologischen Beziehungen. Ihre Geometrien können sich z. B. überlappen, eines im anderen enthalten sein oder sie berühren sich am Rand. Die Umgangssprache bezeichnet solche Beziehungen häufig ungenau, mehrere Bezeichnungen überschneiden sich in ihrer Bedeutung oder können nur unscharf voneinander unterschieden werden [Shariff et. al. 98]. Das Problem für die Definition einer Anfragesprache besteht darin, welche Namen die Operatoren erhalten sollen und welche Bedeutung ihnen zugewiesen wird.

Kommuniziert der Endnutzer über die Anfragesprache direkt mit dem System, so kann eine ungenaue Definition noch toleriert werden oder ist manchmal sogar hilfreich[4], da so keine Ergebnisse verworfen werden, die vielleicht doch gemeint waren. Der Nutzer kann dann interaktiv sein gewünschtes Ergebnis näher beschreiben. Soll die Anfragesprache aber der Integration von GIS und DBMS dienen, so ist eine exakte Definition der Bedeutung der Operatoren unerläßlich.

Diesem Zweck dient die genaue mathematische Beschreibung topologischer Beziehungen. Seit diese von [Abler 87] gefordert wurde, hat sie sich in großen Schritten weiterentwickelt. Ausgehend vom 4-Intersection-Model, das für die Bestimmung der Beziehungen zweier Flächen zueinander [Egenhofer & Herring 90] [Egenhofer & Frank 91] die Schnittmengen aus jeweils deren Innerem und deren Rand berücksichtigt, erarbeiteten [Egenhofer & Herring 91] das 9-Intersection-Model (9IM), das zusätzlich das Äußere eines Objektes mit einbezieht. Das 9IM war Gegenstand intensiver Untersuchungen[5] und berücksichtigt als Dimensionally Extended 9-Intersection-Model (DE-9IM) [Franzosa & Egenhofer 92] auch die Dimension der Schnittmengen. Es bildet in der Form, wie es in die Standardisierung eingegangen ist [OGC SF SQL 98], die Grundlage zur Definition der in Kapitel 5 vorgestellten Anfragesprache und wird im folgenden näher erläutert. Einen sehr guten Überblick über die Entwicklung dieses Gebiets gibt [Egenhofer 97].

4. Stehen keine scharf definierten Objekte zur Verfügung, können auch unscharfe Operatoren sinnvoll sein [Clementini & Di Felice 97] [Erwig & Schneider 97]. Solche Objekte werden im Rahmen dieser Arbeit nicht betrachtet. Darum kann das 9-Intersection-Model eingesetzt werden.

5. [Egenhofer & Mark 95] ermittelten die 19 möglichen topologischen Beziehungen zwischen Flächen und Linien, [Egenhofer 93] die 33 möglichen zwischen zwei Linien im R^2. [Paiva & Egenhofer 95] und [Egenhofer et. al. 94] erweitern das Modell für Flächen mit Löchern, [Clementini et. al. 95] für zusammengesetzte Objekte.

Dimensionally Extended 9 Intersection Model (DE-9IM)

Das DE-9IM beschreibt die Beziehung zwischen zwei räumlichen Objekten. Jedes Objekt unterteilt den 2-dimensionalen Raum in die drei Teile Inneres (interior), Rand (boundary) und Äußeres (exterior). Jede Beziehung zwischen zwei Objekten im Raum kann als das Ergebnis der neun möglichen Schnittmengen zwischen dem Inneren, dem Rand und dem Äußeren dieser beiden Objekte beschrieben werden (siehe Tabelle 3). Die Einträge x_{ab} der Matrix können dabei die Werte T, F, *, 0 oder 1 annehmen, deren Bedeutung Tabelle 4 entnommen werden kann. Ein Operator ist genau dann erfüllt, wenn alle Teilbedingungen seiner entsprechenden Matrix x_{ab} erfüllt sind.

x_{ab}		Element B		
		Inneres	Rand	Äußeres
Element A	Inneres	x_{ii}	x_{ir}	$x_{iä}$
	Rand	x_{ri}	x_{rr}	$x_{rä}$
	Äußeres	$x_{äi}$	$x_{är}$	$x_{ää}$

Tab. 3: Dimensionally Extended 9 Intersection Model (DE 9-IM)

Symbol	Bedeutung	Erklärung
T	Wahr (true)	die Schnittmenge ist nicht leer, aber von beliebiger Dimension, d. h. kann aus Punkten, Linien oder Flächen bestehen
F	Falsch (false)	die Schnittmenge ist leer
*	beliebig	die Schnittmenge ist für das Ergebnis des Operators nicht von Bedeutung
0	0-dimensional	die maximale Dimension der Objekte in der Schnittmenge ist 0, d. h. sie enthält einen oder mehrere Punkte
1	1-dimensional	die maximale Dimension der Objekte in der Schnittmenge ist 1, d. h. sie enthält eine oder mehrere Linien, auch Punkte sind erlaubt

Tab. 4: Bedeutung der Symbole des DE-9IM

Zusammenführung von Theorie und Praxis

[Paredaens et. al. 94] regen Untersuchungen zu einem allgemeinen Modell für räumliche Datenbanken an, welches das relationale Modell erweitert. Das Modell soll die häufig nur auf einen Spezialfall ausgerichteten Systeme unter eine gemeinsame Theorie räumlicher Datenbankanfragen bringen. Paredaens versucht theoretische Modelle und praktische Implementierungsgesichtspunkte zu vereinen, indem er eine allgemeine räumliche Theorie aufstellt verschiedene Modelle damit bewertet [Paredaens & Kuijpers 98].

4.1.2 Eigenständige Sprachen

Für eigenständig entwickelte Sprachen, die keinen Bezug zu einer schon existierenden Sprache nehmen, gibt es wenige Beispiele. Eines ist die Abfragesprache für raumbezogene Da-

ten (Spatial Data Query Language) **SDQL**, eine eigenständig entwickelte Sprache zur Unterstützung der Bilanzierung in der Flurbereinigung von [Findeisen 90]. Weiterhin kann man **QPF** [Wu et. al. 89] dazu rechnen. Es basiert auf dem wissensbasierten GIS KGIS [Wu 88] und besteht aus drei Ebenen: Eine Datenmanipulationssprache stellt die grundlegenden Prozeduren zur Verfügung. Darauf bauen eine formular- und eine icon-basierte Anfragesprache auf. Weitere Beispiele für die Definition formaler Sprachen für räumliche Analyse finden sich nach [Lee & Chin 95] bei [Mohan & Kashyap 88], [Menon & Smith 89] und [Kasturi et. al. 89].

Die überwiegende Mehrzahl räumlicher Anfragesprache erweitert aber eine gängige Anfragesprache um räumliche Operatoren und lehnt sich dabei an deren Syntax an. Damit beschäftigt sich der folgende Abschnitt.

4.1.3 Erweiterungen existierender Datenbanksprachen

Zahlreiche Ansätze erweitern existierende Anfragesprachen. Mit der zunehmenden Verbreitung von RDBMS und der Durchsetzung von SQL als Standard für die Abfrage von Daten konzentrieren sie sich auf SQL[6].

Eine frühe, richtungsweisende Arbeit ist die Sprache **MAPQUERY** von [Frank 82a]. Sie erweitert den Vorläufer von SQL, die Structured English Query Language SEQUEL von [Chamberlin et. al. 76]. Nach [Frank 82a] unterscheiden sich Geo-Objekte durch ihre Geometrie und Lage im Raum von herkömmlichen kommerziellen Daten. Darum enthält MAPQUERY Sprachelemente, die beschreiben, wie das Ergebnis einer Abfrage graphisch dargestellt werden soll. WINDOW wählt einen Ausschnitt der darzustellenden Karte. THE(area) wählt einen Kontext, der als Hintergrundinformation eine Einordnung der gefundenen Ergebnisse ermöglicht. SHOW … AS … ordnet einzelnen Objektarten verschieden Darstellungsstile zu. PICK fordert den Anwender auf, eine Koordinate einzugeben, und erlaubt so auch graphische Interaktion.

Im Gegensatz dazu modelliert GEOVIEW [Waugh & Healey 87] Geo-Objekte relational. Es speichert ihre Attribute in Tabellen und legt die Geometrie als binäre Zeichenkette (BLOB oder LONG) ab, wo sie nicht für die Anfragesprache auswertbar sind. GEOVIEW verwendet reines SQL als Anfragesprache. Allein die Modellierung eines das Geo-Objekt umgebenden Rechtecks (Bounding Box) durch die Koordinaten zweier Eckpunkte (xmin, ymin, xmax, ymax) erlaubt mit SQL eingeschränkte räumliche Anfragen auszudrücken. Dieser rein auf SQL basierende Ansatz hat sich nicht durchgesetzt.

6. Eine Ausnahme bildet **GeoSAL** [Huang93] [Svensson & Huang91], das die Mengenalgebra-Sprache SAL [Arnborg80] um räumliche Datentypen und Operatoren erweitert. [Huang & Svensson93] und [Huang et. al.92] stellen Beispiele vor, wie sich damit GIS-Analysen durchführen lassen. Die Definition einer Schnittstelle zur Integration in prozedural orientierte Systeme und die Optimierung der Anfragen bleiben aber ungelöst.

[Egenhofer 87] schlägt vor, SQL um Operatoren zu erweitern, mit denen topologische und metrische Beziehungen zwischen Geo-Objekten als Bedingungen formuliert werden können, wie z. B. MEETS, OVERLAP oder INTERSECTS (siehe Kapitel 4.1.1). Auch **GEOQL** [Ooi & Sacks-Davis 89], **KGIS** [Ingram & Phillips 87] und **TIGRIS** [Herring et. al. 88] erweitern SQL um topologische Operatoren. Weiterhin erörtert [Egenhofer 90], welche Eigenschaften die Benutzerschnittstelle zu räumlichen Daten haben sollte und fordert wie schon [Frank 82a], Ergebnisse graphisch anzuzeigen. Auch die Konzepte des Kontext und der graphischen Interaktion finden sich bei ihm wieder.

[Egenhofer 94] führt die Ergebnisse dieser Arbeiten zusammen: **Spatial SQL** bettet topologische und metrische Abfragen in SQL ein. Die **Graphical Presentation Language GPL** erlaubt mit Kommandos wie SET LEGEND, SET WINDOW, SET CONTEXT und SET MODE, eine Legende darzustellen, einen Kartenausschnitt und Hintergrund zu wählen, sowie die Darstellung der mit Spatial SQL gewählten Geo-Objekte zu beeinflussen. SQL eignet sich nach [Egenhofer 92] nicht zur Beschreibung der graphischen Präsentation des Ergebnisses. Darum trennt [Egenhofer 94] die Selektion der Daten von ihrer Darstellung in zwei Sprachen. Diese Trennung setzt sich später in der Standardisierung fort, die den topologischen Teil integriert, die Darstellung aber den aufsetzenden Systemen überläßt.

4.1.4 Standardisierung räumlicher Anfragesprachen

Drei internationale Gremien beeinflussen im wesentlichen die Standardisierung von Anfragesprachen als Schnittstelle zu geographischen Informationssystemen: erstens das Technical Committee Geographic Information/Geomatics (ISO TC 211) der International Standards Organisation (ISO), zweitens das gemeinsame Technical Committee Information Technology (ISO/IEC JTC 1) der International Standards Organisation und der International Electrotechnical Commission (IEC) und drittens das OpenGIS Consortium (OGC). Tabelle 5 stellt sie ihren europäischen und deutschen Partnergremien gegenüber.

Ebene	GIS	DB
international	International Organization for Standardization (ISO)	
	ISO TC 211 Geographic Information/Geomatics Standardfamilie 15046-x	ISO/IEC JTC 1 SC 32 WG 4 Int. Electrotechnical Commission JTC 1 Information technology SC 32 Data Management and Interchange WG 3 Database Language (9075) WG 4 SQL/Multimedia and Application Pakkages (13249)
	Open GIS Consortium (OGC)	
	Simple Features for CORBA, OLE/COM und SQL	
europäisch	Comité Européen de Normalisation (CEN)	
	CEN TC 287 Geographic Information	CEN BTS 7 Information Technology
deutsch	Deutsches Institut für Normung (DIN)	
	DIN NABau-3.3 NA Bauwesen (NABau) FB 03 Vermessungswesen, Geoinformation 03.03.00 Kartographie und Geoinformation	DIN NI-32.3 NA Informationstechnik (NI) NI-32 Offene Systeme SQL

Tab. 5: Übersicht der Standardisierungsgremien für geographische Informationen und Anfragesprachen auf internationaler, europäischer und deutscher Ebene

ISO TC 211 Geographic Information/Geomatics beschäftigt sich mit der Standardisierung im Bereich digitaler geographischer Information. Seine Arbeit zielt auf die Einrichtung der Standardfamilie 15046-x für Informationen über Objekte, die direkt oder indirekt mit einer Position auf der Erde verbunden sind [TC 211 Scope 98]. Das Komitee ist untergliedert in 5 Arbeitsgruppen (Working Group WG1 - WG5), die 20 Arbeitsgebiete (Work Item WI1 - WI 20) bearbeiten. Die Standardisierung der Datentypen und Operatoren einer Anfragesprache liegt im Arbeitsfeld der WG 2 *Geospatial Data Models and Operators*. Sie bearbeitet u. a. die Arbeitsgebiete WI 7 *Spatial Schema* und WI 20 *Spatial Operators*.

Das konzeptionelle Schema dieses Teils des Standards beschreibt die räumlichen Eigenschaften (spatial characteristics) eines Geo-Objekts (geographic feature), indem es räumliche Basis-Objekte (primitives) und eine Menge damit konsistenter räumlicher Operatoren (spatial operators) definiert (s. Abb. 3). Zu den räumlichen Eigenschaften gehören Geometrie und Topologie. Die Geometrie beschreibt mit Koordinaten und mathematischen Funktionen quantitativ die Dimension, die Position, die Größe, die Form und die Orientierung im Raum. Die Topologie beschreibt qualitativ die Beziehungen zwischen geographischen Elementen. Der Standard definiert geometrische und topologische Primitive für Punkte, Linien, Flächen und Körper und daraus zusammengesetzte komplexe Objekte. Die räumlichen Operatoren verwenden, befragen, erzeugen, verändern oder löschen die geographischen Elemente.

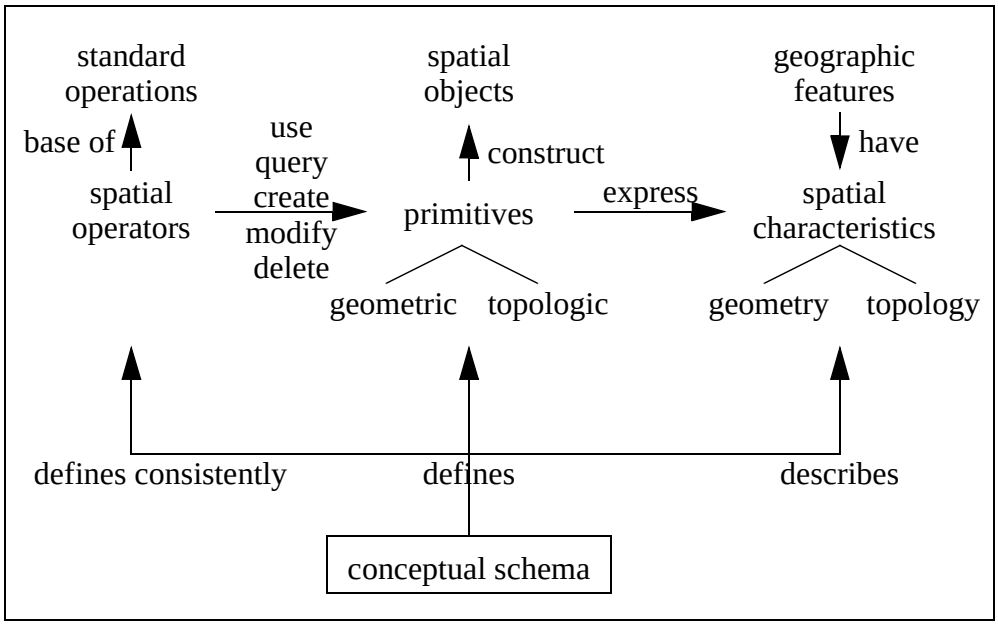


Abb. 3: Zielsetzung des Standards ISO TC 211 15046-7 (graphisch aufbereitet nach [TC 211 N 549]).

Der Operator equals und die Funktion distance werden für alle Geo-Objekte eingeführt. Envelope liefert das kleinste umschließende Rechteck (Minimum Bounding Rectangle, Bounding Box), weiterhin gibt es boundary, buffer, convexHull, und centroid. Für Linien wird length definiert, für Flächen perimeter und area. Der Standard überläßt es dem Anwender auf dieser Basis weitere für die Applikation wichtige Operatoren wie intersect oder cross zu implementieren und verweist auf die Operator-Definitionen von Egenhofer und Clementini.

Im **ISO/IEC JTC 1** beherbergt das Subcommittee SC 32 *Data Management and Interchange* die Working Groups WG 3 *Database Language* (DBL) und WG 4 *SQL/Multimedia and Application Packages* (SQL/MM) [JTC 1 SC 32]. WG 3 DBL arbeitet an der Definition von SQL 3, einem neuen Standard (9075) für die Datenbanksprache SQL. SQL3 erlaubt applikationsspezifische abstrakte Datentypen und Methoden zu definieren. Die Datentypen und Operatoren eines Anwendungsgebietes bilden zusammen ein Applikation Package. Diese standardisiert WG 4 SQL/MM mit der Familie 13249-x. Der Teil 3 Spatial (13249-3) definiert räumli-

che Datentypen und Operatoren, wie sie z. B. GIS-Anwendungen benötigen (s. Abb. 4).

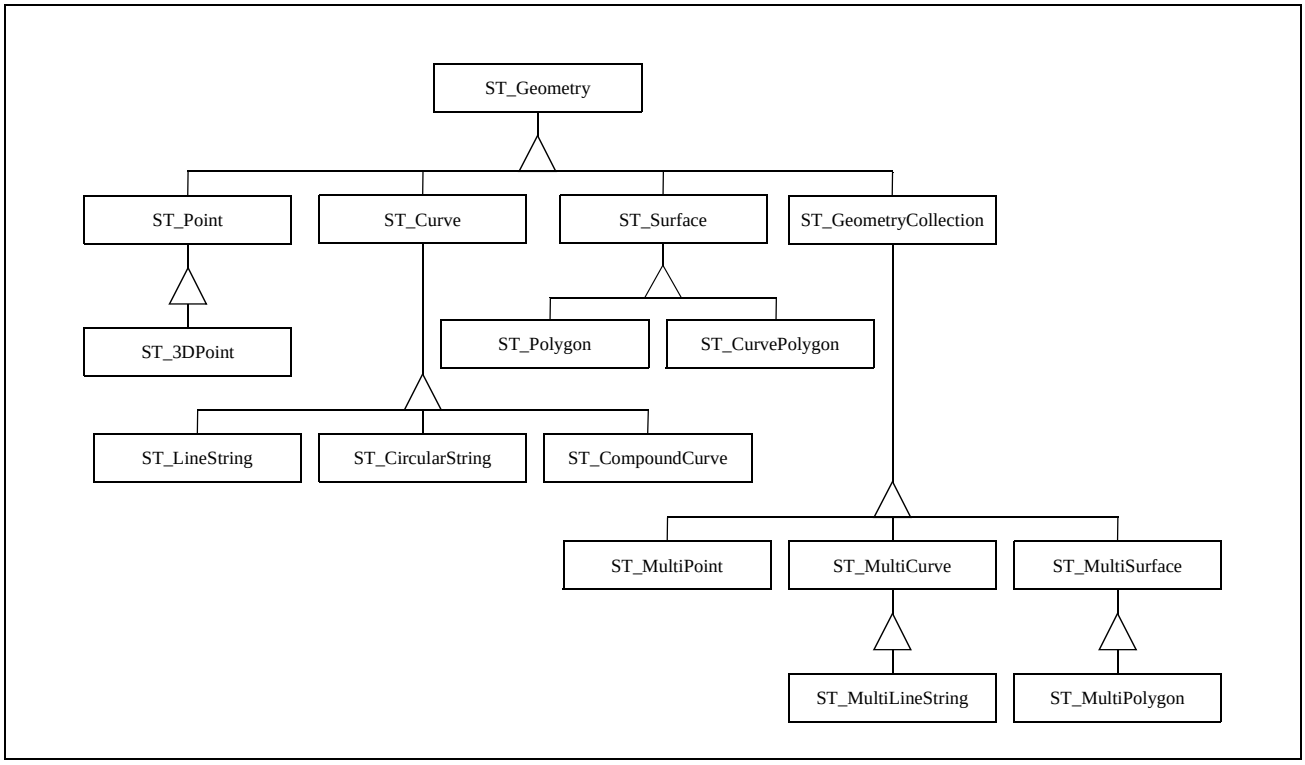


Abb. 4: Hierarchie der räumlichen Datentypen des Standards ISO 13249-3 (nach [SQL/MM FRA-004 98])

Die Hierarchie basiert auf dem Datentyp ST_Geometry. Er faßt die Methoden zusammen, die allen Geometrien zur Verfügung stehen. Dazu gehören die Mengenoperationen Schnitt, Vereinigung, Differenz und symmetrische Differenz. Darauf und auf dem DE-9IM bauen die räumlichen Operatoren ST_Equals, ST_Relate, ST_Disjoint, ST_Intersects, ST_Touch, ST_Cross, ST_Within, ST_Contains und ST_Overlap auf. Sie sind Methoden von ST_Geometry und somit für alle abgeleiteten Datentypen gültig. Auf ST_Geometry gründen sich die Basisdatentypen für 0-dimensionale, 1-dimensionale und 2-dimensionale Objekte: ST_Point, ST_Curve und ST_Surface. Von diesen leiten sich weitere spezialisierte Datentypen für 3-dimensionale Punkte (ST_3DPoint), für Linienzüge (ST_LineString, ST_CircularString, ST_CompoundCurve) und für Flächen (ST_Polygon, ST_CurvePolygon) ab. ST_GeometryCollection faßt verschiedene Objekte vom Typ ST_Geometry zu einem zusammen.

Das **OpenGIS Consortium (OGC)** hat die Vision, Geodaten und deren Verarbeitung voll in die heute übliche Datenverarbeitung zu integrieren und weltweit mit interoperablen, kommerziellen Programmen eine weitverbreitete Nutzung zu ermöglichen [OGC 98]. Auf der Basis einer abstrakten Spezifikation [OGC AS 98] erarbeiten ihre Mitglieder Implementierungsspezifikationen. Die *OpenGIS Simple Features Specifiaction for SQL* [OGC SF SQL 98] legt

grundlegende räumliche Datentypen und deren Methoden fest, die weitgehend mit denen in Abbildung 4 übereinstimmt.

Die enger werdende **Zusammenarbeit** zwischen den drei Standardisierungsgremien [Kottman 98] weckt die Hoffnung auf einen einheitlichen von allen drei Gremien akzeptierten Standard. Einen ersten Ansatz dazu bildet die Harmonisierung zwischen den Datenmodellen von OGC Simple Features und SQL/MM Spatial. Eingeleitet durch einen Vergleich der beiden Modelle [SQL/MM CWB-038] werden die Datentypen in SQL/MM Spatial an die von Simple Features angepaßt. Beide stimmen damit in ihrer Kernfunktionalität überein. Bei komplexeren Geometrien, die in SQL/MM schon definiert sind, stehen die entsprechenden Arbeiten bei OGC noch aus. Das Dimensionally Extended 9-Intersection Model ist als Grundlage für die Definition der Semantik topologischer räumlicher Operatoren allgemein anerkannt und wird in allen drei Standards verwendet.

4.2 Verarbeitung geographischer Anfragen

Die Realisierung einer Anfragesprache erfordert außer der Definition der Semantik der Anfragesprache auch die performante Umsetzung der räumlichen Operatoren. Geeignete Methoden zur Vearbeitung (4.2.1) und Optimierung (4.2.2) räumlicher Anfragen beeinflussen wesentlich die Leistung des Gesamtsystems. Spezialisierte Zugriffsmethoden (4.2.3) beschleunigen den Zugriff nach räumlichen Kriterien.

4.2.1 Anfrageverarbeitung

Anfrageverarbeitung (AV) und Anfrageausführung (Anfrageevaluierung, AE) bilden zusammen die Anfragebearbeitung. Bei der Anfrageverarbeitung geht es darum, die vom Benutzer in einer deklarativen Sprache eingegebene Anfrage so umzuformen, daß ihr Ergebnis vom System mit möglichst geringem Aufwand berechnet werden kann. Dazu gehört, die Anfrage in eine Interndarstellung zu übersetzen, sie zu optimieren und auszuführen [Mitschang 95]. Für traditionelle relationale DBMS existiert inzwischen eine fundierte Theorie, die Einbettung der Operatoren einer räumlichen Anfragesprache aber stellt neue Anforderungen an die Optimierung und Ausführung der Anfrage, zum Beispiel:

- Neue **Operationen** wie z. B. Punkt- und Fensteranfragen oder die Suche nach den nächsten Nachbarn müssen unterstützt werden.
- **Spatial Joins** [Brinkhoff 94] bestimmen für zwei Objektmengen alle Paare, die ein geometrisches Prädikat erfüllen. Verwandt damit ist auch das geometrische Verschneiden (Map Overlay) zweier Mengen von Geo-Objekten.
- **Räumliche Zugriffsstrukturen** sind ein elementarer Bestandteil der internen Bearbeitung einer Anfrage. In Standard-DBMS kommen hier vornehmlich B-Bäume zum Einsatz. Für

Geodaten sind diese jedoch nicht ausreichend. Zahlreiche neue Zugriffsstrukturen wurden entwickelt, wenige wurden bisher in kommerzielle System integriert.

- Viele Aufgaben lassen sich mit Algorithmen aus dem Feld der **Computational Geometry** berechnen. Solche Algorithmen sollten in die Anfrageverarbeitung integriert werden.
- **Approximationen** ermöglichen die näherungsweise Berechnung einer Treffer-Obermenge und helfen so teuere Operationen zu vermeiden. Sie werden erst in einem Verfeinerungsschritt benötigt. Durch dieses **Filter-and-Refine** Verfahren [Bringkhoff et. al. 94a] lassen sich Ergebnisse von GIS-Operationen mehrstufig berechnen.

4.2.2 Anfrageoptimierung

Ziel der Optimierung ist die Generierung eines optimalen Ausführungsplans. Hier fließen als Kriterien z. B. die Kosten der Auswertung verschiedener Operatoren mit ein. Die Kosten bedingen sich auch aus den unterschiedlichen Methoden, die für die Ausführung der Anfrage zur Verfügung stehen. Existiert zum Beispiel ein bestimmter für die Lösung eines Problems optimierter Index, z. B. ein Quadtree für die Fragestellung ob ein Anfragepunkt sich in einem gespeicherten Geo-Objekt befindet, so sollte die Optimierung diesen bei der Auswahl eines geeigneten Plans berücksichtigen.

Die im Rahmen dieser Arbeit unterstützten Optimierungsmethoden sind eine abgewandelte Form von Filter-und-Refine und die Erweiterung der Anfrageverarbeitung nach [Ooi & Sacks-Davis 89]. Letztere erweitern die Anfragebearbeitung eines von [McDonell 86] beschriebenen relationalen DBMS (relational test bed, RTB) und belegen die Einsatzmöglichkeiten am Beispiel der Sprache GEOQL. Die Autoren räumen allerdings ein, daß ihre Methode darauf angewiesen ist, in Interna des DBMS wie z. B. die Pufferverwaltung oder die Speicherung von Zwischenergebnissen eingreifen zu müssen, da eine Ausführung der optimierten Teilabfragen auf der Ebene eines normalen Datenbanknutzers und die Zwischenspeicherung in temporären Tabellen zu ineffizient ist.

4.2.3 Methoden zum Zugriff auf räumliche Daten

Räumliche Zugriffsmethoden oder Indizes (spatial access method, spatial index) unterstützen die Selektion nach räumlichen Kriterien. Ein räumlicher Index unterteilt den Raum und die sich darin befindenden Objekte, so daß nur eine Teilmenge davon bei Anfragen berücksichtigt werden muß [Güting 94c].

Es gibt eine Vielzahl von räumlichen Zugriffsmethoden. Für einen Überblick siehe [Gaede 96] [Gaede & Günther 95] [Widmayer 91]. Im folgenden werden die im Rahmen dieser Arbeit zur Anwendung kommenden Zugriffsmethoden kurz erläutert.

Der **Quadtree** ist ein mehrstufiger Baum, der auf der rekursiven Zerlegung einer Fläche in vier Teile beruht, bis die entstehenden Teilflächen nur noch wenige Objekte enthalten oder

bei Raster-Quadtrees homogene Teilflächen sind. Es gibt zahlreiche Varianten wie z. B. den Point-Quadtree [Finkel & Bentley 74] oder den Region-Quadtree [Samet 84], für einen Überblick siehe [Samet 89a][Samet 89c].

Zerlegt man die Ebene gleichmäßig nach dem Quadtree-Algorithmus, so lassen sich die einzelnen Teilflächen entsprechend der Z-Ordnung[7] [Orenstein & Merrett 84] linearisieren und der Reihenfolge nach durchnumerieren. Speichert man die Inhalte der Teilflächen und die Numerierung als Schlüssel in einer Tabelle zusammen ab und indiziert mit einem normalen B-Tree (B-Baum) [Bayer & McCreight 72] über diese Schlüssel, so kann man einen Quadtree zur Speicherung und effizienten Wiedergewinnung von Geodaten in einem RDBMS einsetzen. Auf dieser Methode basiert der Geodatenserver von SICAD (siehe S. 31).

Der **R-Tree** (R-Baum) [Guttman 84] ist eine mehrdimensionale Erweiterung des B-Baums zur Speicherung von Intervallen in der Art, daß das bei einem Knoten gespeicherte Intervall immer vollständig im Vaterintervall enthalten ist. Dies erlaubt effizientes Suchen, das ähnlich dem Suchen in einem B-Baum an der Wurzel beginnt und alle Blätter findet, deren Intervall den Suchkriterien entspricht. Es gibt zahlreiche Varianten des R-Tree, z. B. den R+-Tree [Sellis et. al. 87] oder den R*-Tree [Beckmann et. al. 90]. Der R-Baum kann verwendet werden, um Rechtecke zu indizieren und wird in GIS als Suchstruktur über Bounding-Boxes eingesetzt. Ein Beispiel einer Datenbankerweiterung, die den R-Baum einsetzt, ist das 2d-Spatial Datablade von Informix (siehe S. 30).

Nach der Meinung verschiedener Autoren gilt die Erforschung weiterer spezialisierter Zugriffsmethoden nicht mehr als vordringliches Problem [Widmayer 91], da nur geringe Performance-Unterschiede bestehen [Abel 96] und sich bis jetzt keine Methode als allen anderen überlegen erwiesen hat [Gaede 96]. Stattdessen ist es wichtig, die Tragfähigkeit der Methoden in existierenden Systemen unter Beweis zu stellen und Raumzugriffstrukturen zu schaffen, die neben Bereichsanfragen auch komplexere Operationen unterstützen, die für die Anwendungen wichtig sind [Widmayer 91].

4.3 Beispiele existierender Systeme

Noch 1987 wurde die Entwicklung von Datenbanken für räumliche Informationssysteme als ein sehr neues Forschungsthema angesehen. Es existierten wenige experimentelle Datenbanken, welche neue Konzepte zur Behandlung von GIS-Daten integrierten, wie z. B. PANDA [Frank 82a], PROBE [Dayal & Smith 86] oder GENESIS [Batory et. al. 88]. Im Anschluß an die von [Egenhofer & Frank 87] vorgelegte Forschungsagenda wurde das Thema intensiv bearbeitet.

7. Die ursprünglich von [Morton 66] als Peano-Curve eingeführte Methode zur Linearisierung 2-dimensionaler Bilder findet man heute auch unter den Namen Morton-Ordnung [Worboys 95], quad codes [Finkel & Bentley 74] oder locational codes [Abel & Smith 83].

4.3.1 Experimentelle Systeme

Es gibt eine Vielzahl von experimentellen Systemen, die als Basis für ein GIS dienen können. Folgt man der Einteilung in [Güting 94c], kann man drei Arten von Architekturen unterscheiden:

Schichten-Architektur

Hier wird die räumliche Funktionalität auf ein relationales DBMS aufgesetzt. Ein früher Ansatz zerlegt die Geometrien der Geo-Objekte in einzelne Tupel, d. h. in ihre jeweiligen Koordinaten. Er erwies sich als nicht gangbar, da die Wiederzusammensetzung für die Berechnung räumlicher Operationen zu aufwendig ist. Ein weiterer Ansatz verwendet Binary Large Objects (BLOBs), um die Geometrien unzerlegt zu speichern. Beispiele für diesen Ansatz sind:

- **GEOVIEW** speichert Geo-Objekte in relationalen Tabellen [Waugh & Healey 87]. Die Tabelle ENTITY speichert in Spalte DATA vom Datentyp LONG oder BLOB die Koordinaten der Geometrie und in vier weiteren Spalten XMIN, YMIN, XMAX und YMAX das kleinste umgebende Rechteck (Bounding Box). Das erlaubt eingeschränkte räumliche Anfragen auf der Bounding Box, aber nicht auf den Geometrien selbst, da sie in den BLOBs für das Datenbanksystem nicht interpretierbar sind. Die Geometrien sind keine dem RDBMS bekannten Datentypen. Die Tabelle ATTRIBUTES nimmt die Attribute eines in ENTITY abgelegten Geo-Objekts auf.
- **SIRO-DBMS** steht für Spatial Information in a Relational Open architecture DBMS [Abel 89]. Es besteht aus einer Sammlung von Werkzeugen, die in C und SQL programmiert sind und einem Anwendungsentwickler als Prozedurbibliothek zur Verfügung stehen. Wie GEOVIEW legt SIRO-DBMS die Geometrien in BLOBs ab. Während GEOVIEW sich jedoch auf SQL beschränkt, um auf die Daten zuzugreifen, ermöglichen die Werkzeuge von SIRO-DBMS über SQL hinausgehende Abfragen. Die Werkzeuge zerlegen die Anfrage in eine Folge von SQL-Befehlen, leiten sie an das RDBMS weiter und setzen die Ergebnisse zusammen.

Der Nachteil dieses Ansatzes ist, daß die Geometrien der Geo-Objekte in den BLOBs dem DBMS völlig unbekannt sind, jegliche Verarbeitung der Geometrien kann nur auf der Ebene der Applikation erfolgen. Das DBMS kann keine Indizes über die Geometrien aufbauen.

Zweifache Architektur

Bei der zweifachen Architektur werden zwei voneinander unabhängige Systeme zur Analyse von räumlichen und nicht-räumlichen Daten in einer Integrationsschicht zusammengeführt. Bei diesem Ansatz können für räumliche Daten effiziente spezialisierte Datenstrukturen verwendet werden. Ein Beispiel für diese Architektur findet sich in der Arbeit von [Ooi et. al. 89].

Sie basiert auf einem Modul, das die räumlichen Anteile einer Anfrage verarbeitet (spatial processor). Das Problem besteht aber darin, eine Anfrage in ihren räumlichen und nicht-räumlichen Anteil zu zerlegen und eine optimale Reihenfolge zu finden, in der die einzelnen Schritte abgearbeitet werden sollen.

Integrierte räumliche DBMS auf der Basis erweiterbarer DBMS

Univesitäten und Forschungslabors der DB-Hersteller entwickelten eine Reihe von experimentellen Systemen, von denen manche auch in Bezug auf GIS untersucht wurden. Einige Beispiele werden im folgenden aufgeführt:

- **PROBE** ist ein objektorientiertes DBMS zur Speicherung komplexer Objekte entwickelt von der Computer Corporation of America [Dayal et. al. 87]. [Orenstein et. al. 88] wendet PROBE für GIS an. Er abstrahiert räumliche Daten als Punktmengen und legt sie als komplexe Objekte ab. Basierend auf den Punktmengen liefert PROBE näherungsweise eine Obermenge der möglichen Treffer einer Anfrage durch einen geometrischen Filter. In einem zweiten Schritt wird dieses Ergebnis verfeinert. Die Annäherung räumlicher Objekte durch Punktmengen findet sich ähnlich auch bei Oracle Spatial wieder.
- **DASDBS**, das Darmstadt Datenbanksystem, ist ein seit 1983 von Grund auf neu entwikkeltes, erweiterbares System. Für eine rückblickende Zusammenfassung siehe [Schek et. al. 90]. Es besteht aus einem für alle Anwendungen gleichen Datenbankkern (Kernel) und anwendungsspezifischen Frontends. [Wolf 89] erweitert DASDBS um den **Geo-Kernel**, der für räumliche Anwendungen spezialisierte Datentypen und Zugriffsmethoden zur Verfügung stellt. [Breunig 96] baut den **Geo-Model Kernel**, eine Art CASE-Tool für 3D-Geo-Objekte in DASDBS ein.
- **GeoStore** [Balownew et. al. 97] ist ein an der Universität Bonn entwickeltes Informationssystem zur Speicherung und Analyse von drei- und vierdimensionalen Modellen für geologische Anwendungen. Darauf aufbauend wurde GeoToolkit entwickelt als eine Sammlung von C++ Klassen, die in Anwendungen eingebettet werden können. GeoToolkit ist damit kein allgemein verwendbares DBMS, auf dem eine Applikation aufgesetzt werden kann.
- **GéoSabrina** [Larue et. al. 93] basiert auf dem von Infosys entwickelten erweiterbaren RDBMS Sabrina [Gardarin et. al. 87]. Sein erweitertes SQL stellt räumliche Operatoren, Funktionen, Prädikate und Aggregate (group by, sum) bereit, Indizes lassen sich für GEOMETRY genauso wie für NUMBER erzeugen und werden von dem erweiterbaren Optimierer bei der Anfrageverarbeitung genutzt. Das erweiterbare System erlaubt, eigene Abstrakte Datentypen zu definieren oder auf der Basis von GEOMETRY abzuleiten. GeoSabrina kommt einem voll integrierten räumlichen Datenbanksystem schon sehr nahe. Seine Ansätze finden sich später bei Oracle Spatial wieder (siehe Kapitel 4.3.2 “Kommerzielle Systeme“). Beide berücksichtigen allerdings nicht die graphische Präsentation von Geo-Ob-

jekten. Darauf geht diese Arbeit in Kapitel 7.1.2 “Entwicklung eines anwendungsunabhängigen Basismodells“ ein.

Defizite

Die beschriebenen Arbeiten erzielten interessante Einzelergebnisse, die zu Verbesserungen in Teilbereichen geographischer Informationsverarbeitung beitragen können. Die Defizite sind, daß sie entweder zu allgemein oder auf einen Spezialfall konzentriert sind und nicht mit realen GIS-Anwendungen integriert sind. Dadurch lassen sich die Ergebnisse nur schwer unter Einsatzbedingungen überprüfen, wie sie beim Anwender auftreten. Probleme, die der Anwender bei seiner täglichen Arbeit mit den Systemen hat, bleiben so unentdeckt.

4.3.2 Kommerzielle Systeme

Inzwischen gibt es einige kommerzielle Systeme zur Speicherung von Geodaten auf dem Markt. Einige von Ihnen bieten eine räumliche Anfragesprache als Schnittstelle an, andere verfügen nur über eine Programmierschnittstelle. Sie werden zum einen von den Datenbankherstellern direkt in die Datenbank integriert (Oracle Spatial [Oracle 98], IBM Spatial Extender [IBM 98]) oder von Drittanbietern auf der Basis erweiterbarer Datenbanken (Spatial Datablades auf Informix Universal Server [Informix 98], Spatial Query Server von Autometric auf Sybase [Autometric 98]) implementiert, zum anderen arbeiten auch GIS-Hersteller an Erweiterungen relationaler und objekt-relationaler Systeme (SICAD Geodatenserver GDB-X [Sicad 98], ESRI Spatial Database Engine [ESRI 98]). Im folgenden wird zu jeder der drei genannten Gruppen ein System vorgestellt.

Oracle Spatial

Oracle Spatial (OSP) erweitert die (objekt-)relationale Datenbank Oracle 8*i* um den Datentyp SDO_Geometry, der verschiedene grundlegende Geometrien enthalten kann. Oracle Spatial ist der Nachfolger des Spatial Cartridge und der Spatial Data Option, welche auf der relationalen Datenbank Oracle 7.x aufbauen.

Datentypen

Es gibt nur einen neuen Datentyp: SDO_Geometry. Er kann die Geometrien für folgende Elemente aufnehmen: Punkt, Punktcluster, Linienzug mit geraden und kreisbogenförmigen Linienstücken, Flächen mit geraden und kreisbogenförmigen Linienstücken, Rechteck, Kreis, linien- und flächenförmige zusammengesetzte Elemente und Flächen mit Löchern. Zusätzlich können vom System selbst nicht unterstützte Elemente gespeichert werden. Für ihre Bearbeitung muß die Applikation selbst Rechnung tragen.

Anfragen

Oracle Spatial erlaubt Anfragen auf den oben genannten Datentypen durch in SQL eingebettete räumliche Operatoren. Es unterstützt die Operatoren relate und within_distance. Der Operator relate entspricht den standardisierten topologischen Operatoren inside, contains, covers, covered by, touch, overlap, equal, disjoint, anyinteract, determine nach Egenhofer. Within_distance verwendet zur Suche von Objekten in einer bestimmten Entfernung voneinander einen Puffer und nutzt die Indexstrukturen zur Filterung der Daten.

Indizierung

Oracle Spatial erlaubt die Indizierung von Punkten, Linien und Flächen. Für den Index werden Quadtrees mit fixer Kachelgröße für das ganze Gebiet oder Quadtrees mit variabler Kachelgröße für jedes einzelnen Element aufgebaut. Die Kachelgröße oder die Anzahl der Kacheln je Element lassen sich zur Optimierung des Index parametrisieren. Die Anfrageverarbeitung filtert mit dem räumlichen Index die Daten vor, wenn sie den räumlichen Verbund berechnet.

Performance und Benutzerfreundlichkeit

Die Datenbankerweiterung Oracle Spatial liefert unter Nutzung der Indizes schnell das Ergebnis einfacher Anfragen. Komplexere Anfragen sind z. T. schwierig zu formulieren und zeigen durch längere Laufzeiten noch vorhandene Probleme des Optimierers.

Wertung

Oracle Spatial übernimmt die Speicherung von Basis-Geometrien und ermöglicht Abfragen darauf. Höhere GIS-Funktionalität muß auf dieser Basis weiterhin in der Applikation realisiert werden. Das System ist als erweitert relational einzustufen, unterstützt aber nicht objektorientierte Modellierungsmöglichkeiten wie z. B. Vererbung objektrelationaler Systeme. Der Anwender muß noch detaillierte Kenntnisse über die Indizierung haben und z. B. selbst bestimmen, welcher Index für die Bearbeitung einer Abfrage zu verwenden ist. Damit ist Oracle Spatial eine gute Grundlage für einfache Anwendungen. Für komplexe Anwendungen erfordert es die Implementierung einer Zwischenschicht durch einen Applikationsprogrammierer.

Informix 2d-Spatial Datablade

Das 2d-Spatial Datablade ist eine Erweiterung der objektrelationalen DBMS Illustra und Informix Universal Server. Es erweitert das Basis-DBMS um Datentypen zur Speicherung von Geometrien und Graphen und stellt Funktionen zu ihrer Bearbeitung und Analyse zur Verfügung. Es ist möglich, den Datenbankkern durch eigene Erweiterungen zu ergänzen.

Datentypen

Das 2d-Spatial Datablade stellt die Datentypen Rechteck, Kreis, Ellipse, Linie, Linienzug, Punkt, Fläche (Polygon), Polygonmenge und unregelmäßiges Viereck zur Verfügung. Jeder Datentyp wird extern textuell repräsentiert und intern in einer C-Struktur gespeichert. Ein- und Ausgabefunktionen wandeln von einer in die andere Darstellung um.

Anfragen

Die eingeführten geometrischen Datentypen können innerhalb von SQL mit den Operatoren bearbeitet werden. Die üblichen Standardoperatoren wie distance, length, equal, contains, intersects, overlap, etc. stehen zur Verfügung sind aber zum Teil nicht ganz vollständig implementiert.

Indizierung

Die Indizierung des 2d-Spatial Datablade basiert auf dem R-Tree [Guttman 84]. Der R-Tree ist über das Datablade in das DBMS eingebettet, aber nicht Bestandteil des DBMS-Kerns selbst. Der Index kann nur über einer gleichartigen Menge von Objekten gebildet werden, kommen verschieden Objekte vor, müssen sie in eine gemeinsame Darstellung überführt werden.

Performance und Benutzerfreundlichkeit

Für die neuen Datentypen stehen Funktionen zur Verfügung, welche die komfortable Erzeugung von Objekten und den effizienten Zugriff darauf zulassen. Das DBMS läßt sich durch das Schreiben eigener Erweiterungsmodule (Datablades) ergänzen. Es bietet im Gegensatz zu dem oben vorgestellten System von Oracle reichhaltige Möglichkeiten zur objektorientierten Modellierung wie z. B. Vererbung an.

Wertung

Das 2d-Spatial Datablade wurde als ein Beispiel von Informix geschaffen, um zu demonstrieren, was mit der Technologie erweiterbarer DBMS möglich ist und um Drittanbieter zur Entwicklung eigener Produkte anzuregen. Es ist eine gute Basis zur Evaluierung der Eignung erweiterbarer DBMS für GIS, da es weitreichende objektorientierte Möglichkeiten bietet.

SICAD Geodatenserver

Der SICAD Geodatenserver (SICAD GDS, auch GDB-X) ist die geographische Datenbank des Geoinformationssystems SICAD/open. Er speichert Geometrie und Sachdaten in einem RDBMS.

Architektur

Der SICAD GDS besteht aus drei Teilen: den Datenbanktabellen zur Speicherung der Geometrie, den Middleware-Modulen zum Zugriff darauf und der Schnittstelle, welche die Funktionalität nach außen zur Verfügung stellt (s. Abb. 5).

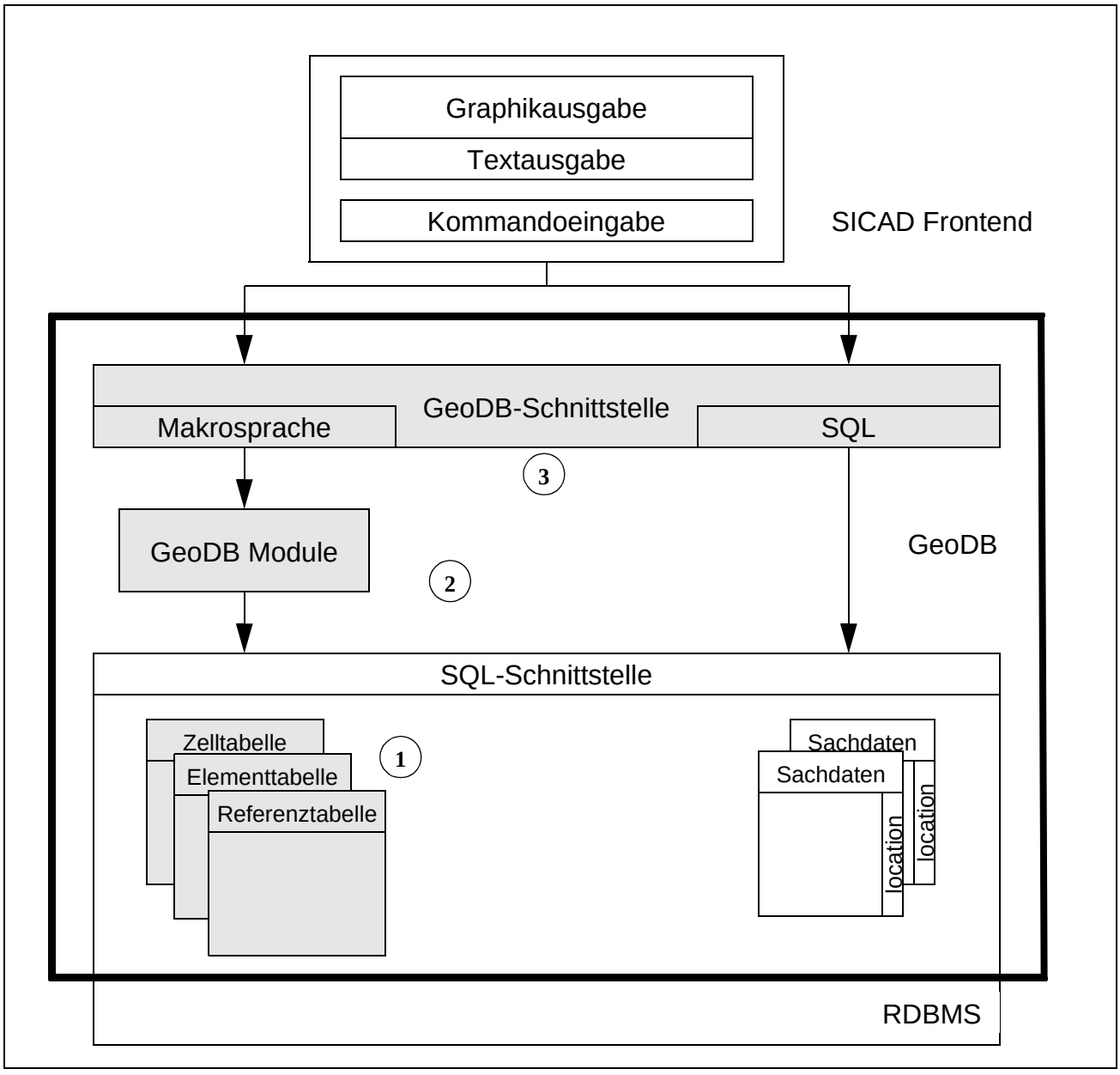


Abb. 5: Die drei Ebenen der Architektur des SICAD Geodatenservers (1) Tabellen, (2) Middleware-Module und (3) Schnittstelle

1. Datenbanktabellen

Wesentlich für die Speicherung der Geometrie von Geodaten im SICAD GDS sind drei Tabellen (s. Abb. 6).

In der **Zelltabelle** werden die Elemente nach der Z-Ordnung des Quadtree abgelegt. Ein B-Tree-Index auf der Zell-ID erlaubt effizienten Zugriff auf räumlich zusammenhängende Bereiche von Daten. Die Elementdaten werden als Large Objects gespeichert.

Die **Elementtabelle** ordnet jedem Element die Zelle zu, in der es abgelegt ist. Die **Referenztabelle** speichert zu jedem Element die Zellen, welche von ihm berührt werden.

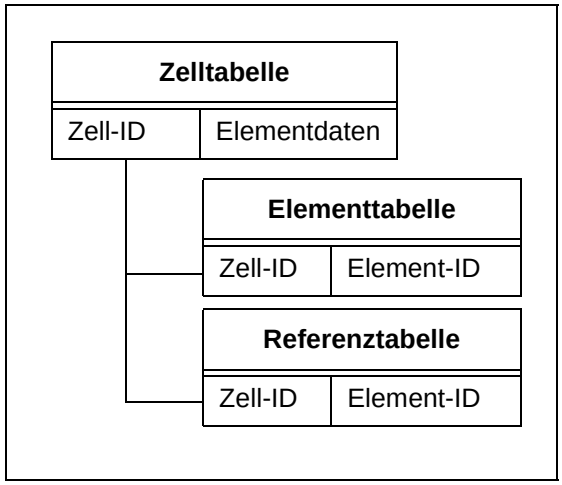
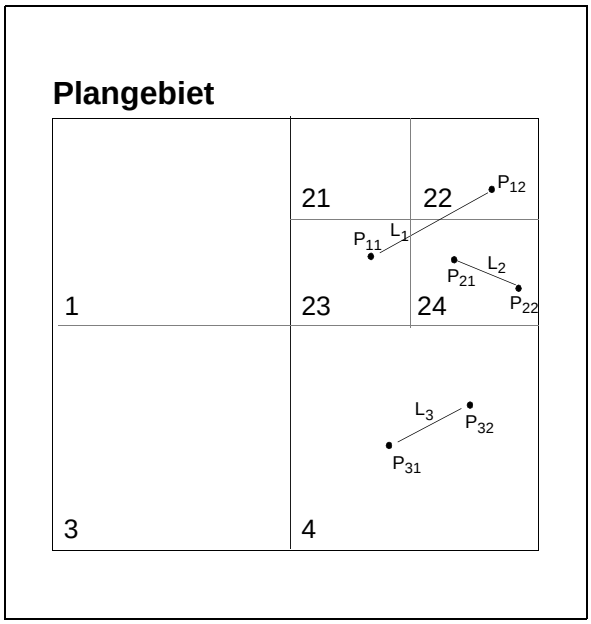


Abb. 6: Speicherung der Geometrie von Geodaten in drei Tabellen des Geodatenservers

Abbildung 7 zeigt am Beispiel eines Plangebiets, wie die Elemente gespeichert werden. Es ist in die Zellen 1, 2, 3 und 4 geviertelt. Zelle 2 ist noch einmal in 21, 22, 23 und 24 unterteilt Das Gebiet enthält die drei Linien L_1, L_2 und L_3 mit ihren jeweiligen Anfangs- und Endpunkten P_{11}, P_{12}, P_{21}, P_{22}, P_{31} und P_{32}.



Zelltabelle	
Zell-ID	Elementdaten
22	P_{12}
23	P_{11}, L_1
24	P_{21}, L_2, P_{22}
4	P_{31}, L_3, P_{32}

Referenztabelle	
Zell-ID	Element-ID
22	L_1
24	L_1

Elementtabelle	
Zell-ID	Element-ID
22	P_{12}
23	P_{11}
23	L_1
24	P_{21}
24	L_2
24	P_{22}
4	P_{31}
4	L_3
4	P_{32}

Abb. 7: Beispiel zur Einteilung des Plangebiets in die Zellen eines Quadtrees und die Speicherung der Elementdaten in der Zelltabelle

Punkte werden in der Zelle gespeichert, in der sie enthalten sind, Linien in der Zelle, die ihren ersten Punkt enthält. Punkt P_{12} kommt z. B. in Zelle 22 und Linie L_1 in Zelle 23. Zu jedem Element ist in der Elementtabelle gespeichert, in welcher Zelle man es findet, wenn man seine Element-ID kennt. In der Referenztabelle ist für alle Linien aufgeführt, welche Zellen sie schneiden, in unserem Fall schneidet die Linie L_1 die Zellen 22 und 24.

2. Middleware-Module

   Über die Middleware-Module des Geodatenservers ist der Zugriff auf die Zellen und einzelne Elemente darin realisiert. Sie lesen, speichern und indizieren die Geometrie, verknüpfen sie mit Sachdaten und stellen die Konsistenz zwischen beiden sicher. Sie sind in C implementiert und kommunizieren über die SQL-Schnittstelle mit dem RDBMS.

3. Schnittstelle

   Die Schnittstelle zum SICAD GDS besteht zum einen aus den proprietären Kommandos zum Zugriff auf die vom SICAD GDS zur Verfügung gestellten Daten und Operationen (GB-Kommandos), zum anderen ist eine SQL-Schnittstelle integriert, über die Kommandos an das darunterliegende RDBMS durchgereicht werden können. Diese SQL-Kommandos werden von einem Parser innerhalb des SICAD GDS bearbeitet, um z. B. Variablenersetzungen durchzuführen.

Datentypen

Der SICAD GDS speichert alle Datentypen von SICAD. Dazu gehören einfache Basisdatentypen wie Punkt (PK), Linie (LI), Linienzug (LY), Kreis (KR) oder Fläche (FL), aber auch für bestimmte Anwendungen wie z. B. das Vermessungswesen oder Energieversorgungsunternehmen spezialisierte Datentypen wie Vermesssungspunkt (PG), Flurstücksnummer (FR) oder Leitung (LT). Für eine detaillierte Aufstellung der Datentypen von SICAD siehe Anhang A.4.

Anfragen

Um Anfragen an den SICAD GDS zu richten, stehen die oben schon beschriebenen SQL- und GB-Kommandos der SICAD GDS-Schnittstelle zur Verfügung. Die GB-Kommandos ermöglichen den Zugriff auf die Geometrie. Ein Beispiel ist hier das Kommando GBSRT zum Suchen in einem Rechteck. Mit den SQL-Kommandos wird auf die Sachdaten zugegriffen. Auf Sach- und Geodaten in einem Kommando zuzugreifen ist nicht möglich. Die beiden Kommandoarten können aber zusammen in einem Skript der SICAD-Makrosprache (SICAD Prozedur) kombiniert werden. Zwischenergebnisse werden dabei in temporären Tabellen gespeichert und können so von einem Kommando zum nächsten weitergereicht werden. Die temporären Tabellen heißen Elementmenge und enthalten die Element-IDs der gefundenen Elemente. Der Inhalt von Elementmengen kann am Bildschirm graphisch dargestellt werden.

Indizierung

Durch die Speicherung der Daten in der Zelltabelle wird ein räumlicher Index (Quadtree, Z-order) aufgebaut. Zusätzliche Möglichkeiten der Indizierung, z. B. über die location-Spalte von Sachdaten bestehen jedoch nicht.

Wertung

Die Architektur baut auf einem Standard-RDBMS auf und enthält einen Parser für SQL. Das vereinfacht, den SICAD GDS um eine geographische Anfragesprache zu erweitern und auf Geo- und Sachdaten gemeinsam innerhalb einer SQL-basierten Sprache zuzugreifen. Ein erster Prototyp namens GISQL [Singer 93] [Costagliola et. al. 95] existiert. Er setzt auf der Forschung von [Ooi et. al. 89] auf.

Die angeführten Eigenschaften machen SICAD zu einer guten Basis, um ein GIS um eine Anfragesprache zu erweitern.

4.4 Defizite existierender Systeme

Die vorgestellten Anfragesprachen und existierenden System weisen eine Reihe von Defiziten auf:

Mangelnde Anwendungsorientierung

Die vorgestellten Anfragesprachen haben ihren Schwerpunkt entweder im Bereich der GIS- oder der DB-Forschung. Erstere sind häufig nur auf die Funktionalität ausgerichtet ohne auf die Performance Rücksicht zu nehmen, letztere konzentrieren sich in der Regel auf ein ganz bestimmtes Problem, wie z. B. zu zeigen, daß eine neue Indizierungmethode bei Rechtecken bessere Ergebnisse liefert als die bisherigen Methoden. Für die Verwendung in einem kommerziellen GIS spielt aber auch die Komplexität der Realisierung eine Rolle und ob die Methode allgemein einsetzbar ist, d. h. auch für andere Datentypen gute Ergebnisse liefert.

Unzureichende Testdaten

Für den Test der experimentellen Systeme werden meist nicht reale Daten verwendet, sondern Testdaten, die in ihrer Quantität aber insbesondere in ihrer Komplexität reduziert sind. Echte im Einsatz befindliche Daten sind oft nicht öffentlich oder teuer. Mit fiktiven Testdaten gemachte Aussagen lassen sich nicht ohne weiteres auf eine bestimmte Anwendung, wie z. B. ein Vermessungsamt einer deutschen Stadt übertragen.

Mangelnde Erweiterbarkeit herkömmlicher GIS

Kommerzielle GIS sind in der Regel historisch gewachsen und nicht leicht in Richtung kombinierter Anfrageverarbeitung zu erweitern. Sie konzentrieren sich auf die Lösung der Kundenprobleme und können sich nicht nur auf eine Teilmenge an Datentypen beschränken. Die Einführung einer neuen Anfragesprache oder einer neuen Architektur zur Unterstützung von Anfragen ist nicht ohne weiteres möglich.

Mangelnde Interoperabilität herkömmlicher GIS

Schon der Datenaustausch über Export und Import von Dateien zwischen verschiedenen herstellerspezifischen GIS-Lösungen ist problematisch. Auf einer höheren Ebene über Funktionen von einem GIS direkt auf Funktionalität oder Daten eines fremden GIS zuzugreifen wird durch die in sich geschlossenen Herstellerwelten verhindert. Für eine ausreichende Interoperabilität fehlen gemeinsame standardisierte Schnittstellen.

Mangelnde GIS-Funktionalität erweiterter DBMS

Kommerzielle erweiterte DBMS sind zur Speicherung von Geometrie geeignet. Damit sind sie eine bessere Grundlage zur Implementierung von GIS als rein relationale Systeme. Sie bieten eine Anfragesprache, die auf den Geometrien operiert. Aber die Basisgeometrien reichen für komplexe GIS-Anwendungen bisher nicht aus. Um eine GIS-Applikation aufzubauen ist also weiterhin eine Zwischenschicht notwendig, welche die fehlende GIS-Funktionalität bietet.

Die Defizite zeigen, daß nach wie vor eine Lücke zwischen den experimentellen und den kommerziellen Systemen zum einen und zwischen den GIS und den DBMS zum anderen besteht.

Zum Schließen dieser Lücken beizutragen, ist das Ziel der in den folgenden Kapiteln beschriebenen Arbeit. Sie bettet eine Anfragesprache in die bestehende Architektur eines kommerziellen GIS ein und evaluiert die sich daraus für Anwendungen ergebenden Möglichkeiten anhand von realen Kundendaten. Ergänzend vergleicht sie diese Lösung mit dem Aufbau eines Gesamtsystems auf der Basis eines erweiterterbaren DBMS.

5. Entwicklung einer geographischen Anfragesprache auf der Basis eines relationalen Datenbankmanagementsystems

Das vorangegangene Kapitel zeigt anhand verschiedener Beispiele aus Forschung und Industrie, wie eine geographische Anfragesprache in eine GIS-Anwendung integriert werden kann. Die vorliegende Arbeit entwickelt in diesem Kapitel eine geographische Anfragesprache innerhalb eines Geoinformationssystems. Sie legt die Spracheigenschaften (5.1) fest und realisiert das Sprachkonzept innerhalb eines existierenden Geoinformationssystems, das auf relationalen Datenbanken basiert (5.2).

5.1 Festlegung der Spracheigenschaften

Zur Beantwortung der Frage, welchen Anforderungen eine in ein GIS eingebettete Anfragesprache genügen muß und welche Datentypen und Operatoren sie unterstützen soll, tragen Theorie (5.1.2) und Praxis (5.1.3) gleichermaßen bei. Sie beeinflussen die Eigenschaften, die Datentypen und die Operatoren der Sprache (5.1.4).

5.1.1 Definition der Geographischen Anfragesprache GQL

Die Geographische Anfragesprache (Geographical Query Language, GQL)

- ist vollständig in ein existierendes Geoinformationssystem integriert
- operiert auf den geographischen Elementen dieses GIS
- basiert auf der standardisierten Datenbankanfragesprache SQL
- erweitert SQL um räumliche Funktionen und Operatoren auf den GIS-Elementen.

Diese Definition integriert Anforderungen aus Theorie und Praxis, die im folgenden genauer untersucht werden.

5.1.2 Anforderungen aus der Theorie

(T1) Mathematisch exakte Definition der Operatoren

Die Bedeutung der Operatoren der Anfragesprache muß mathematisch exakt definiert sein, um sicherzustellen, daß unter gleichem Namen auch gleiche Funktionalität in verschiedenen Systemen zur Verfügung steht. Ein allgemein akzeptiertes Modell sollte der Semantik der ausgewählten Operatoren zu Grunde gelegt werden.

(T2) Integration der Präsentation in die Anfragesprache

Die Ergebnisse einer Anfrage sollen graphisch dargestellt werden können (siehe auch Praxisanforderung 2c). Weiterhin wird gefordert, die graphische Gestaltung des Ergebnisses durch Kommandos der Anfragesprache zu steuern [Egenhofer 94].

5.1.3 Anforderungen aus der Praxis

Die zusammen mit den Partnern erarbeiteten Praxisanforderungen [Ziegler 97] lassen sich in die zwei Bereiche Datentypen und Operatoren unterteilen. Erstens muß die Anfragesprache die Datentypen unterstützen, die dem Anwender sonst auch im GIS zur Verfügung stehen und zweitens sollen die Operatoren auf diesen Datentypen in ihrer Funktionalität den Anwender optimal unterstützen und nicht in seiner gewohnten Arbeitsweise einschränken.

(P1) Anforderungen an die von der Sprache unterstützten Datentypen

(P1a) Verwendung der Datentypen des GIS

Der Anwender erstellt mit Geoinformationssystemen ein Abbild realer geographischer Objekte. Diesem Abbild entspricht ein graphisches Element am Bildschirm und ein Datentyp in der Geodatenbank. Eine Vielzahl unterschiedlicher und zum Teil hochspezialisierter Datentypen wurden entsprechend den verschiedenen Anwendungsbereichen gebildet. Der Praktiker will diese Datentypen in beliebiger Kombination mit den Operatoren verwenden können, die von der Sprache angeboten werden.

(P1b) Interaktive Eingabe von Objekten als Parameter für Operatoren

Für den Anwender ist es wichtig, nicht nur bereits gespeicherte Objekte bearbeiten zu können, sondern auch zum Zeitpunkt der Anfrage ein Objekt angeben zu können, das mit einem Element in der Geodatenbank verglichen werden soll. Dafür müssen geeignete Mittel in der Sprache zur Verfügung gestellt werden.

(P1c) Verfügbarkeit mit der Geometrie gespeicherter Attribute

Üblicherweise enthalten die Datentypen in einer Geodatenbank nicht nur die Geometrie eines Objekts, sondern auch zusätzliche Attribute über die graphische Ausprägung oder die fachliche Bedeutung. Diese sollten dem Anwender bei Anfragen zugänglich sein.

(P1d) Toleranz gegenüber Ungenauigkeiten

Die Daten in einem GIS sind durch die Erfassung mit Meßgeräten mit einer Meßungenauigkeit behaftet. Die Speicherung in einem Rechner als Binärzahlen mit endlich vielen Stellen bewirkt eine begrenzte Darstellungsgenauigkeit. Bei der Erfassung von Daten durch Digitalisierung von analogen Kartenvorlagen und durch Generalisierungseffekte entstehen Ungenauigkeiten. Dadurch können z. B. Punkte am Bildschirm nicht mehr unterscheidbar übereinander liegen, aber das GIS behandelt sie als zwei getrennte Punkte. Deswegen muß es möglich sein, in der Anfragesprache anzugeben, innerhalb welchen Toleranzbereichs zwei Punkte als identisch angesehen werden sollen.

(P1e) Robustheit bei Verwendung nicht unterstützter Datentypen

Moderne Geoinformationssysteme erlauben dem Anwender, sich neue Datentypen zu definieren. Verwendet der Anwender Datentypen als Parameter, die nicht oder noch nicht von der Anfragesprache unterstützt werden, so muß das System stabil bleiben und darf keine falschen Ergebnisse liefern.

(P1f) Thematische Gruppierung von Geodaten in Ebenen

Anwender bauen mehrere Geodatenbanken für das selbe Gebiet zu verschiedenen Themenkreisen auf, die Geodatenbanken selbst können wieder in thematische Ebenen untergliedert sein. Diese Ebenen sollen miteinander verschnitten, überlagert und gemeinsam ausgewertet werden können. Die Datenspeicherung und die Analysefunktionalität muß dafür die Mittel bereitstellen.

(P1g) Komplexe hierarchische Datentypen des GIS

Manche GIS erlauben, aus einfachen Datentypen komplexe Objekte zu bilden[1]. Diese können mehrere geometrische Primitive enthalten. Die Sprache sollte mit diesen komplexen Objekten umgehen können.

(P2)Anforderungen an die Operatoren und die Einsatzmöglichkeiten der Sprache

(P2a) Spracheinbettung

Dem Anwender steht zur Interaktion mit dem GIS in der Regel eine Makro-Sprache zur Verfügung. Ganze Geo-Objekte (z. B. Punkte) oder Teile davon (Attribute) können in Variablen der Makro-Sprache gespeichert werden. Von einer in ein GIS integrierten Abfragesprache wird erwartet, daß sie in die Makro-Sprache eingebettet werden und auf deren Variablen lesend und schreibend zugreifen kann.

(P2b) Rein räumliche Anfragen ohne Sachdatenbezug

Abfragen mit einer Abfragesprache operieren in relationalen Datenbanken auf einer oder mehreren Tabellen. In GIS will der Anwender aber auch ohne Kenntnis der Tabellenstrukturen alle Geo-Objekte eines bestimmten räumlichen Ausschnitts auswählen können. Dafür muß ihm eine Möglichkeit in der Sprache zur Verfügung stehen, mit der er das gesamte Gebiet der Geodatenbank referenzieren kann.

(P2c) Graphische Darstellung der Ergebnisse

Die Ergebnisse von geographischen Anfragen können häufig erst nach einer graphischen Aufbereitung vom Menschen beurteilt werden. Darum müssen die Ergebnisse der Anfragen

1. Als Beispiel kann hier die Modellierung von ATKIS-Objekten durch die OJ-Elemente von SICAD angeführt werden. [Koch 97]) beschreibt z. B. die Modellierung eines Kanals durch das Flußbett (Fläche) und die begrenzenden Dämme (Linien).

zur weiteren Bearbeitung im GIS zur Verfügung stehen. Dort können sie vom Anwender mit den graphischen Möglichkeiten des GIS weiter verarbeitet werden, um z. B. eine Karte zu erstellen. Werden Karten als Ausgabe benötigt, so bestehen hohe Anforderungen an die kartographische Qualität der Karte. Diese Qualität mit automatisch generierten Karten zu erreichen, ist ein noch nicht gelöstes Problem (Generalisierung), aber nicht Inhalt dieser Arbeit.

(P2d) Großräumige Fortführung

Anwender sehen einen Einsatzbereich der Sprache in der großräumigen Fortführung von Geodaten. Dazu muß die Sprache nicht nur die Auswahl, sondern auch die Manipulation von Daten unterstützen. Spezielle Operatoren zum Auftrennen und Zusammenführen von der Geometrie sind hier ebenso denkbar wie Operatoren zum Setzen von Attributwerten.

(P2e) Fehlererkennung

Ein Einsatzbereich ist die Erkennung von Fehlern in der Geodatenbank durch die Auswahl aller Elemente, die eine bestimmte Kombination von Bedingungen erfüllen.

(P2f) Unterstützung typischer Vorgehensweisen

Die traditionelle Vorgehensweise bei der Durchführung von Analysen mit mehreren Bedingungen in GIS ist, die Suche über eine der Bedingungen (Sachdaten oder Geometrie) zu beginnen und dann das Ergebnis weiter zu verfeinern. Dabei werden Zwischenergebnisse gespeichert, auf denen die weiteren Bearbeitungsschritte aufsetzen. Daher ist es wünschenswert, daß die Ergebnisse der Anfragesprache in geeigneter Form zur Weiterbearbeitung zur Verfügung stehen und daß die Anfragesprache auf Zwischenergebnissen anderer Analysewerkzeuge aufsetzen kann.

(P2g) Unterstützung der Standards

Der zunehmende Einsatz von GIS als ein in die Informations-Infrastruktur eines Unternehmens oder einer Behörde integrierter Baustein erfordert standardisierte Komponenten und Schnittstellen. Die Anfragesprache und ihre Operatoren müssen den entsprechenden Standards der ISO und des OGC entsprechen.

5.1.4 Auswahl der Datentypen und Operatoren

Die letzten drei Abschnitte enthalten Anforderungen aus Theorie und Praxis an eine Anfragesprache. Die Auswahl der Datentypen und Operatoren und die Festlegung der Spracheigenschaften von GQL berücksichtigt diese Anforderungen.

(1) Unterstützte Datentypen

Trotz unterschiedlicher graphischer Darstellungen und fachlicher Bedeutungen lassen sich die Datentypen eines GIS nach der räumlichen Ausdehnung ihrer Geometrie in die Gruppen

Punkt, Linie und Fläche einteilen. Typische Datentypen für Basis- und Fach-GIS und ihre Zuordnung zu diesen Gruppen zeigt Tabelle 6, eine detaillierte Darstellung der Datentypen ei-

Gruppe	Typische Datentypen eines Basis-/Fach-GIS	Konstante Elemente
Punkt	einfacher Punkt, Text, Symbol, Vermessungspunkt, Flurstücksnummer, Objekt	point(x y)
Linie	Linie, Linienzug, Bogen, Spline, Leitung, Kanal	line(x_1 y_1, x_2 y_2, … x_n y_n)
Fläche	Fläche, Rechteck, Kreis	polygon(x_1 y_1, x_2 y_2, … x_n y_n) rectangle(x1 y1, x2 y2) circle(x1 y1, radius)

Tab. 6: Typische Datentypen eines Basis-/Fach-GIS und ihre Abbildung auf die Gruppen Punkt, Linie und Fläche

nes GIS befindet sich in Anhang A.4. Die Anfragesprache GQL unterstützt alle Datentypen des GIS, die einer dieser Gruppen zugeordnet werden können (Praxisanforderung 1a). Der Anwender kann mit den konstanten Elementen interaktiv Objekte eingeben und damit in Anfragen einbeziehen. (Praxisanforderung 1b).

(2) Ausgewählte Operatoren

GQL stellt 8 Operatoren und 5 Funktionen zur Verfügung. Operatoren vergleichen die Geometrien zweier Elemente bezüglich eines gewählten Kriteriums und liefern TRUE zurück, wenn es erfüllt ist, sonst FALSE. Funktionen bestimmen eine Eigenschaft eines Elements und liefern als Ergebnis eine Zahl, z. B. die Länge einer Linie. Im folgenden wird ihre Funktionalität beschrieben, und inwieweit sie mit den beschriebenen Anforderungen übereinstimmt.

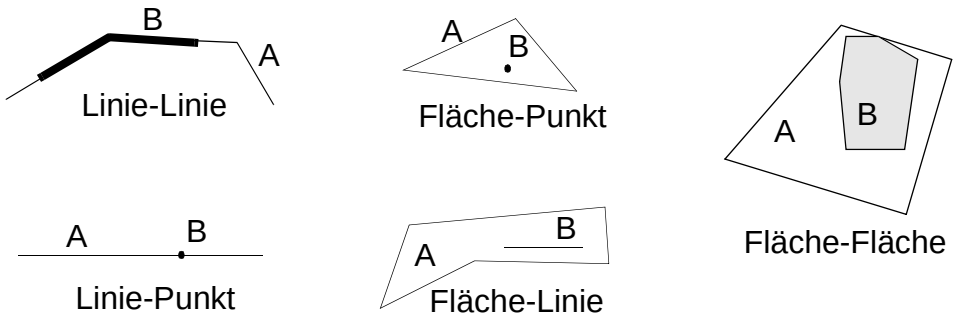
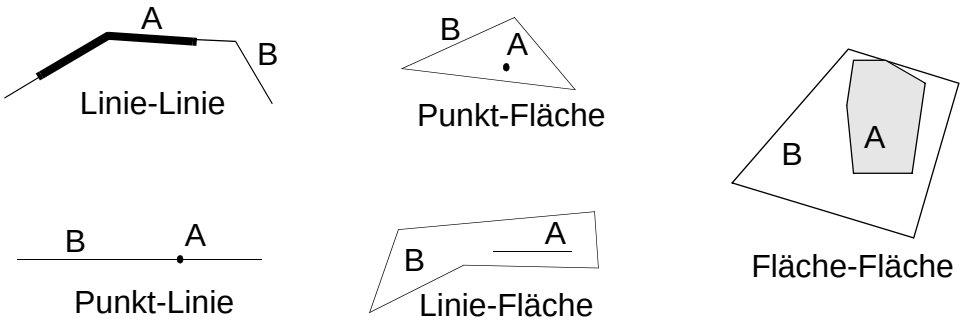
Die Funktionen dienen zur Bestimmung der Eigenschaften von Objekten in der Geodatenbank (siehe Tabelle 7). Area, len(gth) und perimeter berechnen einfache Objekteigenschaften wie Flächeninhalt, Länge und Umfang eines Objekts. Distance liefert die Entfernung zwischen zwei Elementen. Die Funktion value_of liefert die Werte von zusammen mit den Geodaten gespeicherten Attributen eines geographischen Elements in der Datenbank (Praxisanforderung 1c).

Funktion	Parameter		Konstanten erlaubt	Beschreibung
	erster	zweiter		
1. area	Fläche	keiner	nein	Flächeninhalt der Fläche
2. len(gth)	Linie		nein	Länge der Linie
3. perimeter	Fläche		nein	Umfang der Fläche
4. value_of	Punkt, Linie oder Fläche	Deskriptor-name	nein	Wert des Deskriptors (Text, Zahl)
5. distance	Punkt, Linie oder Fläche	Punkt, Linie oder Fläche	ja	Entfernung zwischen zwei Elementen[a]

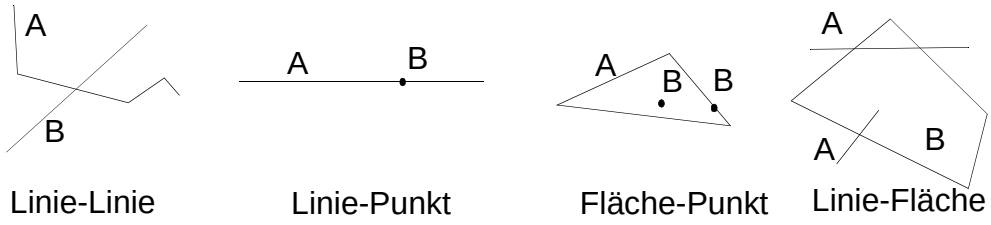
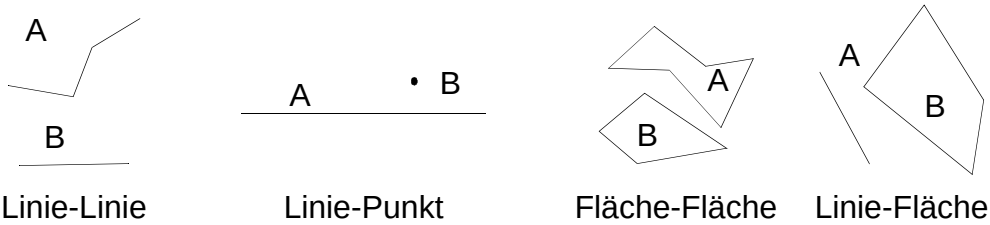
Tab. 7: Beschreibung der für GQL ausgewählten Funktionen

a. Distance berechnet die euklidische Entfernung zwischen zwei Punkten, bei Linien und Flächen die kleinste Entfernung zwischen zwei beliebigen Punkten auf der Linie oder Fläche. Dies entspricht der map distance aus [TC 211 N 549]. Entfernungen auf dem Ellipsoid oder unter Berücksichtigung des Terrains sind nicht vorgesehen.

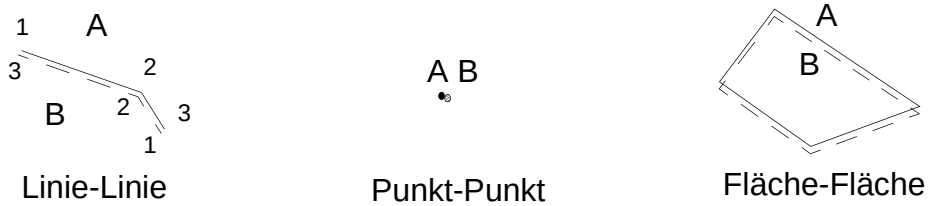
GQL unterstützt die Operatoren contains, within, cross, disjoint, equals, intersect, overlap und touch (siehe Tabelle 8). Ihre Semantik entspricht der OGC (Praxisanforderung 2g) und dem dort verwendeten und in Kapitel 4.1.1 erläuterten Dimensionally Extended 9-Intersection Model (DE-9IM), wodurch sie mathematisch exakt definiert sind (Theorieanforderung 1).

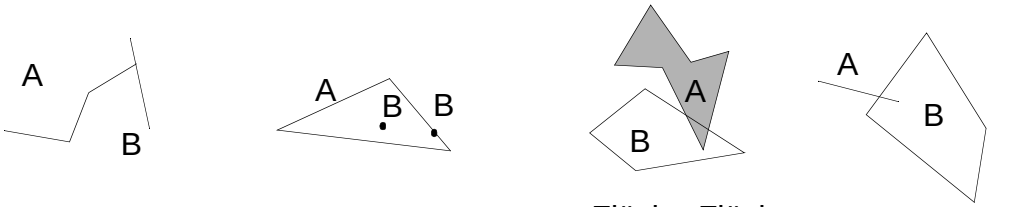
<table>
<tr><th rowspan="2">Operator</th><th colspan="2">Parameter</th><th rowspan="2">9-Schnittmengen-Matrix (DE-9IM)</th><th rowspan="2">Beschreibung / Beispiel</th></tr>
<tr><th>erster (A)</th><th>zweiter (B)</th></tr>
<tr><td rowspan="2">1. contains</td><td>Linie</td><td>Linie</td><td rowspan="5">$\begin{bmatrix} T & * & * \\ * & * & * \\ F & F & * \end{bmatrix}$</td><td rowspan="5">A enthält B vollständig, die Ränder dürfen sich berühren.
Inverse zu within.</td></tr>
<tr><td>Linie</td><td>Punkt</td></tr>
<tr><td>Fläche</td><td>Fläche</td></tr>
<tr><td>Fläche</td><td>Linie</td></tr>
<tr><td>Fläche</td><td>Punkt</td></tr>
<tr><td colspan="5"> 
</td></tr>
<tr><td rowspan="2">2. within</td><td>Linie</td><td>Linie</td><td rowspan="5">$\begin{bmatrix} T & * & F \\ * & * & F \\ * & * & * \end{bmatrix}$</td><td rowspan="5">A liegt vollständig im Inneren von B, darf aber den Rand berühren.
Inverse zu contains.</td></tr>
<tr><td>Punkt</td><td>Linie</td></tr>
<tr><td>Fläche</td><td>Fläche</td></tr>
<tr><td>Linie</td><td>Fläche</td></tr>
<tr><td>Punkt</td><td>Fläche</td></tr>
<tr><td colspan="5"> 
</td></tr>
</table>

Tab. 8: Beschreibung der für GQL ausgewählten Operatoren

<table>
<tr><th rowspan="2">Operator</th><th colspan="2">Parameter</th><th rowspan="2">9-Schnittmengen-Matrix (DE-9IM)</th><th rowspan="2">Beschreibung / Beispiel</th></tr>
<tr><th>erster (A)</th><th>zweiter (B)</th></tr>
<tr><td rowspan="5">3. cross</td><td>Linie</td><td>Linie</td><td>$\begin{bmatrix} 0 & * & * \\ * & * & * \\ * & * & * \end{bmatrix}$</td><td>zwei Linien kreuzen sich in einem oder mehreren Punkten</td></tr>
<tr><td>Linie</td><td>Punkt</td><td rowspan="3">$\begin{bmatrix} T & * & T \\ * & * & * \\ * & * & * \end{bmatrix}$</td><td rowspan="3">eine Linie oder Fläche wird von einem Punkt gekreuzt, eine Linie kreuzt eine Fläche
Linien mit Linien und Flächen mit Flächen können sich nicht kreuzen</td></tr>
<tr><td>Fläche</td><td>Punkt</td></tr>
<tr><td>Linie</td><td>Fläche</td></tr>
<tr><td colspan="4"> 
</td></tr>
<tr><td rowspan="9">4. disjoint</td><td>Punkt</td><td>Punkt</td><td rowspan="9">$\begin{bmatrix} F & F & * \\ F & F & * \\ * & * & * \end{bmatrix}$</td><td rowspan="9">zwei beliebige Elemente haben keine gemeinsamen Punkte</td></tr>
<tr><td>Linie</td><td>Linie</td></tr>
<tr><td>Fläche</td><td>Fläche</td></tr>
<tr><td>Punkt</td><td>Fläche</td></tr>
<tr><td>Fläche</td><td>Punkt</td></tr>
<tr><td>Punkt</td><td>Linie</td></tr>
<tr><td>Linie</td><td>Punkt</td></tr>
<tr><td>Linie</td><td>Fläche</td></tr>
<tr><td>Fläche</td><td>Linie</td></tr>
<tr><td colspan="4"> 
</td></tr>
</table>

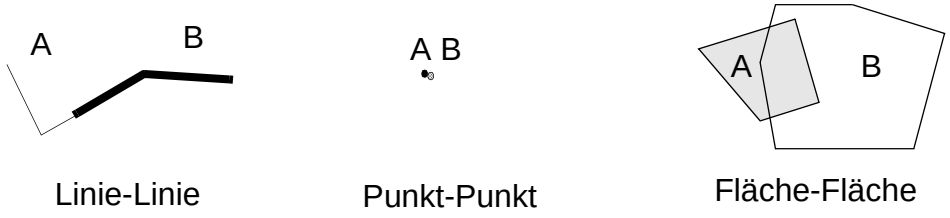
Tab. 8: Beschreibung der für GQL ausgewählten Operatoren

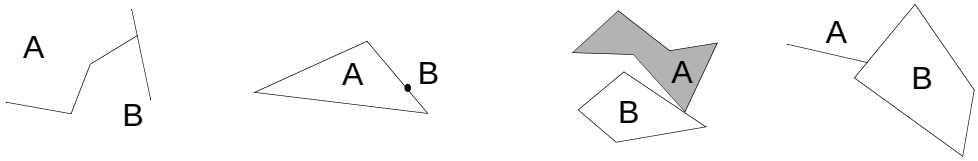
<table>
<tr><th rowspan="2">Operator</th><th colspan="2">Parameter</th><th rowspan="2">9-Schnittmengen-Matrix (DE-9IM)</th><th rowspan="2">Beschreibung / Beispiel</th></tr>
<tr><th>erster (A)</th><th>zweiter (B)</th></tr>
<tr><td>5. equals</td><td>Punkt</td><td>Punkt</td><td rowspan="3">$\begin{bmatrix} T & F & F \\ F & T & F \\ F & F & T \end{bmatrix}$</td><td rowspan="3">je zwei Punkte, Linien oder Flächen stimmen in ihrer Geometrie überein, die Toleranz wird berücksichtigt</td></tr>
<tr><td>Linie</td><td>Linie</td></tr>
<tr><td>Fläche</td><td>Fläche</td></tr>
<tr><td></td><td colspan="4"> 

Linie-Linie Punkt-Punkt Fläche-Fläche</td></tr>
<tr><td>6. intersect</td><td>Punkt</td><td>Punkt</td><td rowspan="9">$\begin{bmatrix} T & * & * \\ * & * & * \\ * & * & * \end{bmatrix}$ or $\begin{bmatrix} * & * & * \\ * & T & * \\ * & * & * \end{bmatrix}$ or $\begin{bmatrix} * & T & * \\ * & * & * \\ * & * & * \end{bmatrix}$ or $\begin{bmatrix} * & * & * \\ T & * & * \\ * & * & * \end{bmatrix}$</td><td rowspan="9">zwei beliebige Elemente haben mindestens einen Punkt gemeinsam egal ob in ihrem Inneren oder auf dem Rand
intersect = not disjoint</td></tr>
<tr><td>Linie</td><td>Linie</td></tr>
<tr><td>Fläche</td><td>Fläche</td></tr>
<tr><td>Punkt</td><td>Fläche</td></tr>
<tr><td>Fläche</td><td>Punkt</td></tr>
<tr><td>Punkt</td><td>Linie</td></tr>
<tr><td>Linie</td><td>Punkt</td></tr>
<tr><td>Linie</td><td>Fläche</td></tr>
<tr><td>Fläche</td><td>Linie</td></tr>
<tr><td></td><td colspan="4"> 

Linie-Linie Fläche-Punkt Fläche-Fläche Linie-Fläche</td></tr>
</table>

Tab. 8: Beschreibung der für GQL ausgewählten Operatoren

<table>
<tr><th rowspan="2">Operator</th><th colspan="2">Parameter</th><th rowspan="2">9-Schnittmengen-Matrix (DE-9IM)</th><th rowspan="2">Beschreibung / Beispiel</th></tr>
<tr><th>erster (A)</th><th>zweiter (B)</th></tr>
<tr><td rowspan="4">7. overlaps</td><td>Punkt</td><td>Punkt</td><td rowspan="2">$\begin{bmatrix} T & * & T \\ * & * & * \\ T & * & * \end{bmatrix}$</td><td rowspan="2">zwei Punkte oder Flächen überlappen</td></tr>
<tr><td>Fläche</td><td>Fläche</td></tr>
<tr><td>Linie</td><td>Linie</td><td>$\begin{bmatrix} 1 & * & T \\ * & * & * \\ T & * & * \end{bmatrix}$</td><td>zwei Linien überlappen in einem Linienstück</td></tr>
<tr><td colspan="4"> 

Linie-Linie Punkt-Punkt Fläche-Fläche</td></tr>
<tr><td rowspan="10">8. touch</td><td>Punkt</td><td>Punkt</td><td rowspan="10">$\begin{bmatrix} F & T & * \\ * & * & * \\ * & * & * \end{bmatrix}$ or $\begin{bmatrix} F & * & * \\ T & * & * \\ * & * & * \end{bmatrix}$ or $\begin{bmatrix} F & * & * \\ * & T & * \\ * & * & * \end{bmatrix}$</td><td rowspan="10">zwei Elemente berühren sich mit ihrem Rand
Punkte haben keinen Rand, also Ergebnis unbekannt</td></tr>
<tr><td>Linie</td><td>Linie</td></tr>
<tr><td>Fläche</td><td>Fläche</td></tr>
<tr><td>Punkt</td><td>Fläche</td></tr>
<tr><td>Fläche</td><td>Punkt</td></tr>
<tr><td>Punkt</td><td>Linie</td></tr>
<tr><td>Linie</td><td>Punkt</td></tr>
<tr><td>Linie</td><td>Fläche</td></tr>
<tr><td>Fläche</td><td>Linie</td></tr>
<tr><td colspan="4"> 

Linie-Linie Fläche-Punkt Fläche-Fläche Linie-Fläche</td></tr>
</table>

Tab. 8: Beschreibung der für GQL ausgewählten Operatoren

(3) Spracheigenschaften

Toleranz

GQL berücksichtigt bei der Berechnung der Operatoren eine Toleranz. Beträgt der Abstand zwischen zwei Punkten weniger als die Toleranz, so werden sie als gleich angesehen. Die Operatoren berechnen ihre Ergebnisse auf der Basis der Toleranz (Praxisanforderung 1d).

Die Toleranz kann vom Benutzer mit GQL-Befehlen eingestellt werden. Nicht an die Daten angepaßte Werte für die Toleranz können jedoch zu Ergebnissen führen, die zwar mathematisch der Definition des Operators entsprechen, aber der intuitiven Erwartung des Benutzers widersprechen.

Graphische Präsentation

GQL bietet im Widerspruch zu Theorieanforderung 2 keine sprachlichen Elemente an, um die graphische Gestaltung des Ergebnisses zu steuern. Diese Arbeit sieht GQL, wie [Gradwell 90a] SQL, nicht als hauptsächliche Schnittstelle zum Endbenutzer einer Anwendung. Stattdessen wird auf der Basis von GQL eine Anwendung mit einer graphischen Benutzeroberfläche (Graphical User Interface, GUI) von einem Anwendungsentwickler geschaffen, die der Endanwender dann benutzt. Der Anwendungsentwickler nutzt die schon bestehende Funktionalität des GIS zur Darstellung der Ergebnisse. Die Ergebnisse der Anfrage mit den sonstigen Werkzeugen des GIS weiterbearbeiten zu können, ist darum wichtiger als, die Darstellung innerhalb der Anfragesprache steuern zu können.

Wichtiger als eine Steuerung der Darstellung innerhalb der Anfragesprache ist es darum, die Ergebnisse der Anfrage mit den sonstigen Werkzeugen des GIS weiterbearbeiten zu können.

Vollständige Integration ins GIS

GQL ist in die SQL-Schnittstelle der zu Grunde liegenden Geodatenbank vollständig integriert. Dadurch kann GQL problemlos in Prozeduren mit anderen Kommandos der Makrosprache verwendet werden. Es kann die Variablen der Anwendung lesen und schreiben. Elemente tauscht es mit anderen Werkzeugen des GIS über das auf Seite 33 beschriebene Konzept der Elementmengen aus. Dadurch ist Praxisanforderung 2a voll erfüllt.

Berücksichtigung der Standards

Die Entwicklung der Sprache GQL entspricht dem aktuellen Stand der Standardisierung wie unter Abschnitt 4.1.4 vorgestellt. Die Definition der räumlichen Operatoren in den Standards folgt einheitlich dem DE-9IM, es ist auch die Basis der Operatoren von GQL. GQL unterstützt die Basistypen für Punkte, Linien und Flächen. Auf ihrer Basis arbeiten die räumlichen Operatoren. Abgeleitete Datentypen werden auf die Basistypen abgebildet und können damit auch von den Operatoren bearbeitet werden.

5.2 Design und Implementierung

Von den in Abschnitt 4.3.2 vorgestellten kommerziellen Systemen zur Speicherung und Bearbeitung von Geodaten ist ausschließlich der SICAD Geodatenserver (SICAD GDS) voll in ein Geoinformationssystem integriert und baut auf relationalen Datenbanken auf. In seine Makrosprache ist ein SQL-Interpreter zum Zugriff auf Sachdaten integriert. Diese Eigenschaften machen ihn zu einer guten Basis für den im folgenden beschriebenen Ansatz zur Integration einer geographischen Anfragesprache (GQL) in ein GIS.

5.2.1 Architektur

Die geographische Anfragesprache wird in die Architektur des Geodatenservers (s. Abb. 8, zum Vergleich auch Abb. 5) eingebettet. Ebenso wie der Geodatenserver besteht die Architektur von GQL aus drei Teilen: den Datenbanktabellen, den Middleware-Modulen und der Schnittstelle

1. Datenbanktabellen

   Die Datenbanktabellen für GQL nehmen Zwischenergebnisse bei der Berechnung der räumlichen Operatoren und Funktionen auf. Auf sie wird in Abschnitt 5.2.2 noch näher eingegangen.

2. Middleware-Module

   Die Middleware-Module von GQL bauen auf denen des Geodatenservers auf. Sie parsen das GQL/SQL-Kommando, bearbeiten die Operatoren von GQL, legen die Zwischenergebnisse in Tabellen ab und reichen ein SQL-Kommando an die Datenbank weiter, das auf der Basis der Zwischenergebnisse das Gesamtergebnis berechnet. Der genaue Ablauf der Bearbeitung eines GQL-Kommandos wird in Abschnitt 5.2.3 vorgestellt.

3. Schnittstelle

   GQL erweitert die vorhandene SQL-Schnittstelle des Geodatenservers um die in Abschnitt 5.1.4 vorgestellten Operatoren. Dadurch ist GQL ebenso in Prozeduren der Makrosprache verwendbar wie andere SQL-Kommandos.

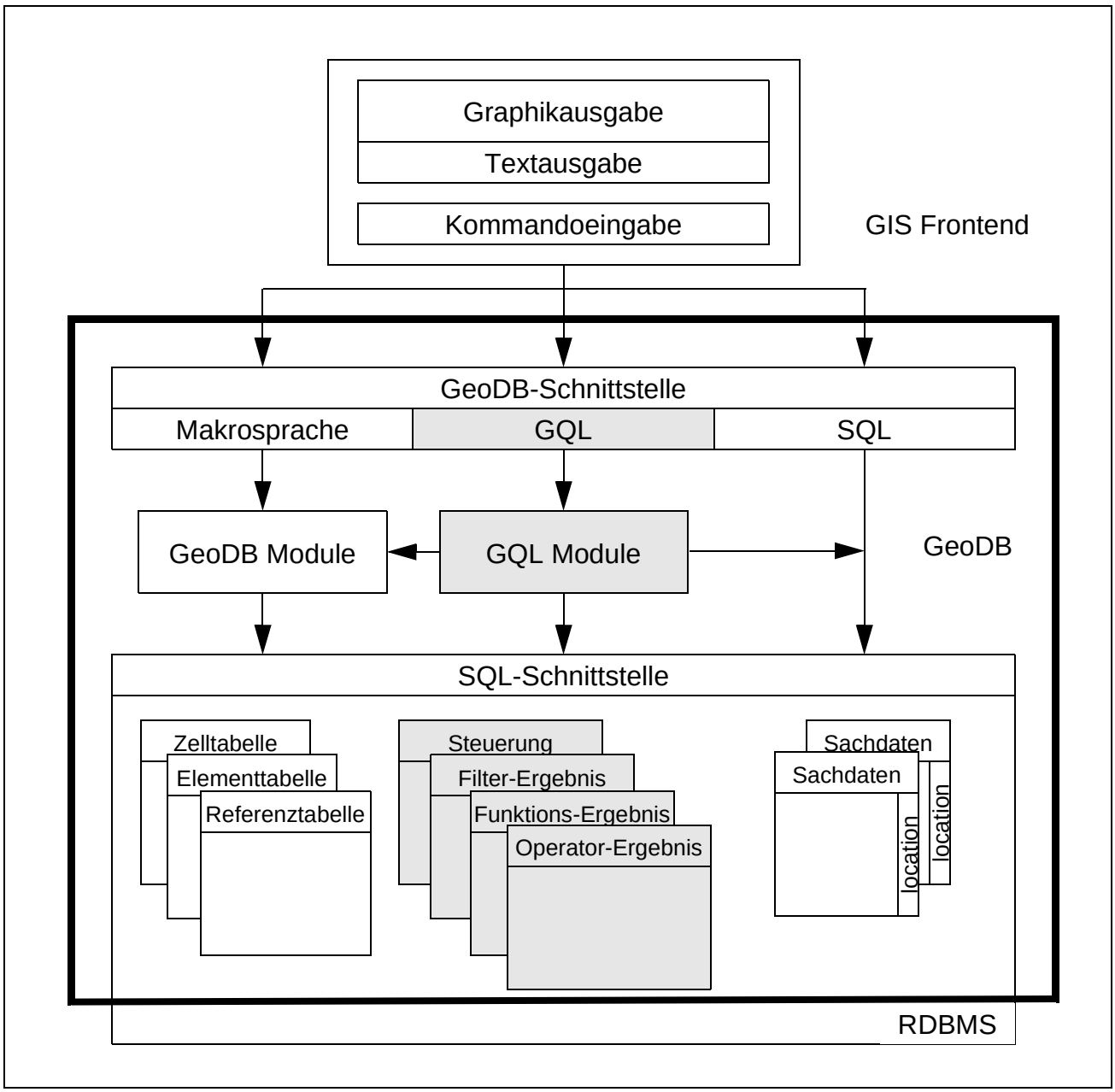


Abb. 8: Einbettung der geographischen Anfragesprache GQL in die Architektur eines Geodatenservers auf der Basis eines relationalen Datenbankmanagementsystems, z. B. Oracle.

5.2.2 Datenbankdesign

Es gibt drei Arten von GQL-Tabellen: Steuerungstabellen, Filtertabellen und Ergebnistabellen. Die Einträge der Steuerungstabellen beeinflussen, welche Elemente GQL kennt (Elementtypentabelle) und in welcher Reihenfolge es die Operatoren in einem GQL-Befehl abarbeitet (Prioritätstabelle). Die Filter-Ergebnistabelle nimmt die IDs der Elemente auf, die von einem räumlichen Filterschritt zur weiteren Bearbeitung ausgewählt werden. Die Ergebnistabellen halten das Ergebnis des Verfeinerungsschritts (Operator-Ergebnistabelle) oder der Berechnung von Funktionen in der Projektion (Funktionsergebnis-Tabelle) und eine Identifikation der beteiligten Elemente zur weiteren Verarbeitung durch das RDBMS bereit.

Elementtypentabelle
(Auszug)

Elementtypen	Gruppe
Fläche	Fläche
Kreis	Fläche
Bogen	Linie
Linie	Linie
Leitung	Linie
Linienzug	Linie
Spline	Linie
Flurstücksnummer	Punkt
Vermessungspunkt	Punkt
Symbol	Punkt
Text	Punkt
...	...

Tab. 9: Die Tabelle der Elementtypen

Elementtypentabelle

Die Elementtypentabelle ordnet die von GQL unterstützten Elementtypen des Geodatenservers einer der Gruppen Punkt, Linie oder Fläche zu (s. Tab. 9). Nicht eingetragene Elementtypen des Geodatenservers werden nicht bearbeitet.

Prioritätstabelle

Die Prioritätstabelle steuert die Ausführungsreihenfolge der Operatoren in einer GQL-Anfrage (s. Tab. 10). Dazu weist sie jedem Operator abhängig von der Art seiner Eingabeparameter eine Priorität aus dem Wertebereich von 1 bis 100 zu. 1 ist die höchste Priorität, 100 die niedrigste. Eingabeparameter können konstante Elemente oder Tabellenspalten sein. Die Priorität ist um so höher (näher an 1), je mehr der Eingabeparameter des Operators Konstanten sind. Die Priorität ist niedriger (näher an 100), wenn ein oder beide Parameter Tabellenspalten sind, da der Operator für jedes Element in der Spalte ausgeführt werden muß. Operatoren mit einer höheren Priorität werden vor denen mit einer niedrigeren Priorität abgearbeitet. Die Prioritäten in einer Tabelle zu speichern ermöglicht, das System automatisch oder durch einen Administrator zur Laufzeit anzupassen.

Prioritätstabelle
(Auszug)

Operator		Anzahl konstante Parameter	Priorität
ID	Name		
1	DISTANCE	0	30
1	DISTANCE	1	10
1	DISTANCE	2	1
2	LEN	0	20
2	LEN	1	1
2	LEN	2	NULL
...	...	...	...
12	EQUALS	0	7
12	EQUALS	1	2
12	EQUALS	2	1
13	DISJOINT	0	8
13	DISJOINT	1	2
13	DISJOINT	2	1

Tab. 10: Die Tabelle der Prioritäten steuert die Ausführungsreihenfolge der Operatoren.

Filter-Ergebnistabelle

Die Filter-Ergebnistabelle speichert temporär die Element-IDs der Elemente, welche die Bedingung eines räumlichen Filters erfüllen (s. Tab. 11). Sie wird verwendet, wenn ein Join auf der Basis eines räumlichen Operators berechnet wird, um die Anzahl der Elemente zu reduzieren, für die das Join-Kriterium überprüft werden muß (zu den Datenflüssen siehe auch 5.2.3).

Filter-Ergebnistabelle
Element-Identifikation

Tab. 11: Die Filter-Ergebnistabelle

Operator-Ergebnistabelle

Die Operator-Ergebnistabelle speichert die Schlüssel der Tupel, welche die Bedingung des Operators erfüllen (s. Tab. 12). Mehrere Operatoren in einer Anfrage werden durchnumeriert (Operatornummer). Die als erster und zweiter Parameter angegebenen Elemente werden durch ihre Tabelle und ID referenziert. Sie enthält somit eine Auswahl an Elementen, die von der Datenbank mit reinem SQL weiterbearbeitet werden kann.

Operator-Ergebnistabelle				
Operator-nummer	Element 1		Element 2	
	Tabelle	ID	Tabelle	ID

Tab. 12: Die Operator-Ergebnistabelle

Funktions-Ergebnistabelle

Die Funktions-Ergebnistabelle speichert die Zwischenergebnisse, die anfallen, wenn die Projektion der GQL-Anfrage geographische Funktionen wie z. B. die Berechnung der Länge einer Linie enthält (s. Tab. 13). Jede Funktion in der Projektion wird durch eine Funktionsnummer identifziert. Die als erster oder zweiter Parameter vorkommenden Elemente werden über den Tabellennamen und ihre Element-ID identifiziert. Das berechnete Ergebnis wird in eine der vier Ergebnisspalten eingetragen. Eine Ausnahme bildet die Funktion value_of, die den Wert eines Parameters oder Deskriptors eines Elements liefert. Da sie nicht nur Fließkommazahlen, sondern auch Text enthalten können, gibt *par* den Namen des Parameters an und in *int* oder *str* kann dann das jeweilige Ergebnis stehen.

Funktions-Ergebnistabelle								
Funktions-nummer	Element 1		Element 2		Ergebnisse			
	Tabelle	ID	Tabelle	ID	real	par	int	str

Tab. 13: Die Funktions-Ergebnistabelle speichert die Ergebnisse der Funktionen zur weiteren Verarbeitung mit SQL

5.2.3 Datenströme

Die Abarbeitung eines GQL-Kommandos betrifft alle drei Ebenen der Architektur: die Schnittstelle zum Benutzer, die GeoDB-Middleware und die relationale Datenbank.

Über die graphische und textuelle **Schnittstelle** formuliert der Anwender die Anfrage. Sie wird vom Frontend an die GeoDB-Middleware übergeben. Findet der dort integrierte Parser GQL-Operatoren oder -Funktionen in dem SQL-Ausdruck, so ruft er die GQL-Hauptroutine auf und übergibt ihr den Parsebaum.

Die **GQL-Middleware** bringt die Operatoren nach heuristischen Regeln, die in der Prioritätstabelle abgelegt sind, in eine Ausführungsreihenfolge und berechnet die Ergebnisse der einzelnen Operatoren, jeden für sich und der Reihe nach. Die Zwischenergebnisse legt GQL in den Zwischenergebnistabellen zusammen mit den Element-IDs der Elemente ab, die zu dem Ergebnis geführt haben. Außerdem ändert GQL den SQL-Ausdruck so ab, daß der GQL-Operator durch einen äquivalenten Ausdruck ersetzt wird, der reines SQL enthält und die Ergebnisse aus den Ergebnistabellen über einen Join dazubindet.

Sobald alle Operatoren und Funktionen abgearbeitet sind, übergibt GQL den veränderten Ausdruck, der nur noch reines SQL enthält, an die **Datenbank**. Diese errechnet das Gesamtergebnis und liefert je nach Projektion entweder eine Tabelle oder legt die IDs der gesuchten Elemente in einer Elementmenge ab. Dort steht sie zur Visualisierung und Weiterbearbeitung im GIS zur Verfügung.

Von den in Kapitel 4.2 beschriebenen Methoden zur Verarbeitung von Anfragen setzt GQL die folgenden ein:

- Eine Voroptimierung wird von GQL durch eine heuristische Priorisierung der Ausführungsreihenfolge von räumlichen Operatoren vorgenommen. Diese wird durch eine Tabelle gesteuert. Das entspricht der Forderung von [Stonebraker 96] nach flexiblen, extern beeinflußbaren Mechanismen zur Steuerung der Anfrageverarbeitung.
- GQL filtert die Daten vor der räumlichen Verarbeitung, wie unter Filter and Refine (siehe Seite 52) beschrieben, um unnötige Berechnungen räumlicher Operatoren zu eliminieren.
- Das DBMS optimiert anschließend die verbleibende, rein alphanumerische Anfrage mit dem ihm zur Verfügung stehenden Mitteln.

Filter and Refine

Enthält der Bedingungsteil eines GQL-Kommandos UND-verknüpfte räumliche und alphanumerische Bedingungen, so werden aus allen Elementen diejenigen als Zwischenergebnis gefiltert, welche die alphanumerische Bedingung erfüllen (Filter-Schritt). Das Zwischenergebnis wird mit der räumlichen Bedingung verfeinert und liefert das Endergebnis (Refine-Schritt), siehe Abb. 9.

Der Filter-Schritt wird über einen Cursor realisiert und ist somit nur effektiv, wenn die Elemente einer Tabelle und nicht dem gesamten Plangebiet entnommen werden. Der Refine-Schritt wird in einer Schleife (Loop) über dem Cursor ausgeführt.

Dies ermöglicht dem Anwender, die Antwortzeit für eine Abfrage zu reduzieren, wenn er zusätzliche Sachbedingungen angeben kann, die mit der räumlichen Bedingung durch eine Obermengeneigenschaft verknüpft sind.

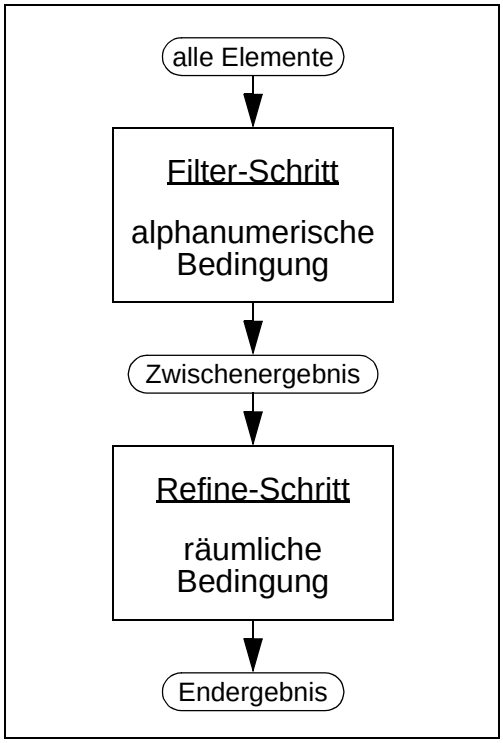


Abb. 9: Anwendung von Filter & Refine bei der Abarbeitung von GQL-Anfragen

6. Tests und Ergebnisse

Tests mit synthetischen und realen Geodaten überprüfen die Einsatzfähigkeit der vorgestellten Anfragesprache. Es geht bei diesen Tests darum, nicht nur die reine Performance von ausgewählten Operationen auf speziell darauf zugeschnittenen Datensätzen zu messen, sondern auch umfassend zu untersuchen, wie die Sprache aus der Perspektive eines GIS-Anwenders zur Lösung seiner Probleme eingesetzt werden kann. Dabei haben sowohl synthetische als auch anwendungsbezogene Tests ihre Berechtigung.

Tests mit synthetischen Daten werden in der Forschung[2] verwendet, um z. B. die Tragfähigkeit einer neuen Zugriffsmethode zu zeigen. Sie versuchen die Bedingungen in der Realität hinreichend genau abzubilden, um daraus Aussagen über die Einsatzfähigkeit in der Praxis abzuleiten. Dies wird durch einige Eigenschaften erschwert, die reale Daten im Gegensatz zu synthetischen Daten aufweisen:

1. **Qualität**: Die Fehler, die durch Ungenauigkeiten bei der Eingabe oder durch Maßstabsunterschiede in den realen Daten bedingt sind, treten in synthetischen Daten nicht auf.
2. **Komplexität**: Synthetische Daten beschränken sich häufig auf wenige Datentypen und erreichen dadurch nicht die Komplexität von realen Daten.
3. **Quantität**: GIS operieren typischerweise auf sehr großen Datenmengen im Bereich von einigen Gigabyte bis ein Terabyte. Dies stellt besondere Anforderungen an die Anfrageverarbeitung, die Zugriffsmethoden und die Optimierung.

Während sich die Quantität einfach künstlich erhöhen läßt, liefern nur reale Anwenderdaten Aussagen über die Probleme, mit denen Anwender aufgrund der Fehler in den Daten und der Komplexität kämpfen. Aus diesen Gründen wird GQL auf drei Arten getestet:

1. Der systematische Funktionalitätstest überprüft, ob die Implementierung korrekt ist.
2. Der Sequoia 2000 Benchmark testet die reine Datenbank-Performance des Geodatenservers mit GQL und macht sie mit der anderer räumlicher Datenbanken und Spracherweiterungen vergleichbar.
3. Zwei Anwenderszenarien bringen die Anforderungen realer Anwender ein und stellen sicher, daß die Sprache in ihrer Gesamtfunktionalität diesen Anforderungen entspricht.

Die Tests laufen auf SGI Workstations unter IRIX 5.3 und PCs unter NT 4.0 mit Datenbanken von Informix und Oracle.

2. Weitere Literatur zum Einsatz von Testdaten für GIS findet man bei [Hoop et. al. 93] [David et. al. 93] [Karimi 96] [Wellar 95] [Zwart & Greener 94] [Abel 96].

6.1 Tests mit synthetischen Daten

GQL wird zwei Tests mit synthetischen Daten unterzogen: Der systematische Funktionalitätstest (siehe Kapitel 6.1.1 und Abb. 10) überprüft, ob die Operatoren funktional korrekt implementiert sind. Der Sequoia 2000 Benchmark (6.1.2) macht Aussagen darüber, wie lange der Datenbankanteil typischer Anfragen dauert.

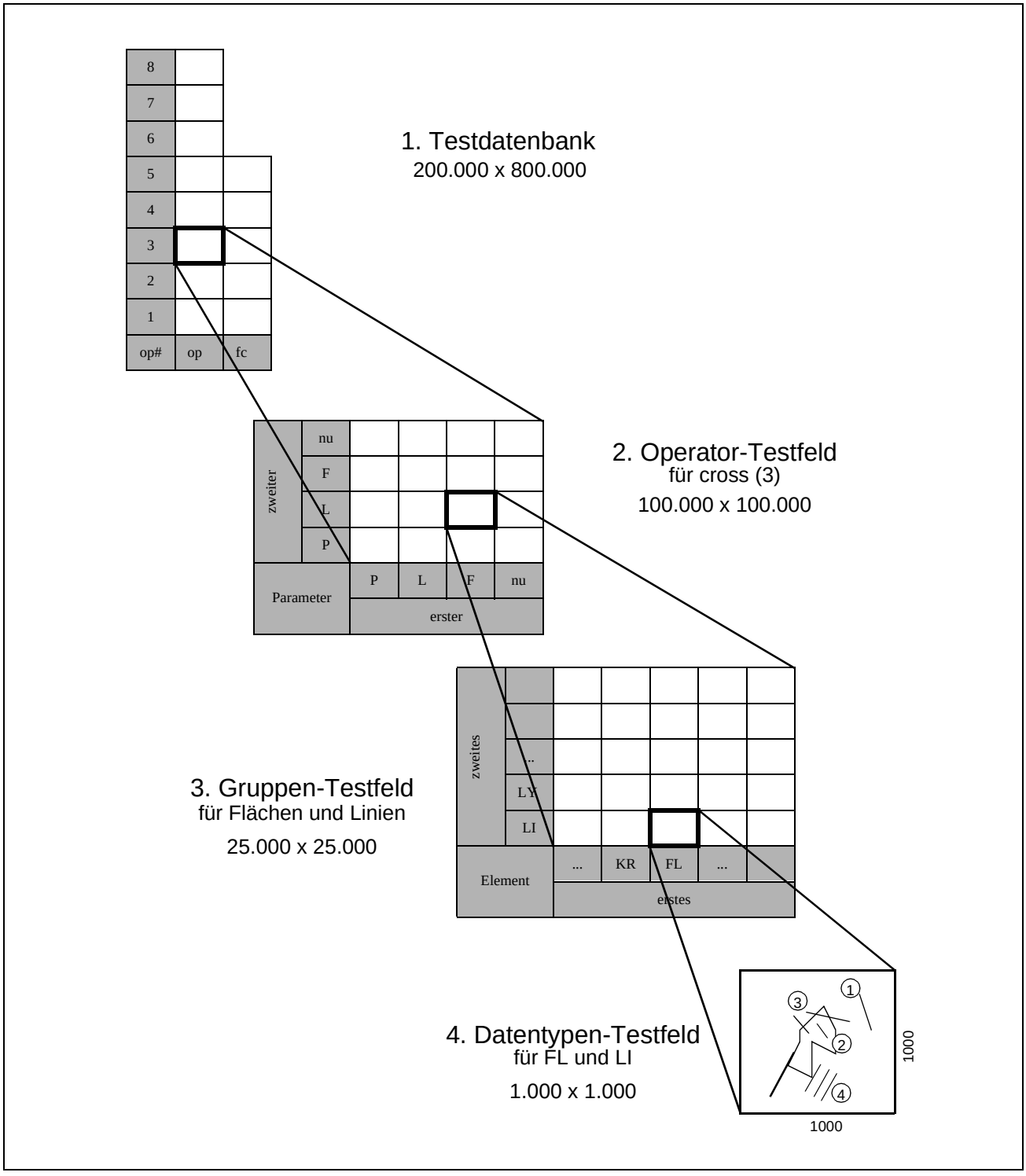


Abb. 10: Aufbau der Testdatenbank für den systematischen Funktionalitätstest

6.1.1 Systematischer Funktionalitätstest

Der Systematische Funktionalitätstest überprüft systematisch für alle Operatoren, ob sie mit allen möglichen Eingabeparametern verträglich sind und korrekt funktionieren. Er legt eine Testdatenbank an, füllt sie mit Testdaten und führt Testroutinen darauf aus.

Aufbau der Testdatenbank

Abbildung 10 zeigt den Aufbau der Testdatenbank in vier Stufen, die von oben nach unten immer weiter ins Detail gehen. Die weißen Kästchen entsprechen einem Bereich der Testdatenbank, die grau hinterlegten Bereiche dienen nur der Bezeichnung und Orientierung.

Stufe 1 zeigt die **Testdatenbank** in ihrer gesamten Ausdehnung von 200.000 x 800.000 Einheiten. Dies entspricht zwei Bereichen von je 100.000 Einheiten Breite für den Test von Operatoren (op) und Funktionen (fc). Innerhalb einer Spalte gibt es für jede/n Operator/Funktion ein Operator-Testfeld. Die Felder sind nach der Reihenfolge der Operatoren in Abschnitt 5.1.4 durchnumeriert (op#).

Stufe 2 zeigt die weitere Untergliederung eines **Operator-Testfelds** am Beispiel des Operators cross (s. Seite 43). Es hat eine Größe von 100.000 x 100.000 Einheiten. Ein Operator hat zwei Parameter. Ein Parameter gehört einer der Gruppen Punkt (P), Linie (L) oder Fläche (F) an oder er wird nicht unterstützt (nu). Daraus ergeben sich 16 mögliche Kombinationen der 4 Gruppen. Jeder Kombination entspricht ein Gruppen-Testfeld.

Stufe 3 zeigt ein **Gruppen-Testfeld** der Größe 25.000 x 25.000 Einheiten für den Fall, bei dem der erste Parameter eine Fläche und der zweite eine Linie ist. Es gibt mehrere Datentypen des zugrundeliegenden GIS, die für Flächen und Linien in Frage kommen. Das Gruppen-Testfeld bietet Platz für die Kombination von je 25 verschiedenen Datentypen als ersten und zweiten Eingabeparameter, die Abbildung zeigt der Übersicht halber nur je 5 an. In der Abbildung sind beispielhaft für Flächen die Datentypen FL und KR aufgeführt und für Linien die Datentypen LI und LY. Für jede Kombination von Datentypen gibt es ein Datentypen-Testfeld.

Stufe 4 zeigt ein **Datentypen-Testfeld** der Größe 1.000 x 1.000 Einheiten. In ihm sind die graphischen Elemente für die Datentypen, hier FL und LI, in der Testdatenbank so angeordnet, daß alle denkbaren Fälle von Beziehungen für diese beiden Elemente vorkommen. Im Beispiel gibt es Linien, welche völlig außerhalb ① oder völlig innerhalb ② einer Fläche liegen, welche die Fläche voll oder teilweise kreuzen ③ oder welche in verschiedenen Toleranzbereichen von der Fläche entfernt liegen ④.

Anhang A.1 beschreibt ausführlich, wie der Testdatenbestand aufgebaut wird und der Test durchgeführt wird. Seine Ergebnisse zeigen, daß GQL funktional den Anforderungen entspricht.

6.1.2 Sequoia 2000 Benchmark

Über die im vorherigen Abschnitt bestätigte, rein formal korrekte Abarbeitung einer Anfrage hinaus, spielt die Laufzeit bei größeren Datenmengen eine entscheidende Rolle. Diese läßt sich gut mit dem im folgenden vorgestellten Sequoia 2000 Benchmark messen.

Beschreibung

Das U.S. Global Change Research Program beobachtet die globale Klimaverschiebung. Dabei fallen Datenmengen in der Größenordnung von 100en von Terabytes an, die gespeichert, organisiert, zugegriffen, verteilt, visualisiert und analysiert werden müssen. Dies überfordert die verfügbaren Speichersysteme, Netzwerke, Visualisierungswerkzeuge und Datenbanken. Der im Rahmen des Sequoia 2000 Projekts der Universität von Kalifornien entwickelte Benchmark [Stonebraker 92] macht die Datenbankanforderungen meß- und vergleichbar.

Art der Daten	Datenmenge nach Gebietsgröße		
	regional	national	global
Punkt	1,8 MB[a]	27,5 MB	k. A.
Polygon	19,1 MB[b]	286 MB	k. A.
Gerichteter Graph	47,8 MB	1.100 MB	k. A.
Raster	1.064,0 MB	17.000 MB	2.000.000 MB
Gesamt-menge	~ 1.133 MB	~ 18.414 MB	~ 2.000.000 MB

Tab. 14: Die Datenmengen des Sequoia 2000 Benchmarks nach Art der Daten und Ausdehnung des Gebiets

a. 76584 Punkte x (4+4 byte Koord. + ∅16 byte Name)
b. 93607 Polygone mit ∅50 Koord. + 4 byte Landuse

Der Sequoia 2000 Benchmark (S2K-Benchmark) faßt die charakteristischen Anforderungen von Geowissenschaftlern an Datenbanksysteme zusammen: große Datenmengen, komplexe Datentypen und anspruchsvolle Suchprobleme. Darauf sind die **Daten** und Anfragen ausgerichtet. Es gibt Raster-, Polygon-, Punkt- und Graphdaten von lokaler, regionaler, nationaler und globaler Ausdehnung (siehe Tabelle 14).

Die **Modellierung** der Daten ist von geringer Komplexität. Für jede der dargestellten Datenarten existiert genau eine Tabelle (s. Abb. 11). Sie enthalten in einer Spalte die Geodaten (location, segment) und zusätzlich

Sachdatenspalten. Außer der räumlichen Beziehung gibt es in der Modellierung keine weiteren Beziehungen zwischen den Daten.

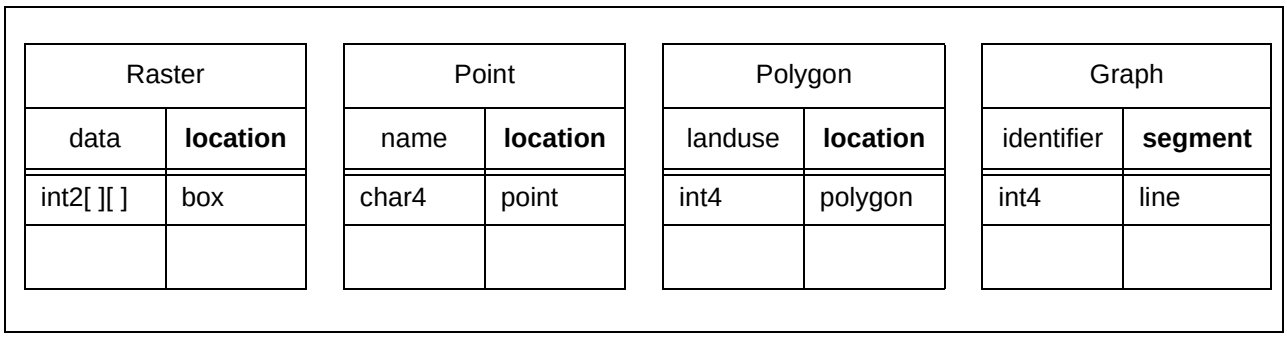


Abb. 11: Die Tabellen Raster, Point, Polygon und Graph des Sequoia 2000 Benchmarks

Der Benchmark besteht aus elf **Anfragen** in den fünf Gruppen: Daten laden, Rasteranfragen, Polygon- und Punktanfragen, räumlicher Verbund (spatial join) und Rekursion. (siehe Tabelle 15).

Gruppe	Anfrage	Test
Daten laden	1. Erzeugen der Datenbank, Laden der Daten, Erzeugen der Indizes	X
Raster-anfragen	2. Auswahl nach räumlichen, zeitlichen und sonstigen Kriterien	
	3. Anwendung einer Aggregatfunktion	
	4. Veränderung von Daten (mit Erzeugung einer neuen Tabelle)	
Einfache Punkt- und Polygonan-fragen	5. nicht-räumliche Selektion: Suche einen Punkt mit einem Namen n	X
	6. räumliche Selektion (mit Erzeugung einer neuen Tabelle): Suche alle Polygone, die sich mit einem Rechteck r überschneiden.	X
	7. Kombination räumlicher Kriterien: Suche alle Polygone, innerhalb des Kreises k und einer Fläche größer x	X
Räumlicher Verbund (spatial join)	8. Verbund zwischen Punkt- und Polygondaten: Gib die Landnutzung (landuse) aller Polygone innerhalb eines Rechtecks von 50 km um einen Punkt p aus	X
	9. Verbund zwischen Raster- und Polygondaten	
	10. Verbund zwischen Punkt- und Polygondaten (mit Erzeugen einer Tabelle): Ermittle die Namen aller Punkte, die innerhalb von Polygonen mit Landnutzung (landuse) liegen und lege sie in einer Tabelle ab	X
Rekursion	11. Lösen eines Erreichbarkeitsproblems in einem Graph	

Tab. 15: Zusammenfassung der elf Anfragen des Sequoia 2000 Benchmarks in fünf Gruppen

Durchführung

Mit der Sprache GQL und dem System SICAD GDS werden die Daten geladen und die Anfragen 5, 6, 7, 8 und 10 durchgeführt. Rasteranfragen und Rekursion können mit GQL nicht

bearbeitet werden. Aufgrund von Plattenplatzbeschränkungen wurde der regionale Benchmark benutzt. Die GQL betreffenden Ausschnitte der Testprozeduren werden im folgenden kurz aufgeführt. Die kompletten Prozeduren können in Anhang A.2 eingesehen werden.

ad 1. Erzeugen der Datenbank, Laden der Daten, Erzeugen der Indizes

Zum Laden der Daten werden keine GQL-Kommandos benötigt. Die Daten liegen in einem textuellen Format vor, bei dem eine Zeile einem Datensatz einer der Tabellen aus Abbildung 11 entspricht. Diese Darstellung wird in das SQD-Format[3] übersetzt und in mehrere Einheiten zerlegt, so daß sie von der Geodatenbank mit dem Kommando gbqgel gelesen werden können.

ad 5. nicht-räumliche Selektion: Suche einen Punkt mit einem Namen n

Gesucht ist ein Punkt mit einem beliebig gewählten Namen. Der Name wird in der Prozedur zufällig ausgewählt und in der Variable %t52 (4) abgelegt. Die Namen der Punkte sind in einem Deskriptor bei der Geometrie gespeichert und werden mit der Funktion value_of abgefragt (s. Abb. 12).

```
1  INSERT   INTO elementset
2  SELECT   location
3  FROM     point
4  WHERE    value_of(location.txt) = %t52
```

Abb. 12: GQL-Lösung für Anfrage 5 aus Tabelle 15

Alternativ kann der Punkt auch durch eine reine SQL-Abfrage gefunden werden, da der Name auch in der Tabelle point gespeichert ist (s. Abb. 13).

```
1  INSERT   INTO elementset
2  SELECT   location
3  FROM     point
4  WHERE    name = %t52
```

Abb. 13: Geodatenbank-Lösung für Anfrage 5 aus Tabelle 15

ad 6. räumliche Selektion (mit Erzeugung einer neuen Tabelle): Suche alle Polygone, die sich mit einem Rechteck r überschneiden.

Gesucht sind alle Polygone, die sich mit einem Suchrechteck überschneiden. Die Koordinaten des Rechtecks werden in der Prozedur festgelegt (könnten aber ebensogut von einem Benutzer durch Aufziehen eines Rechtecks am Bildschirm bestimmt werden) und in den Variablen %i61 bis %i64 abgelegt. Mittels des Konstruktors rectangle ist es möglich, das Rechteck in die GQL-Anfrage einzubinden und mit dem Operator intersect mit den in der Tabelle poly gespeicherten Polygonen zu vergleichen (s. Abb. 14).

```
1  INSERT   INTO elementset
2  SELECT   location
3  FROM     poly
4  WHERE    intersect(rectangle(%i61 %i62 %i63 %i64), poly.location)
```

Abb. 14: GQL-Lösung für Anfrage 6 aus Tabelle 15

3. Ein textuelles proprietäres Austauschformat der verwendeten Geodatenbank.

Alternativ kann der Befehl gbsrt der Geodatenbank verwendet werden (s. Abb. 15). Dazu müssen zuerst die Eckkoordinaten des Rechtecks in Zeichenketten umgewandelt werden, damit sie verwendet werden können (1). Aus den einzelnen Texten kann die Suchbedingung zusammengesetzt werden (2). Zu beachten ist allerdings, daß die Suche mit gbsrt die Elemente in allen Zellen liefert, die von dem Suchrechteck geschnitten werden. Das ist eine Obermenge der eigentlichen Treffermenge.

```
1   SET     %t1 cvt %i61
    SET     %t2 cvt %i62
    SET     %t3 cvt %i63
    SET     %t4 cvt %i64

2   GBSRT   fln 'X>'//%t1//',X<'//%t3//',Y>'//%t2//',Y<'//%t4 0 0 1100 1400
```

Abb. 15: Geodatenbank-Lösung für Anfrage 6 aus Tabelle 15

ad 7. Kombination räumlicher Kriterien: Suche alle Polygone, innerhalb des Kreises k und einer Fläche größer x

Gesucht sind alle Polygone aus Tabelle poly, die sowohl innerhalb eines durch zwei Punkte definierten Kreises (4) liegen, als auch einen Flächeninhalt von mehr als 1 km^2 haben. Ähnlich wie in der vorherigen Abfrage wird der Konstruktor circle verwendet, um den Kreis innerhalb von GQL zur Verfügung zu stellen (s. Abb. 16). Da die Flächen vollständig enthalten sein sollen, wird diesmal der Operator contains verwendet. Der Flächeninhalt wird mit der Funktion area ermittelt.

```
1   INSERT  INTO elementset
2   SELECT  location
3   FROM    poly
4   WHERE   contains(circle(%i71 %i72 %i73 %i74), poly.location)
5   AND     area(poly.location) > 1000000
```

Abb. 16: GQL-Lösung für Anfrage 7 aus Tabelle 15

Hier existiert keine alternative Lösung mit den Mitteln der Geodatenbank, da diese weder innerhalb einer Anfrage den Flächeninhalt berechnen kann (area), noch kreisförmige Suchgebiete zur Verfügung stehen.

ad 8. Verbund zwischen Punkt- und Polygondaten: Gib die Landnutzung (landuse) aller Polygone innerhalb eines Rechtecks von 50 km um einen Punkt p aus

Gesucht sind alle Polygone, die innerhalb eines Rechtecks von 50 km um einen gewählten Punkt liegen. Ein solches Rechteck kann nicht in-

```
1   CREATE TABLE q8temp (landuse char(5), location char(16))

2   INSERT  INTO q8temp
3   SELECT  landuse, location
4   FROM    poly
5   WHERE   contains(rectangle(%i81 %i82 %i83 %i84), poly.location)
```

Abb. 17: GQL-Lösung für Anfrage 8 aus Tabelle 15

nerhalb von GQL berechnet werden, es muß also in der Prozedur geschehen. Liegen die Koordinaten des Rechtecks vor, so lautet die GQL-Anfrage wie in Abb. 17 angegeben.

Alternativ kann hier ähnlich wie in Anfrage 6 gbsrt verwendet werden, um die Elemente in dem gewünschten Ausschnitt in die Elementmenge zu bringen (s. Abb. 18). Es gelten allerdings die selben Einschränkungen wie dort. Mit einem SQL-Kommando können dann für die in der Elementmenge befindlichen Elemente die Werte für landuse und location ausgegeben werden.

```
1  GBSRT    fln ’X>’//%t1//’,X<’//%t3//’,Y>’//%t2//’,Y<’//%t4 0 0 1100 1400

2  SELECT   landuse, location
3  FROM     poly
4  WHERE    location IN (SELECT location FROM elementset)
```

Abb. 18: Geodatenbank-Lösung für Anfrage 8 aus Tabelle 15

ad 10. Verbund zwischen Punkt- und Polygondaten (mit Erzeugen einer Tabelle): Ermittle die Namen aller Punkte, die innerhalb von Polygonen mit Landnutzung (landuse) liegen und lege sie in einer Tabelle ab

Gesucht sind alle Punkte die innerhalb eines Polygons liegen, das eine bestimmte Landnutzung hat. Die Landnutzung wird aus einer Tabelle mit zufällig generierten Werten ausgewählt und in der Variable %t22 gespeichert (s. Abb. 19). Dort steht sie für die Abfrage zur Verfügung (6). Der Operator contains findet alle Punkte innerhalb dieser ausgewählten Polygone.

```
1  CREATE TABLE q10temp (name char(70))

2  INSERT   INTO q10temp
3  SELECT   name
4  FROM     point, poly
5  WHERE    contains(poly.location, point.location)
6  AND      poly.landuse = %t22
```

Abb. 19: GQL-Lösung für Anfrage 10 aus Tabelle 15

Die Geodatenbank kann hier keine alternative Lösung bieten.

Ergebnisse und Wertung

Die Laufzeit der Anfragen faßt Tabelle 16 zusammen. Die lange Dauer für das Laden der Geodaten liegt daran, daß die Geodatenbank topologische Datenstrukturen über die gelesenen Daten aufbaut und mit speichert.

Gruppe	Nr.	Anfrage	Laufzeit/sec	
			GDS	GQL
Daten laden	1	Laden	62000	-
Einfache Punkt- und Polygonanfragen	5	nicht-räumliche Selektion	1	3[a]
	6	räumliche Selektion	2374	2466[b]
	7	kombinierte Selektion	n[c]	5286
Räumlicher Verbund (spatial join)	8	Punkt- und Polygondaten mit oder ohne Erzeugen einer Tabelle	2685	2532
	10		n	3140

Tab. 16: Die Ergebnisse des Sequoia 2000 Benchmarks mit GQL

a. Längere Laufzeit, da die an den Geodaten gespeicherten Attribute durchsucht werden.
b. Die Filterfunktion von GQL beruht auf denselben Mechanismen wie gbsrt. Vergleichbare Ergebnisse sind deshalb zu erwarten.
c. n = Anfrage mit Geodatenbankmitteln nicht möglich.

Bei einfachen räumlichen Selektionen ist GQL von der Performance mit der Lösung in der Geodatenbank vergleichbar, aber freundlicher in der Bedienung.

Werden allerdings mehrere Bedingungen in einer Anfrage kombiniert, so steigt die Laufzeit stark an. Ein Vergleich mit der Geodatenbank kann hier nicht gezogen werden, da sie diese Operation nicht durchführen kann.

Der S2K-Benchmark zeigt Mängel in der Datenbank-Performance von GQL. Für die Benutzbarkeit mit realen Anwenderdaten spielen aber auch noch andere Faktoren eine Rolle, die mit den im nächsten Abschnitt vorgestellten Anwenderszenarien verdeutlicht werden.

6.2 Anwenderszenarien mit realen Daten und Anfragen

Benchmarks wie der Sequoia 2000 Benchmark eignen sich gut, um die reine DB-Performance zu überprüfen. Die Leistung, die der Anwender wahrnimmt, enthält aber zusätzlich die folgenden Komponenten: Visualisierung der Ergebnisse, Erlernbarkeit der Sprache und Anwendbarkeit im Gesamtkontext. Somit kann ein Gesamteindruck besser mit Anwenderszenarien überprüft werden.

Die Arbeit verwendet zwei zusammen mit ausgewählten Anwendern erarbeitete Szenarien: die Forstbetriebskarte Bayern (FBK), welche die Kartographische Anstalt mit einem Forst-GIS im Auftrag des Bayerischen Staatsministeriums für Ernährung, Landwirtschaft und Forsten erstellt (6.2.1), und die Digitale Stadtgrundkarte der Stadt München (6.2.2). Ein Szenario besteht jeweils aus Daten und Anfragen, die für einen Anwender typisch sind.

6.2.1 Forst-GIS der Kartographischen Anstalt

Beschreibung

Die Kartographische Anstalt (KA) des Bayerischen Staatsministeriums für Ernährung, Landwirtschaft und Forsten (BStMELF) hat u. a. die Aufgabe, die Forstämter mit Karten zu versorgen. Um die Herstellung dieser Karten zu vereinfachen, wurde das Forst-GIS aufgebaut. Es wird eingesetzt, um die Daten zu digitalisieren, die Produktion der Waldkarten zu automatisieren und Auswertungen auf den Daten durchzuführen. Dazu gehört z. B. die Flächeninhalte von Flächen mit bestimmten Eigenschaften zu berechnen oder thematische Karten zu erstellen.

Die **Daten** liegen als ca. 200 MB an Vektordaten (Polygone) und 2,2 MB attributiv verknüpfte Sachdaten im SQD-Format, einem proprietären ASCII-Format, vor. Rasterdaten sind nicht enthalten. Vom Gesamtgebiet Bayern wurde eine Fläche von 41 x 45 km^2 geladen und Ausschnitte davon in den Anfragen bearbeitet.

Die Daten der *Forstbetriebskarte* sind in einer dreistufigen Hierarchie aus 158.144 Punkten, 79.907 Linien und 9.096 Flächen aufgebaut. Die Linien sind einfache Linien (7.190) und Linienzüge mit gerader (17.353) oder bogenförmiger (55.364) Interpolation. Die Flächen sind mit Sachsätzen der Tabelle OR (s. Abb. 20) verknüpft. Außerdem gibt es die *Waldkarte* mit 2.039 Flächen, 16.257 Linien und 32.166 Punkten und eine *Zusatzkarte*, die u. a. 7.265 Symbole, 292 Texte und weitere Flächen und Linien enthält. Die Daten sind nur eingeschränkt mit denen des Sequoia 2000 Benchmarks vergleichbar, weil der Anteil an Sachdaten höher ist und die Geodaten mit diesen mehrfach verknüpft sind. Die Anzahl dieser Verknüpfungen erhöht nicht nur den Speicherbedarf für die Daten sondern auch die Komplexität bei der Formulierung und Verarbeitung der Anfragen, wie später noch zu sehen sein wird.

Die **Modellierung** der Sachdaten besteht aus den 7 Tabellen AS, BK, FU, HB, OR, TF und VZ. Von diesen sind die Tabellen OR (3.211 Zeilen) und VZ (165 Zeilen) für die späteren Untersuchungen interessant. Die Tabelle OR ordnet einen Bestand in die Hierarchie der Forstverwaltung ein und vergibt einen Bestandsschlüssel. Die Tabelle VZ enthält Zusatzinformationen wie z. B. die Baumart. Die Einträge der Tabelle OR sind über das Jahr und den Bestandsschlüssel mit den Bestandsflächen in der Karte verknüpft. Es handelt sich um eine n:m Beziehung. Sie ist über die Tabelle ORI_J realisiert, die über die Spalte location den Bezug zu den Geodaten herstellt (s. Abb. 20).

OR (Bestand)	
Spalte	Bedeutung
lfzfe	Jahr
bestkey	Bestandsschlüssel
vondat	Gültigkeitsbereich
bisdat	des Datensatzes
ofod	Oberforstdirektion
foa	Forstamt
fodst	Forstdienststelle
dis	Distrikt
abt	Abteilung
uabt	Unterabteilung
best	Bestand
bfl	Bestandsfläche
bkl	Bestandsklasse
wugn	zusätzliche
wbzn	Angaben

VZ (Zusatzinformation)	
Spalte	Bedeutung
lfzfe	Jahr
bestkey	Bestandsschlüssel
bestlfd	...
bafe	Baumart
vzh	zusätzliche
vzn	Angaben

ORI_J (Verknüpfung)	
Spalte	Bedeutung
lfzfe	Jahr
bestkey	Bestandsschlüssel
location	Element-ID

Abb. 20: Die drei Tabellen OR, ORI_J und VZ der Forstbetriebskarte Bayern

Die untersuchten **Anfragen** erstrecken sich typischerweise auf einen **lokalen** Ausschnitt, der bereits zur Darstellung in das GIS geladen ist. Sie beziehen sich auf **interaktiv** am Bildschirm ausgewählte Elemente und liefern dazu zwei Arten von Ergebnissen: Entweder sie heben graphisch Elemente hervor, die zu den ausgewählten Elementen in einer bestimmten Beziehung stehen, die z. B. angrenzend oder in einer bestimmten Entfernung liegen, oder sie listen mit räumlichen Funktionen berechnete Informationen zu den ausgewählten Elementen tabellarisch auf. Die Anzahl zu verarbeitender Elemente ist dabei durch den gewählten Ausschnitt begrenzt. Tabelle 17 gibt einen Überblick über die Anfragen und ordnet sie ähnlich wie beim Sequoia 2000 Benchmark in Gruppen ein. Sie zeigt welche Daten (Sachdaten und Geodaten, dort Flächen und Linien) verarbeitet werden und gibt an, ob das Ergebnis tabellarisch (T) oder graphisch (H, highlight, in einer schon dargestellten Karte graphisch hervorgehoben) dargestellt wird. Zusätzlich führt sie die Methoden auf, die zur Berechnung des Ergebnisses beitragen.

Gruppe	Sach	Geo	Anfrage	Ergebnis		Methoden
räumliche Selektion		F	1. Große Flächen finden	H		Geo-Selektion (area)
räumliche Projektion	S	F	2. Flächenumfang bestimmter Flächen		T	Sach-Selektion und Geo-Projektion (perimeter)
räumliche Aggregation	S	F	3. Gesamtfläche berechnen		T	Summe über Area()
	S	F, L	4. Länge von Waldwegen berechnen		T	Summe Len(), meets
räumlicher Verbund (spatial join)	S	F	5. Fichtenbestände benachbart zum Waldweg	H		touch + Sach-Selektion
	S	F	6. Flächen in Entfernung x vom Waldweg	H		distance, Puffer
	S	F, L	7. Bestände am Wanderweg	H		within

Tab. 17: Typische Anfragen auf der Forstbetriebskarte Bayern. Die Einteilung in Gruppen folgt teilweise der Einteilung des Sequoia 2000 Benchmarks (siehe Tabelle 15). Die Anfragen unterscheiden sich weiter durch ihre Eingabedaten: Sach- (S) und Geodaten (G), Flächen (F) und Linien (L), und nach ihrem Ergebnis (E): hervorgehoben in der Graphik (H) oder tabellarisch (T).

Durchführung

Wie oben erwähnt, führt der Anwender die Anfragen interaktiv auf einem lokalen Ausschnitt durch. Daraus resultiert eine besondere Art des Vorgehens, die bei allen Anfragen zum Tragen kommt: Der Anwender manipuliert die graphischen Elemente am Bildschirm mit den ihm vertrauten GIS-Werkzeugen. Mit ihnen bereitet er die GQL-Anfrage vor und stellt die Ergebnisse der Anfrage dar. Die Anfrage wird insgesamt also in drei Schritten durchgeführt.

1. Vorbereitung der Eingabedaten der Anfrage
   Der Anwender selektiert die zu untersuchenden Elemente graphisch am Bildschirm und überträgt ihre Identifier (location) in die Elementmengen mit den Namen picked, flaechen, linien oder punkte. Die Elementmengen sind als Tabellen in der Datenbank realisiert. Dadurch kann GQL bei der Abarbeitung von Anfragen darauf zugreifen.

2. Ausführung der GQL-Anfrage
   Die Geodatenbank führt die GQL-Anfrage aus und speichert die IDs der gefundenen Elemente in der Tabelle result.

3. Graphische Darstellung
   Der Anwender überträgt die IDs der von GQL gefundenen Elemente zurück in das GIS und stellt sie mit traditionellen GIS-Werkzeugen dar.

Bei den im folgenden geschilderten Lösungen der einzelnen Anfragen wird im Wesentlichen auf den mittleren Teil, die GQL-Anfrage, eingegangen. Bei einzelnen Anfragen wird zur Illustration das graphische Ergebnis mit dargestellt.

ad 1. Große Flächen finden

Gesucht sind alle Flächen des gewählten Ausschnitts (f), deren Flächeninhalt größer als 80.000 ist. Diese Information kann zum einen über die GQL-Funktion area berechnet werden, zum anderen liegt sie aber auch vorberechnet am Element gespeichert vor und kann mit der GQL-Funktion value_of abgefragt werden (s. Abb. 21). Zu den Laufzeitunterschieden siehe Ergebnistabelle 18.

```
1  INSERT    INTO result
2  SELECT    f.location
3  FROM      flaechen f
4  WHERE     area(f.location) > 80000
5
6  oder
7  ...
8  WHERE     value_of(f.location, FLA) > 80000
```

Abb. 21: GQL-Lösung für Anfrage 1. "Große Flächen finden" aus Tabelle 17

ad 2. Flächenumfang bestimmter Flächen

Gesucht sind alle Flächen im Jugendstadium mit einem Mortalstadiumsanteil von mehr als 30 %. Die Flächen sind nach diesen Kriterien bereits im Datenmodell des Forst-GIS Bayern vorklassifiziert. Die genannte Kombination entspricht einem Feature-Code (fco) von 517 und kann deshalb mit GQL direkt über value_of abgefragt werden (s. Abb. 22). Dies zeigt die Bedeutung der Integration von GQL mit den bisherigen Möglichkeiten des GIS, die es dem Anwender ermöglicht seine bisherigen Daten weiterhin zu nutzen.

```
1  SELECT    value_of(f.location, TXT),
             perimeter(f.location)
2  FROM      flaechen f
3  WHERE     value_of(f.location, FCO) = 517
```

Abb. 22: GQL-Lösung für Anfrage 2. "Flächenumfang bestimmter Flächen" aus Tabelle 17

ad 3. Gesamtfläche berechnen

Gesucht ist der Gesamtflächeninhalt der Flächen aller Bestände einer Abteilung (abt) eines Distriktes (dis). Die Flächeninhalte werden über die Funktion area berechnet und mit sum aufsummiert und nach Beständen (bestkey) gruppiert tabellarisch ausgegeben (s. Abb. 23).

```
1  SELECT    o.bestkey, sum(area(oj.location))
2  FROM      ori o, ori_j oj
3  WHERE     o.dis=20 AND o.abt=6
4  AND       o.bestkey = oj.bestkey
5  GROUP BY  o.bestkey
```

Abb. 23: GQL-Lösung für Anfrage 3. "Gesamtfläche berechnen" aus Tabelle 17

ad 4. Länge von Waldwegen berechnen

Gesucht sind für alle Abteilungen (dis, abt) die Längen der darin liegenden Waldwege. Waldwege sind als einzelne Flächenstücke (f) digitalisiert, die von Linien (splines s) begrenzt werden. Die Waldwegflächen unterscheiden sich durch ihre weiße Farbe (st=29) von den anderen Waldflächen (s. Abb. 25).

```
1  SELECT    o.dis, o.abt, sum(len(s.location))/2
2  FROM      flaechen f, splines s, ori o, ori_j oj
3  WHERE     meets(s.location, f.location)
4  AND       value_of(f.location, ST) = 29
5  AND       f.location = oj.location
6  AND       oj.bestkey = o.bestkey
7  GROUP BY  o.dis, o.abt
```

Abb. 24: GQL-Lösung für Anfrage 4. "Länge von Waldwegen berechnen" aus Tabelle 17

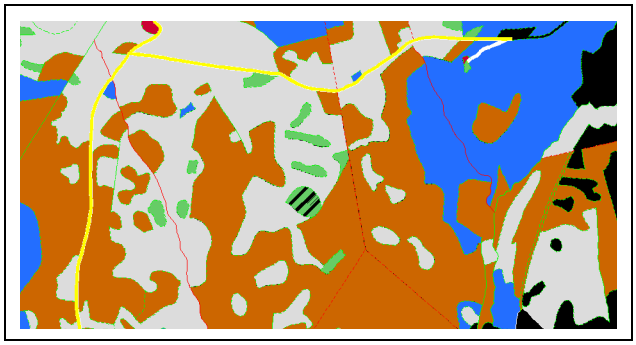


Abb. 25: Graphische Darstellung der selektierten Waldwege

Die Gesamtlänge ergibt sich als die Hälfte der Summe der Längen der linken und rechten Begrenzungslinien (s. Abb. 24). Hier ist gut zu erkennen, wie die Modellierung die späteren Abfragemöglichkeiten beeinflußt. Wird ein Waldweg als Fläche und nicht als Linie modelliert, so ist die Bestimmung seiner Länge nicht einfach durch den Operator len möglich, da dieser nicht auf Flächen angewendet werden kann.

ad 5. Fichtenbestände benachbart zum Waldweg

Gesucht sind die Fichtenbestände entlang eines Waldwegs, also die Flächen in einem bestimmten Ausschnitt (f), die benachbart (meets) zu einer ausgewählten Wegfläche (p) liegen (s. Abb. 26). Die Information über die Baumart (bafe) liegt in der Tabelle vz verschlüsselt vor (Fichte = 10).

```
1  INSERT    INTO result
2  SELECT    f.location
3  FROM      picked p, flaechen f, ori_j oj, vz v
4  WHERE     meets(p.location, f.location)
5  AND       f.location = oj.location
6  AND       oj.bestkey = v.bestkey
7  AND       v.bafe=10
```

Abb. 26: GQL-Lösung für Anfrage 5. "Fichtenbestände benachbart zum Waldweg" aus Tabelle 17

Abb. 27: Graphische Darstellung der selektierten Fichtenbestände

Abbildung 27 zeigt die durch die Abfrage ausgewählten Flächen in der Grafik. Die große Fläche unterhalb des Waldwegs wird wegen Problemen von GQL mit Flächen mit Löchern nicht selektiert.

ad 6. Flächen in Entfernung x vom Waldweg

Gesucht sind alle Flächen, die im Abstand von 100 m von einem gewählten Waldweg liegen. Die Lösung erfolgt mit der GQL-Funktion distance (s. Abb. 26).

```
1  INSERT    INTO result
2  SELECT    f.location
3  FROM      flaechen f, picked p
4  WHERE     distance(f.location, p.location) < 100
```

Abb. 28: GQL-Lösung für Anfrage 6. "Flächen in Entfernung x vom Waldweg" aus Tabelle 17

Abb. 29: Graphische Darstellung des selektierten Waldwegs

Der Waldweg wird im GIS selektiert (s. Abb. 29) eine Referenz auf seine Geometrie in Tabelle picked abgelegt. Die Ergebnisstabelle result wird graphisch dargestellt (s. Abb. 30).

Abb. 30: Graphische Darstellung der Bestände

ad 7. Bestände am Wanderweg

Gesucht sind alle Flächen (f), durch die ein Wanderweg führt (s. Abb. 31). Das Problem liegt hier weniger in der Formulierung der GQL-Anfrage, sondern in der Ermittlung der Linien (s), die den Wanderweg bilden.

```
1  INSERT    INTO result
2  SELECT    f.location
3  FROM      flaechen f, splines s
4  WHERE     within(f.location, s.location)
```

Abb. 31: GQL-Lösung für Anfrage 7. "Bestände am Wanderweg" aus Tabelle 17

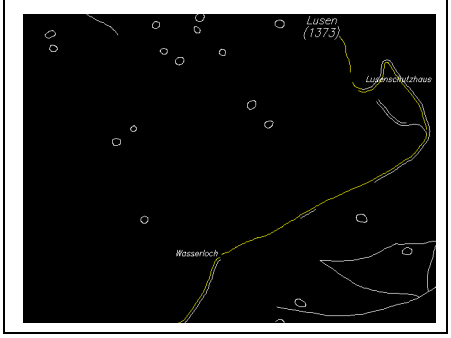


Abb. 32: Graphische Darstellung des unvollständig digitalisierten Wanderwegs

Diese sind nämlich nur insoweit digitalisiert, als sie auch auf der Karte sichtbar werden (s. Abb. 32). Laufen sie entlang der Grenze eines Bestandes (s. Abb. 33), hat man darauf verzichtet. Das ist ein weiteres Beispiel dafür, wie die aus-

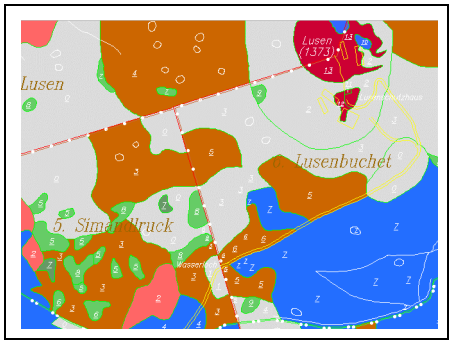


Abb. 33: Graphische Darstellung der Bestände

schließlich an der graphischen Gestaltung orientierte Modellierung der Daten eine Analyse mit Datenbankmitteln erschwert.

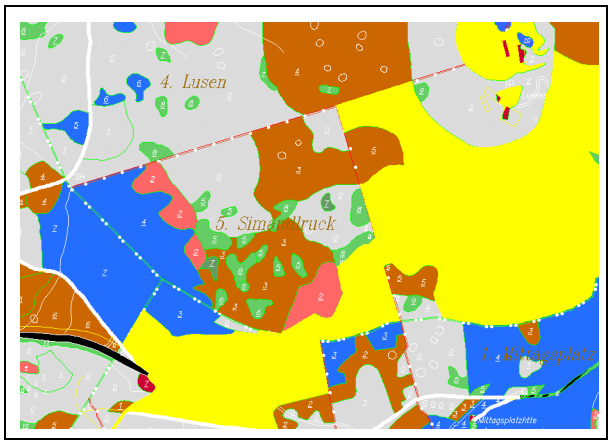


Abb. 34: Graphische Darstellung der Bestände am Wanderweg

Für die Abfrage mit GQL wurden die Wanderwege ausgewählt und in Tabelle splines gespeichert, ebenso die relevanten Bestände in Tabelle f. Das Ergebnis der Abfrage wird in der Tabelle result abgelegt. Die graphische Darstellung der Ergebnismenge aus Tabelle result ist in Abb. 34 zu sehen.

Alternative Berechnungsmöglichkeiten im GIS

Mit GQL kann der Anwender neue Wege bei der Analyse seiner Daten gehen. Daneben steht ihm aber weiterhin die volle Palette traditioneller Methoden zur Verfügung. Um vergleichen zu können, ob und wenn ja welche Vorteile GQL gegenüber diesen hat, werden hier einige der obigen Anfragen ausgewählt und ohne GQL nur mit den bisher schon vorhandenen Mitteln des GIS gelöst.

ad 1. Große Flächen finden (Lösung in der Geodatenbank)

Als Alternative zu der Lösung mit dem GQL-Operator value_of kann ein Befehl der Geodatenbank verwendet werden, der die Selektion unter Sachdatenbedingungen erlaubt (s. Abb. 35). Er ermöglicht die Formulierung von Bedingungen, die zu denen von GQL äquivalent sind. Daten die nicht als Deskriptoren am Element gespeichert sind, können allerdings nicht wie bei GQL (area) während der Abarbeitung der Anfrage berechnet werden.

```
1   ...
2   GBSRT FL 'FLA>80000' %P0 %P1

statt

3   ...
4   WHERE       value_of(f.location, FLA) > 80000
```

Abb. 35: Geodatenbank-Lösung für Anfrage 1. "Große Flächen finden" aus Tabelle 17

ad 5. Fichtenbestände benachbart zum Waldweg (Lösung im GIS-Frontend)

Die Lösung im GIS basiert auf der Kenntnis, das benachbarte Flächen gemeinsame Linien in der topologischen Datenstruktur des GIS haben (s. Abb. 36). Der Anwender wählt eine Fläche des Waldwegs aus (1) und merkt sie vor (2) um sie später aus der Ergebnismenge entfernen zu können (5). Die entscheidenden Schritte sind (3) und (4). Der Anwender folgt der topologischen Struktur vom Master (ausgewählte Fläche) zu ihren Details (Splines sn). Anschließend in umgekehrter Richtung von den Details (alle Splines, die den Rand der ausgewählten Fläche bilden) zu ihren Mastern. Da zusammenhängende Flächen sich ihre Begrenzungslinien teilen, sind nun alle benachbarten Flächen inklusive der Ausgangsfläche in der Treffermenge. Die Folge von Kommandos ist nicht länger als der GQL-Befehl. Die Ausführungszeit liegt unter 1 Sekunde pro Kommando bei interaktiver Ausführung. Um diesen Vorgang für mehrere Flächen durchzuführen empfiehlt es sich eine Prozedur zu schreiben. Um zusätzlich Sachdaten abzufragen wird wieder SQL benötigt.

1	supo fl	wähle Waldwegfläche
2	paranf buf=weg	gewählte Fläche merken
3	skks md fl va=1	navigiere zu den Linien
4	skks dm sn va=1	wieder zu den Flächen
5	seman -2 buf=weg va=1	ohne gewählte Fläche
6	dfn	neu anzeigen
7	dsel	selektierte Flächen

Abb. 36: GIS-Frontend-Lösung für Anfrage 5. "Fichtenbestände benachbart zum Waldweg" aus Tabelle 17

ad 5. Fichtenbestände benachbart zum Waldweg (Lösung in der Geodatenbank)

Diese Lösung basiert auf den gleichen Prinzipien der topologischen Speicherung wie oben. Im Unterschied dazu findet die Berechnung der Nachbarn aber nicht im GIS-Frontend statt, sondern in der Geodatenbank (s. Abb. 37). Dafür wird die selektierte Fläche (1) in eine Tabelle (Elementmenge) der Geodatenbank übertragen (2). Die GeoDB-Middleware berechnet die Nachbarn (3). Anschließend wird das Ergebnis zur Darstellung zurück ins Frontend transferiert (4). Diese Lösung hat Vorteile, wenn die Nachbarflächen zu Flächen gesucht werden, die räumlich weit verteilt sind. Sie müssen dann nicht jeweils einzeln ins Frontend geladen werden, sondern können in einem Schritt ermittelt werden.

1	supo fl	wähle Waldwegfläche
2	gbmgmg smem 0 0	übertrage sie in die DB
3	gbmgmg emem 0 0 va=1	finde ihre Nachbarn
4	gbmgmg emsm 0 0	übertrage sie in das GIS

Abb. 37: Geodatenbank-Lösung für Anfrage 5. "Fichtenbestände benachbart zum Waldweg" aus Tabelle 17

ad 6. Flächen in Entfernung x vom Waldweg
(Lösung mit GIS-Aufsatzpaket zur Flächenverschneidung)

Gesucht sind alle Flächen, die im Abstand von 100 m von einem gewählten Waldweg liegen. Die Lösung erfolgt mit einem GIS-Aufsatzpaket zur Flächenverschneidung in mehreren aufeinander aufbauenden interaktiven Schritten (s. Abb. 38): Um die spätere Berechnung zu beschleunigen, löscht der Anwender vorab Flächen, die er nicht zum Ergebnis rechnet (1). Um die gesuchten Flächen zu finden, erzeugt der Anwender einen Puffer (3) und verschneidet (5) ihn mit den Ausgangsflächen. Die Verschneidung setzt auf speziellen topologischen Datenstrukturen auf, die sowohl für die Ausgangsflächen (2), als auch für den Puffer (4) gebildet werden müssen.

Die Durchführung dieser Operationen erfordert detaillierte Kenntnisse sowohl der unterliegenden Datenstrukturen als auch des verwendeten Analysewerkzeugs. Eine intuitive Formulierung der Anfrage wie bei GQL ist nicht möglich.

Die Ausführung der Flächenverschneidung in Schritt 5 dominiert mit 300 Sekunden Laufzeit die Gesamtzeit der Anfrage.

```
1   Bildausschnitt vorbereiten
    sav via1
    old via1
    supo fl
    ls
    sav via2

2   Flächen und Linien für Verschneidung vorbereiten
    old via2
    semi; lose sn; lose li
    arconv tgt=ly
    lose fl
    archeck tgt=fl; arcorr tgt=fl res=1.0; archeck tgt=fl
    sav via3

    argbkey lay=2
    pas fl obj2='00'
    sav bestand3

3   Puffer erzeugen
    old via3
    op wid=100 hei=100
    supo fl
    arbuffer
    semi; lose fl bed='nam.ne.buffer'; ls
    dfn; sav buffer3

4   Puffer für Verschneidung vorbereiten
    old buffer3
    op res=1.0
    semi; lose fl
    archeck ...; arcorr ...
    pas fl obj1='00'
    argbkey lay=1
    sav buffer4

5   Flächenverschneidung
    old bestand3
    semi; lose fl
    arcomv 2 best
    add buffer4 0 0
    semi; lose fl bed='nam.eq.buffer'
    arcomv 1 buff
    op dcd='obj1:gbkez1 obj2:gbkey2'
    arovers 5 5
    ls
    sav result
```

Abb. 38: GIS-Lösung für Anfrage 6. "Flächen in Entfernung x vom Waldweg" aus Tabelle 17 mit einem Aufsatzpaket zur Flächenverschneidung

Ergebnisse

Die oben beschriebenen Anfragen erzielten die in Tabelle 18 dargestellten Laufzeiten. Für drei ausgewählte Anfragen (2,6,7) werden den Zeiten von GQL die von alternativen GIS-Lösungen gegenübergestellt. Die Anzahl der zur Berechnung der Lösung herangezogenen Elemente ist in den Spalten input und picked erfaßt.

Gruppe	Anfrage	Elemente			Laufzeit/sec	
		input	picked	join	nur GIS	GQL
räumliche Selektion	1. große Flächen finden	270	-	-	40[a]	30/15[b]
räumliche Projektion	2. Flächenumfang bestimmter Flächen	270	-	-	n[c]	15/15[d]
räumliche Aggregation	3. Gesamtfläche berechnen	270	-	-	n	10
	4. Länge von Waldwegen berechnen	234	71	-	n	60
räumlicher Verbund (spatial join)	5. Fichtenbestände benachbart zum Waldweg	234	5	1170 meets	2/2[e]	50/70[f]
	6. Flächen in Entfernung x vom Waldweg	302	19	5738 distance	300[g]	100/780[h]
	7. Bestände am Wanderweg	270	45	12150	n	90/110[i]

Tab. 18: Die Laufzeiten der Anfragen auf der Forstbetriebskarte Bayern.

a. in der Geodatenbank mit gbsrt gesucht
b. 30 bei Berechnung mit area, 15 bei Abfrage mit value_of
c. n = nicht getestet
d. 15 unabhängig davon, ob gleichzeitig Sachdaten (join) mit abgefragt werden oder nicht (im Unterschied zu g und h)
e. Berechnung im GIS-Frontend oder in der GeoDB-Middleware für nur 1 selektiertes Element
f. 70 bei gleichzeitiger Berechnung der Flächeninhalte
g. Berechnung durch GIS-Aufsatzpaket zur Flächenverschneidung
h. 780 bei gleichzeitiger Ausgabe von Sachdaten, die über einen zusätzlichen Join verknüpft werden
i. 110 bei gleichzeitiger Ausgabe von Sachdaten, die über einen zusätzlichen Join verknüpft werden

Interpretation

Bei Anfragen, die vollständig vom GIS-Frontend abgearbeitet werden können (Anfrage 5) ist dessen Performance deutlich höher als die von GQL, das für seine Berechnungen auf die Datenbank zugreift. Im Vergleich zur Abarbeitung mit anderen Kommandos der GeoDB-Middleware (Anfrage 1) zeigt GQL vergleichbare Ergebnisse. Dies ist auch im Vergleich mit dem GIS-Aufsatzpaket zur Verschneidung zur Lösung von Anfrage 6 der Fall.

Die Abfragen 5, 6 und 7 zeigen, daß die Zeiten stark ansteigen, wenn GQL-Operatoren für räumliche Joins verwendet werden.

Wertung

Die dargestellten Anfragen (interaktiv, lokal) sind typisch für GIS. Einige Anfragen können in dem untersuchten GIS mit traditionellen Methoden nicht gelöst werden. GQL stellt hier neue Ansätze zur Verfügung. Andere Probleme können im GIS bisher auch gelöst werden, aber nur durch eine lange Abfolge von Befehlen und die Erzeugung vieler Zwischenergebnisse (siehe Beispiel mit Verschneidung). Hier ist die Lösung mit GQL wesentlich kürzer und eleganter. Sie ist intuitiv verständlich, leichter erlern- und anwendbar. Berücksichtigt man auch die Zeit, die der Nutzer für die Interaktion mit dem System benötigt, ist die Performance bei diesen lokalen Anfragen mit der üblicher GIS-Befehle vergleichbar, kann aber die Ausführungszeiten der GIS-Befehle, die im Hauptspeicher auf bereits vorgeladenen Daten ablaufen, nicht erreichen. Hier macht sich bemerkbar, daß GQL zu seiner Ausführung Daten erst aus der Datenbank übertragen muß, bevor räumliche Operatoren ausgewertet werden können. Insgesamt stellt GQL bei dieser Art von Anfragen eine sinnvolle Ergänzung zu den traditionellen GIS-Analysemethoden dar.

Die graphischen Elemente - die Geometrie, auf denen GQL arbeitet, befinden sich auf der Denkebene des Benutzers. Dadurch können Anfragen direkt umgesetzt werden. Daß das nicht selbstverständlich ist, wird bei dem nächsten Anwenderszenario noch deutlich herausgestellt.

6.2.2 Digitale Stadtgrundkarte

Beschreibung: Die Digitale Stadtgrundkarte (DIST) der Stadt München bildet die Flurstücke, Gebäude und Straßen der Gemarkungen des Stadtgebiets ab.

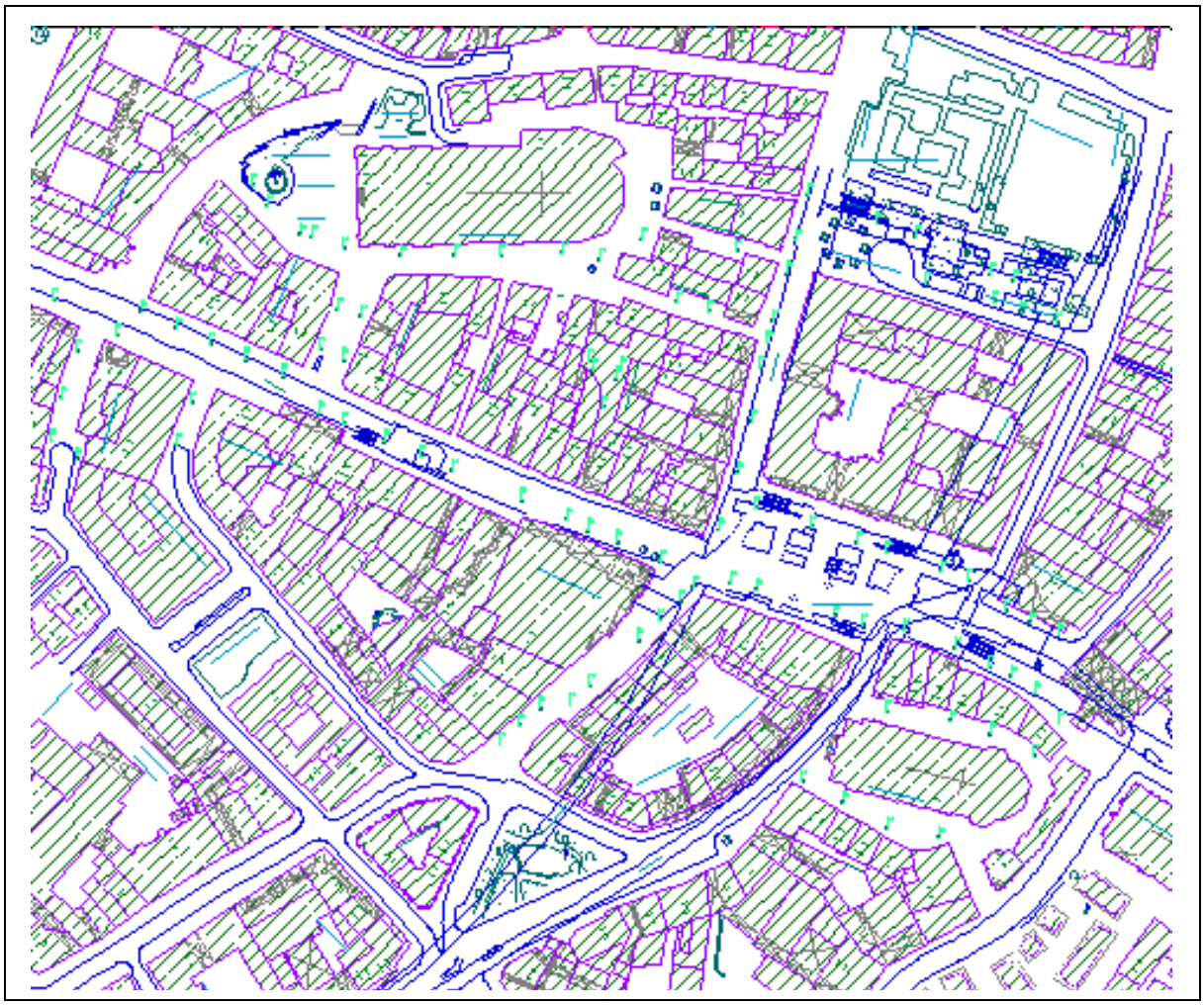


Abb. 39: Ausschnitt der Digitalen Stadtgrundkarte (DIST) der Landeshauptstadt München mit den Umrissen von Gebäuden und Flurstücken

Flurstücke werden durch ihre Grenzen, ihre Fläche und die Flurstücksnummer dargestellt. Zu Gebäuden wird der Umfang, die Fläche, Gebäudedetails, die Anzahl der Geschoße, die Hausnummer und ein Gebäudebezugspunkt erfaßt. Die Straßen werden durch ihre Namen und die Topographie der Straßen und Grünflächen präsentiert. Bögen und Linien, Flächen und Texte, sowie Vermessungspunkte und Flurstücksnummern stellen diese Informationen in der Karte graphisch dar (s. Abb. 39). Zusätzlich ergänzen alphanumerische Sachdaten die graphischen Informationen in der Karte. Sie sind entsprechend Abbildung 42 strukturiert und in einer proprietären Datenbank mit netzwerkartigen Verzeigerungen abgelegt.

Die Flurstücke (FL) gehören Gemarkungen (GM) an, in die das Stadtgebiet eingeteilt ist. Die zusätzlich durch Zähler und Nenner der Flurstücksnummer (NR1, NR2) identifizierten Flurstücke (s. Abb. 40) sind aus einzelnen Flächenteilen (Bereichssatz, Parcel Feature, PF) zu-

sammengesetzt, da sie früher soldnerblattweise (BLASOL) erfaßt wurden. Die Parcel Features sind mit den Flächen FL und den Flurstücksnummern FR TX in der Karte verknüpft.

Außerdem enthalten sie den amtlichen Flächeninhalt und einen Feature Code (FCE), der etwas über ihre Nutzung aussagt. Zu den Parcel Features gehören die Umrißlinien (Liniensatz, F1, F2), die mit den Bögen und Linien BO LI in der Karte verbunden sind. Die Gebäude (GB) sind den Flurstücken zugeordnet, auf denen sie gebaut sind, sie tragen eine Liegenschaftsnummer (LNU). In der Karte sind sie mit den Gebäudeflächen FL und dem Gebäudebezugspunkt PG verknüpft. Außerdem entspricht jedem Gebäude eine Adresse (AD), die in der Karte durch die Hausnummer TX dargestellt wird (s. Abb. 41 und 42).

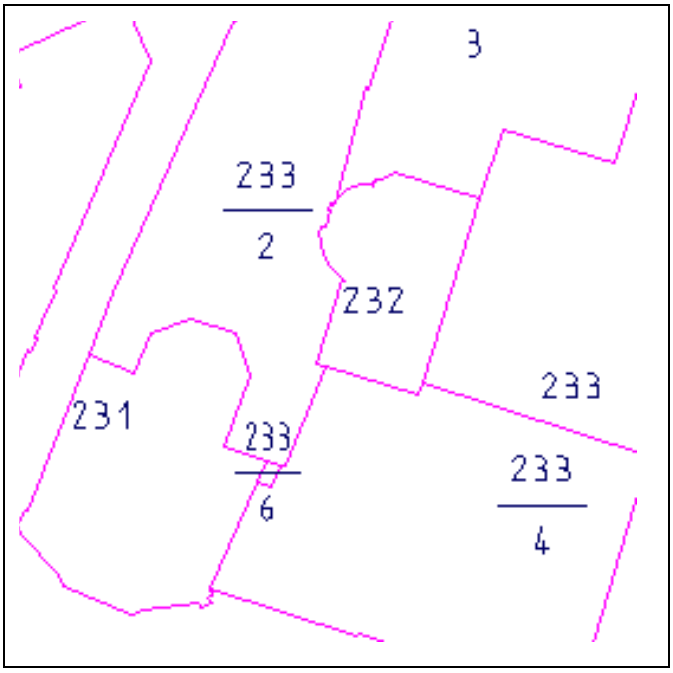


Abb. 40: Flurstücke und Flurstücksnummern

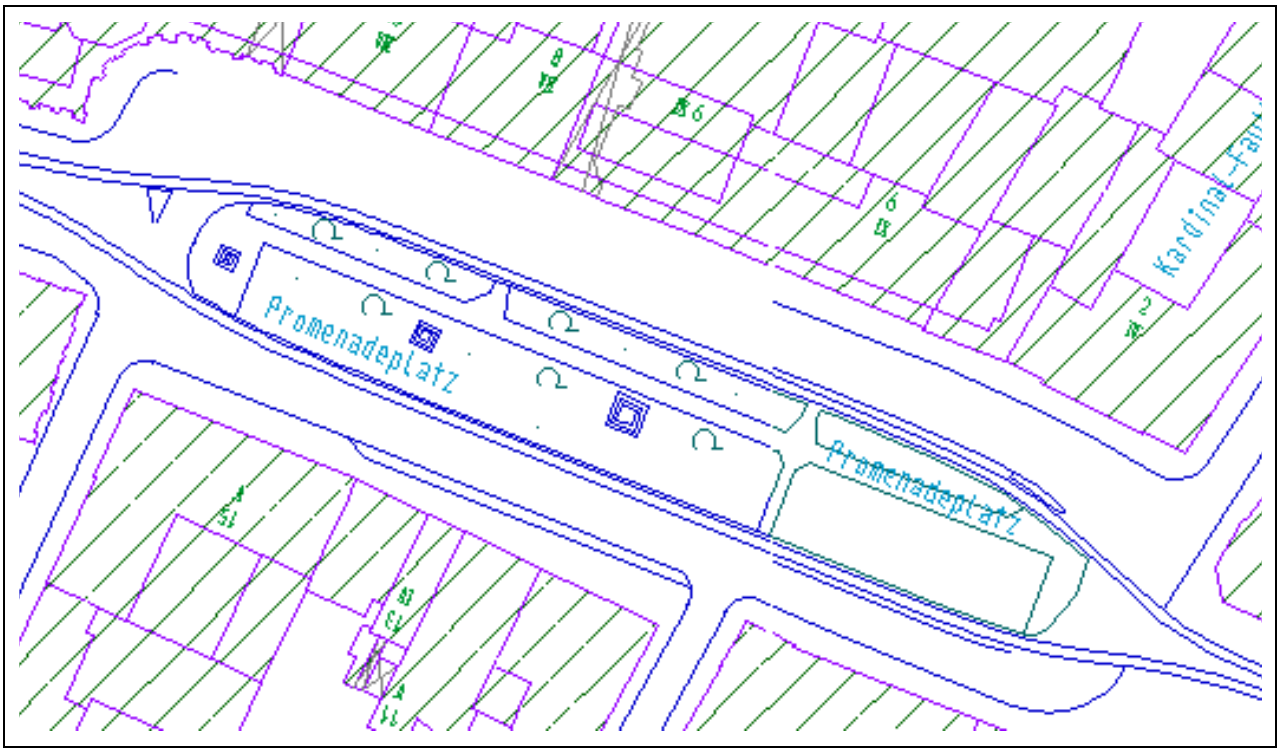


Abb. 41: Gebäudeumrisse und -flächen, Straßennamen und -topographie

.

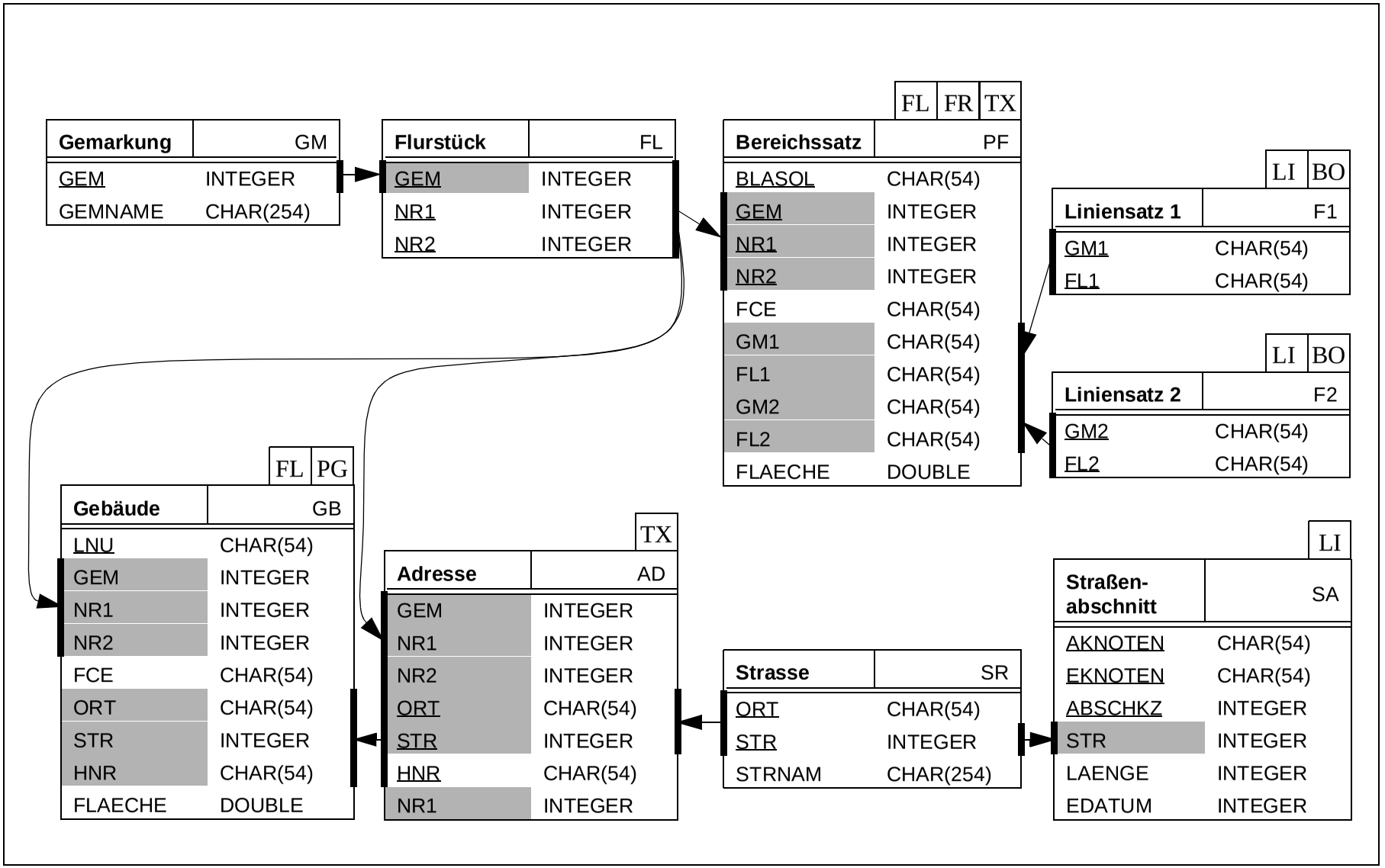


Abb. 42: Die Sachsatzstruktur zur Digitalen Stadtgrundkarte (DIST) der Landeshauptstadt München (Auszug).

Die Straßen (SR) erhalten einen Namen (STRNAM) und werden durch ihren Ort (ORT) und einen Straßenschlüssel (STR) identifiziert. Der Straßenschlüssel stellt die Verbindung zu den einzelnen Straßenabschnitten her, die in der Graphik durch Linien LI dargestellt sind.

Die untersuchten **Daten** der Digitalen Stadtgrundkarte (DIST) werden zusätzlich ergänzt durch die Straßenachsen (369 Linien und 321 Texte) und die Baublöcke (351 Flächen). Die Baublocksflächen sind mit Tabelle BL verknüpft. Die DIST enthält in dem ausgewählten Ausschnitt 7.576 Flächen, 16.194 Linien und als punktförmige Objekte 3.323 Flurstücksnummern, 20.133 eigenständige Punkte, 4.648 Symbole und 8.878 Texte.

Auf diesen Daten werden **Anfragen** durchgeführt. Die Anfragen arbeiten global auf dem gesamten Datenbestand der Geodatenbank und erstellen als Ergebnis eine thematische Karte oder eine tabellarische Ergebnisliste. Tabelle 19 gibt einen Überblick über die untersuchten Anfragen, ihre Eingabe und Ausgabe, und die verwendeten Methoden. Zum Vergleich siehe auch Tabelle 17.

Gruppe	Sach	Geo	Anfrage	Ergebnis		Methoden
räumlicher Verbund (spatial join)	S	F, P	1. Flächenaggregation	H		within, thematische Karte
	S	F, L	2. Standortanalyse	H		distance
	S	P	3. Qualitätsprüfung		T	distance, Modellierung

Tab. 19: Ausgewählte Anfragen auf der digitalen Stadtgrundkarte München. Die Einteilung in Gruppen folgt der Einteilung des Sequoia 2000 Benchmarks (siehe Tabelle 15). Die Anfragen unterscheiden sich weiter durch ihre Eingabedaten: Sach- (S) und Geodaten (G), Flächen (F), Linien (L) und Punkte (P), und nach ihrem Ergebnis (E): hervorgehoben in der Graphik (H) oder tabellarisch (T).

Durchführung

Im Unterschied zu den beschriebenen Anfragen des Anwenderszenarios Forst ist bei den Anfragen der Stadt München vor Ausführung der Anfrage nicht bekannt, in welchem Ausschnitt das Ergebnis liegt (Standortanalyse) oder das ganze Gebiet soll dargestellt werden, aber mit einer reduzierten Auflösung (Aggregation). Sie erlauben damit nicht die traditionelle Vorgehensweise, einen Ausschnitt zu laden und im Hauptspeicher zu bearbeiten. Dies verbietet sich zum Teil auch allein wegen der Datenmenge. Die Abarbeitung der Anfrage betrifft stattdessen die gesamte räumliche Ausdehnung der Datenbank. Für die graphische Präsentation der Ergebnisse kann aber, wie dargestellt, auf die Werkzeuge des GIS zurückgegriffen werden.

Diesen Aufgaben und deren Lösung ist gemeinsam, daß sie nicht direkt auf der Geometrie aufsetzen können, wie im vorangegangenen Anwenderszenario. Mehr oder weniger umfangreiche Vorbereitungen sind erforderlich, um die Modellierung anzupassen oder das Ergebnis darzustellen.

ad 1. Flächenaggregation

Gesucht ist eine Karte eines Stadtgebietes, die baublockbezogen den Versiegelungsgrad darstellt. Diese Information ist für die einzelnen Flurstücke erfaßt. Ein Baublock enthält mehrere Flurstücke. Die Werte der Versiegelung der einzelnen Flurstücke müssen zusammengefaßt und im Verhältnis zu der Gesamtfläche des Baublocks ausgegeben und graphisch dargestellt werden. Die Lösung erfolgt in drei Schritten (s. Abb. 43): erstens der Zusammenführung der Information über das räumliche Kriterium, zweitens der Auswertung in der Datenbank und drittens der graphischen Darstellung im GIS (s. Abb. 44).

```
1   Räumlichen Bezug herstellen
    INSERT      INTO tmp(bl_location, fl_location, fl_inhalt, fl_vg)
    SELECT      bl.location, pf_j.location, pf.inhalt, pf.vg
    FROM        pf, pf_j, bl
    WHERE       within(bl.location, pf_j.location)
    AND         value_of(pf_j.location,etyp) = 'FR'
    AND         pf.gem = pf_j.gem
    AND         pf.nr1 = pf_j.nr1
    AND         pf.nr2 = pf_j.nr2
    AND         pf.blasol = pf_j.blasol

2   Versiegelungsgrad berechnen
    INSERT      INTO result(bl_location, bl_vg)
    SELECT      bl_location, sum(fl_inhalt*fl_vg) / sum(fl_inhalt)
    FROM        tmp
    GROUP BY    bl_location

3   Thematische Karte bilden
    INSERT      INTO elementset
    SELECT      bl_location from result
    WHERE       bl_vg = 9

    semi; gbmgmg emsm 0 0;
    pas fl st=9
```

Abb. 43: GQL-Lösung für Anfrage 1. "Flächenaggregation" aus Tabelle 19

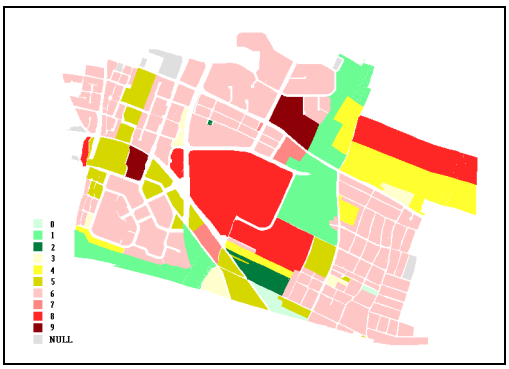


Abb. 44: Graphische Darstellung des Versiegelungsgrads von Baublöcken

Die Darstellung in der Karte erfordert hier mehr als nur ausgewählte Elemente graphisch hervorzuheben. Stattdessen werden die Baublöcke entsprechend dem errechneten Versiegelungsgrad eingefärbt. Dazu werden die Elemente mit einem bestimmten Versiegelungsgrad selektiert und die Farbe gesetzt. Dafür wird Schritt 3 für jede Stufe einmal durchgeführt.

ad 2. Standortanalyse

Gesucht ist ein Grundstück, das den Bedingungen für den Bau eines Kindergartens genügt (s. Abb. 45). Dafür muß es:

- eine Fläche (5)
- größer als 2.000 m^2 (3) sein,
- der Stadt gehören (4) und
- mehr als 100 m (10) entfernt
- von Bundesstraßen (9) sein.

```
1   SELECT    pf_j.location
2   FROM      pf, pf_j, sa_j
3   WHERE     area(pf_j.location) > 2000
4   AND       pf.fce = '01,004'
5   AND       value_of(pf_j.location,etyp) = 'FL'
6   AND       pf.gem = pf_j.gem
7   AND       pf.nr1 = pf_j.nr1
8   AND       pf.nr2 = pf_j.nr2
9   AND       value_of(sa_j.location,layer) = 107
10  AND       distance(pf_j.location, sa_j.location) > 100
```

Abb. 45: GQL-Lösung für Anfrage 2. "Standortanalyse" aus Tabelle 19.

Die gefundenen Flächen werden in das GIS übertragen und graphisch dargestellt.

ad 3. Qualitätsprüfung

Eine Anfragesprache kann dazu verwendet werden, räumliche Bedingungen zu definieren und ihre Korrektheit zu überprüfen [Ubeda & Egenhofer 97]. Bei dieser Anfrage geht es darum, den Anwender bei der Überprüfung der Korrektheit von Adressen zu unterstützen. Eine gespeicherte Adresse kann einem falschen Straßenabschnitt zugeordnet sein. Da die Adressen über diese Zuordnung georeferenziert werden, können Abweichungen mit der GQL-Funktion distance festgestellt werden. Überschreitet die Entfernung zweier aufeinanderfolgender Adressen einen bestimmten Schwellenwert, so ist möglicherweise eine der beiden falsch zugeordnet. Das Ergebnis ist eine tabellarische Liste der abweichenden Adressen, die der Anwender dann von Hand nachprüft.

Eine ideale Umsetzung dieser Anfrage in GQL könnte wie in Abb. 46 dargestellt lauten. Die Modellierung der Daten im GIS macht diese einfache Abfrage aber unmöglich, da erstens die Hausnummern nicht fortlaufend durchnumeriert sind und zweitens alphanumerische Zusätze zur Hausnummer (wie z. B. in 2a oder 12R) in der Datenmodellierung nicht getrennt sind.

```
1   SELECT    sr.strnam, ad1.hnr, ad2.hnr, distance(ad1.location, ad2.location)
2   FROM      ad_j ad1, ad_j ad2, sr
3   WHERE     to_number(ad1.hnr) = to_number(ad2.hnr) + 2
4   AND       ad1.str = sr.str
5   AND       ad2.str = sr.str
6   AND       distance(ad1.location, ad2.location) > 150
```

Abb. 46: Ideale GQL-Lösung für Anfrage 3. "Qualitätsprüfung" aus Tabelle 19

Um die Anfrage trotzdem bearbeiten zu können, ist es notwendig, die Daten vorher aufzubereiten, indem die Hausnummern und Buchstaben getrennt, nach Straßenseiten aufgeteilt, aufsteigend sortiert, mit einer fortlaufenden Nummer versehen und in einer Tabelle ad_ord abgelegt werden. Dann lautet die Anfrage wie in Abb. 47 angegeben.

```
1  SELECT   sr.strnam, ad1.hnr, ad2.hnr, distance(ad1.location, ad2.location)
2  FROM     ad_ord ad1, ad_ord ad2, sr
3  WHERE    ad1.id = ad2.id + 1
4  AND      ad1.str = ad2.str
5  AND      distance(ad1.location, ad2.location) > 150
```

Abb. 47: Reale GQL-Lösung für Anfrage 3. "Qualitätsprüfung" aus Tabelle 19

Der zusätzlich benötigte Aufwand, bevor die Anfrage mit GQL abgesetzt werden kann, schränkt die Nutzbarkeit wesentlich ein. Sind entsprechende Datenstrukturen nicht vorhanden, können die nötigen Vorarbeiten genauso viel Zeit in Anspruch nehmen, wie ein Programm zu schreiben, das direkt auf dem GIS aufsetzt. Dies ist hier auch der Fall. Der Anwender löst das vorliegende Problem mit einem Fortran-Programm, das auf einer Schnittstelle der Geodatenbank aufsetzt. Das Programm ist speziell auf die Datenstrukturen zugeschnitten, arbeitet die netzartigen Verbindungen zwischen den Daten ab und erreicht so das Ergebnis schneller als GQL. GQL steht dieses Wissen nicht für die Optimierung zur Verfügung. Dadurch werden sehr viele unnötige Vergleiche innerhalb des räumlichen Verbunds gerechnet, was sich negativ auf die Performance auswirkt.

Ergebnisse

Die Laufzeiten der Anfrage auf der Digitalen Stadtgrundkarte München sind in Tabelle 20 dargestellt. Die Laufzeiten liegen im Unterschied zu den räumlich stark eingeschränkten Anfragen des Anwenderszenarios Forst im Bereich von Minuten statt von Sekunden.

Gruppe	Anfrage	Elemente	Laufzeit/min	
			GIS	GQL
räumlicher Verbund (spatial join)	1. Flächenaggregation		-	17
	2. Standortanalyse	3323 + 16	-	5
	3. Qualitätsprüfung	2405	1	360

Tab. 20: Die Laufzeiten der Anfragen auf der Digitalen Stadtgrundkarte München.

Die Ursache dafür sind: Die Anfragen betreffen mehr geographische Elemente, was in mehr teuren Operationen auf Geodaten resultiert. Außerdem sind die Geo- und Sachdaten stärker miteinander verknüpft. In Abfragen werden diese durch Joins mit den Geoanfragen verbunden.

Wertung

Wenn die Modellierung der Daten (verknüpfte Sachdaten oder Geometrie) nicht der Denkebene entspricht, auf der sich der Anwender bewegt, ist die Formulierung von Anfragen mit einer Sprache wie GQL umständlich. Aufwendige Vorarbeiten, wie das Umformen der Hausnummern in Beispiel 3. "Qualitätsprüfung", verringern den Vorteil der einfachen Formulierung einer deklarativen Anfrage. D. h. paßt die Modellierung nicht, kann eine prozedurale Bearbeitung bessere Ergebnisse liefern. Sind zur Lösung der Anfrage Joins über große Datenmengen notwendig, ohne daß die Menge der in Fragen kommenden Elemente vorher eingeschränkt werden kann, so zeigt sich hier die Schwäche von GQL in der Anfrageverarbeitung.

6.3 Ergebnisse

Es gibt zwei **Klassen von Anfragen**, für die eine Anfragesprache wie GQL, die on top auf einem RDMBS sitzt und in ein GIS integriert ist, mehr oder weniger gut geeignet ist.

Zum einen gibt es Anfragen, bei denen das zu untersuchende Gebiet schon festliegt und komplett in das GIS geladen ist. Aus den in diesem Gebiet vorhandenen Objekten werden mit Hilfe von GQL solche mit bestimmten Eigenschaften herausgefiltert. Dies entspricht dem traditionellen inkrementellen Vorgehen in GIS. Hier spielt GQL seine Stärken aus, wie z. B. die gute Integration in das GIS, und ergänzt die bestehende GIS-Funktionalität sinnvoll.

Zum anderen gibt es Anfragen, bei denen noch nicht festgelegt ist, wie das Ergebnis räumlich verteilt ist, wenn die Anfrage gestellt wird. Deswegen ist es erforderlich, die gesamte Datenbank zu durchsuchen, um die Treffermenge zu finden und dann erst im GIS graphisch darzustellen. Bei diesen Anfragen überwiegt die Datenbankfunktionalität. Hier macht sich mit Leistungseinbußen bemerkbar, daß GQL nicht voll in die Datenbank integriert ist.

6.4 Zusammenfassung

Die geographische Anfragesprache (Geographical Query Language, GQL) basiert auf relationaler Datenbanktechnik. Sie ist voll in den SICAD Geodatenserver integriert, den sie um Operatoren für räumliche Anfragen erweitert. Sie entspricht dem aktuellen Standard des OpenGIS Konsortiums und praktischen Anwenderanforderungen. GQL greift über einen Quadtree, der in Z-Ordnung in relationalen Tabellen gespeichert ist, auf die Geodaten zu und filtert die Daten nach räumlichen und alphanumerischen Kriterien, bevor es die endgültige Treffermenge berechnet (Filter and Refine). Die Ergebnisse stehen im GIS zur Verfügung und können dort zur graphischen Darstellung weiter bearbeitet werden.

Die Funktionalität eignet sich zur Lösung von Anwenderproblemen. Die Performance reicht bei GIS-typischer Arbeitsweise, d. h. der zu bearbeitende Ausschnitt wird geladen und vor-

bereitet und dann Anfragen gestellt, aus. Die Performance ist nicht ausreichend bei Anfragen, die Daten aus dem gesamten Gebiet der Datenbank herausfiltern. Gründe dafür sind:

1. Indizierung
   GQL basiert auf einer Geodatenbank, deren Haupteinsatzzweck das sichere Laden und Speichern von räumlich zusammenhängenden Daten, nicht aber die Analyse von räumlich weitverteilten Daten ist. Deswegen steht in der Geodatenbank zwar ein für räumlichen Zugriff effizienter 2-dimensionaler Index (Quadtree), aber kein multidimensionaler Index auf sonstige in dieser Datenstruktur versteckte Sachdaten (Deskriptoren an Elementen) zur Verfügung. Mehrdimensionale Indizes werden aber für gemischte Anfragen benötigt. Denn wenn über diese Attribute nicht indiziert werden kann, müssen für eine Analyse immer alle Objekte untersucht werden (Bsp.: Suche alle Linien mit Strichstärke 3; für diese Anfrage kann kein Index benutzt werden). GQL selbst führt keine neuen Indizierungsmöglichkeiten in das zugrunde gelegte GeoDBMS ein.

2. Anfrageverarbeitung
   GQL nutzt nicht alle Möglichkeiten, die Filter-und-Refine noch bieten könnte. Da es nicht in der Datenbank integriert ist, sondern on top aufgesetzt ist, muß GQL bei der Berechnung von Funktionsergebnissen alle Ergebnisse und die nötigen Informationen zur Identifizierung in Tabellen zwischenspeichern. Das verschlechtert die Performance. Nur eine integrierte oder vollkommen eigenständige Verarbeitung eröffnet die Möglichkeit, zwischen den einzelnen Prozessen der Berechnung einer Anfrage eine Pipeline aufzubauen und somit teuere externe Zwischenspeicherung zu vermeiden.

3. Nested-Loop-Join
   GQL arbeitet den räumlichen Verbund (spatial join) nach der Methode des einfachen Nested-Loop-Join ab. Das hat bei großen Datenmengen stark wachsende Ausführungszeiten zur Folge und macht Auswertungen auf dem gesamten Plangebiet ohne vorherige Einschränkung der Eingabedaten unmöglich.

Will GQL vom Anwender als vollwertiges Werkzeug akzeptiert werden, ist es notwendig die Performance zu erhöhen. Dies kann auf zwei Arten geschehen:

1. Durch die Verbesserung von GQL selbst. Das ist bei den Punkten 2 und 3 möglich, indem die Anfrageverarbeitung optimiert, die bestehende Datenstruktur des Quadtree verstärkt zur Filterung herangezogen und eine bessere Join-Methode eingebaut wird. Beispiele für Joinmethoden für geographische Daten finden sich in der Literatur z. B. bei [Günther 98]. Außerdem könnten die internen Datenstrukturen (Master-Detail) in der Anfrageverarbeitung genutzt werden.

2. Oder indem GQL auf einer Datenbank aufgesetzt wird, die eine räumliche Anfragesprache unterstützt. Dabei ist zu berücksichtigen, daß der Ansatz dieser Arbeit ist, eine An-

fragesprache *innerhalb eines GIS* zur Verfügung zu stellen. Deswegen genügt es nicht, einfach ein solches Produkt unter ein GIS zu setzen. Das allein ist noch kein Ersatz für GQL. Vielmehr muß die Anfragesprache in das GIS integriert werden, um eine vergleichbare Funktionalität zu bieten.

Die zweite Lösung hat den Vorteil, daß Neuerungen der DB-Forschung auch innerhalb von GQL zur Verfügung stehen, sobald sie von dem DB-Hersteller umgesetzt werden. Statt zu versuchen, mit ständig sich weiterentwickelnden Produkten schrittzuhalten, kann sich GQL auf die für das GIS wesentlichen Eigenschaften konzentrieren. Diese liegen mit der Verbesserung der Datenbanken als Basis für GQL verstärkt auf der anfragegerechten Modellierung der Geodaten und der Integration in das Gesamtsystem.

Darum untersucht das folgende Kapitel objektrelationale Techniken zur Modellierung von Geodaten und die Integration objektrelationaler Datenbanken in GIS und zeigt, welche Konsequenzen das auf die Durchführbarkeit von räumlichen Anfragen in GIS hat.

7. Untersuchung einer geographischen Anfragesprache auf der Basis eines objektrelationalen Datenbankmanagementsystems

Der Test der auf relationaler Datenbanktechnologie basierenden Anfragesprache GQL zeigt deren Defizite im Bereich der Performance. Sie treten bei Joins mit großen Datenmengen auf. Die Ursachen dafür liegen in der Anfrageverarbeitung. Sie ist als On-Top-Lösung nicht optimal ins Datenbanksystem integriert. Dem gegenüber bieten erweiterte Datenbanksysteme Vorteile: räumliche Spracherweiterungen existieren und die Anfrageverarbeitung wird durch die in das DBMS integrierten räumlichen Zugriffsmethoden unterstützt. Im folgenden wird an einem Beispielsystem untersucht, inwieweit sich diese Datenbankerweiterungen für reale GIS-Anwendungen und insbesondere geographische Anfragen eignen und vor welchen Aufgaben ein Anwender steht, der eine GIS-Anwendung auf der Basis einer objektrelationalen Datenbank mit integrierter räumlicher (spatial[1]) Komponente aufbauen will.

Aus den in Kapitel 4.3.2 vorgestellten relationalen oder objektrelationalen Datenbankmanagementsystemen (RDBMS bzw. ORDBMS), die um eine Komponente zur räumlichen Datenhaltung erweitert sind, wird das objektrelationale System Oracle8i ausgewählt, da es objektorientierte Modellierungsmöglichkeiten und das Datenbankerweiterungsmodul Oracle Spatial mit räumlichen Datentypen und Operatoren bereitstellt.

Will der Anwendungsentwickler ein solches System nutzen, stellen sich ihm andere Aufgaben als bei der Entwicklung und dem Einsatz von GQL: Er kann keinen Einfluß auf die Auswahl der Operatoren nehmen, sondern muß die angebotene Sprache verwenden, wie sie ist. Außerdem liegen heute üblicherweise die Anwenderdaten bereits in einem GIS vor. Die Modellierung der Geodaten ist auf die Möglichkeiten dieses GIS zugeschnitten. Will man die Möglichkeiten der ORDBMS nutzen, müssen die Daten transferiert werden. Im ORDBMS stehen bestimmte Mittel zur Modellierung zur Verfügung, die nicht notwendigerweise mit denen des GIS übereinstimmen. Deswegen ist das Datenmodell im GIS auf ein entsprechendes Datenmodell im ORDBMS umzusetzen. Dabei ist der spätere Einsatzzweck der Daten zu berücksichtigen.

1. Eine Komponente zur Speicherung räumlicher (spatial) Daten allein ist noch keine GIS-Komponente. Dazu wird sie erst durch entsprechende Modellierung, welche die für GIS fehlende Basisfunktionalität wie Präsentation und Bildung von Geo-Objekten ergänzt.

Das ORDBMS bietet selbst keine Möglichkeit, die Ergebnisse geographischer Anfragen graphisch anzuzeigen. Das macht den Einsatz eines GIS-Frontends als Schnittstelle zum Nutzer notwendig. Über das GIS-Frontend werden die Anfragen formuliert und an die Datenbank geschickt. Das geschieht über eine Zwischenschicht, welche die Datenbank mit dem Frontend verbindet. In ihr werden auch die zurückgelieferten Anfrageergebnisse umgeformt, bevor das Frontend sie anzeigt.

Nach diesen Überlegungen stellen sich, wie in Abbildung 48 dargestellt, für den Aufbau der GIS-Anwendung die drei Aufgaben

1. Transformation der Modellierung,
2. Transfer der Daten und
3. Integration von Datenbanksystem und GIS-Frontend.

Die drei Aufgaben werden am Beispiel von Landmanagementdaten exemplarisch durchgeführt und in den folgenden Abschnitten im einzelnen erläutert.

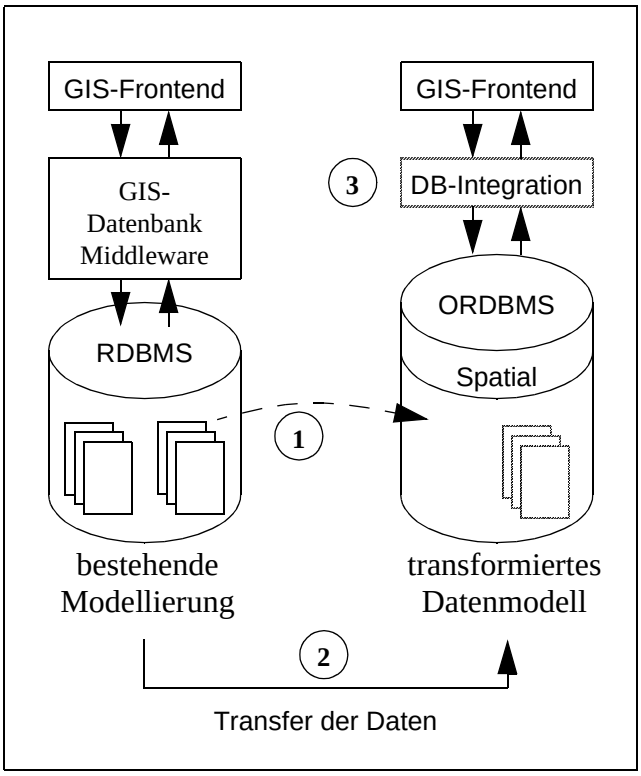


Abb. 48: Die drei Schritte zum Aufbau eines GIS auf der Basis eines ORDBMS mit geographischem Erweiterungsmodul bei vorhandenen GIS-Daten: (1) Transformation der Modellierung, (2) Transfer der Daten und (3) Implementierung der DB-Integrationsschicht

7.1 Transformation des Datenmodells und Datenmigration

Dieses Kapitel entwickelt nach bestimmten Modellierungsprinzipien (7.1.1) ein anwendungsunabhängiges GIS-Basismodell (7.1.2). Darauf baut ein anwendungsspezifisches Applikationsmodell (7.1.3) auf, das am Beispiel einer Landmanagement-Applikation vorgestellt wird. Vorhandene alte Datenbestände werden an dieses Modell angepaßt und in die Datenbank eingespielt (7.1.4).

7.1.1 Prinzipien für die objektrelationale Modellierung

Die Neumodellierung der Geodaten mit den Mitteln der objektrelationalen Datenbank stützt sich auf einige Grundprinzipien, die im folgenden kurz erläutert werden:

1. Allgemeinheit und Übertragbarkeit des verwendeten Modells
   Das Modell wird am Beispiel einer Landmanagementanwendung aufgestellt. Dabei wird trotzdem versucht, das Modell so allgemein zu gestalten, daß es auf andere GIS-Anwendungen leicht übertragen werden kann. Dies kann zum Beispiel durch Trennung von Geometrie, Präsentation und Sachdaten erreicht werden.
2. Ausnutzung der Modellierungmöglichkeiten der objektrelationalen Datenbank
   Das objektrelationale Datenbankmanagementsystem stellt Modellierungsmöglichkeiten wie z. B. Objektbildung und Referenzen zur Navigation zwischen Objekten zur Verfügung. Die Arbeit soll diese ausnützen, um zu zeigen, inwieweit sie die Anwendungsentwickler unterstützen können.
3. Einfache Formulierung von Anfragen
   Die Modellierung der Daten zielt darauf ab, Anfragen so einfach wie möglich stellen zu können. Eine Modellierung, die dem Anwender der Anfragesprache unnötig komplexe Sprachkonstrukte aufzwingt, wird kaum akzeptiert. Zu berücksichtigen ist hier, auf welcher gedanklichen Ebene der Anwender einsteigt und wo die Elemente sind, mit denen er sich befaßt. Auf dieser Ebene sollten auch die Operatoren der Anfragesprache angreifen können.
4. Einfache Implementierung des Modells
   Die angestrebte Modellierung sollte mit den Mitteln, die das DBMS zur Verfügung stellt, einfach umgesetzt werden können. Das bedeutet, wenig Code selbst zu schreiben, auf möglichst hohen Schnittstellen aufzusetzen und möglichst wenig in das System einzugreifen.

Die Modellierung basiert auf den Möglichkeiten von Oracle8i und Oracle Spatial, das in Kapitel 4.3.2 vorgestellt wurde.

Basierend auf diesen Grundprinzipien entwickelt die Arbeit das Modell in zwei Teilen: Ein anwendungsunabhängiger Teil (7.1.2) stellt grundlegende Mechanismen zum Auffinden der Daten über Metadaten und für die Verknüpfung von Präsentations- und Geometrieinformationen bereit. Er bildet den Rahmen für den anwendungsspezifischen Teil (7.1.3), das Applikationsdatenmodell.

7.1.2 Entwicklung eines anwendungsunabhängigen Basismodells

Konzept

Das anwendungsunabhängige Basismodell deckt drei Aufgabenbereiche ab: Erstens speichert es Informationen darüber, wo welche Geodaten in welcher Darstellung in der Datenbank zu finden sind (Metadaten). Zweitens gibt es ein Schema vor, wie diese Geodaten in Tabellen abgelegt werden (Geo-Objekte). Und drittens verknüpft es Präsentationsinformationen mit den Geo-Objekten.

1. Metadaten

   Im OGC Standard [OGC SF SQL 98] ist festgelegt, welche Informationen mindestens über ein Geo-Objekt (bei OGC wird ein Geo-Objekt Feature genannt) vorhanden sein müssen. Diese werden in der Tabelle ogis_geometry_columns abgelegt. Sie identifiziert durch den Benutzer- und Tabellennamen eindeutig eine Tabelle der Datenbank, die eine Spalte mit einem räumlichen Attribut (spatial attribute, bei Oracle: SDO_GEOMETRY) enthält. Auf die Speicherung der Geo-Objekte selbst wird im Abschnitt 2 näher eingegangen.

   Das vorgestellte Modell erweitert die vom OGC vorgeschriebenen Metadaten um Informationen über die graphische Präsentation der Geo-Objekte. Diese Metainformation ist in der Tabelle geoobject_presentations abgelegt (s. Tab. 21).

geoobject_presentations						
gp_id	descriptive_name	geoobject_table			style_table	geoobject_style_ intersection_table
		user	name	column		

Tab. 21: Die Metadaten-Tabelle geoobject_presentations ordnet einem Geo-Objekt eine Style-Tabelle und eine Verknüpfungstabelle zu.

   Sie enthält zu einem durch die Namen von Benutzer, Tabelle und Geometriespalte eindeutig referenzierten Geo-Objekt (geoobject_table_user, -_name, -_column) erstens den Namen der zugehörigen Style-Tabelle (style_table) und zweitens den Namen einer Verknüpfungstabelle (geoobject_style_intersection_table), welche einem Geo-Objekt einen Style zuordnet. Diese beiden Tabellen werden im Detail in Abschnitt 3 beschrieben.

2. Geo-Objekte

Räumliche und nichträumliche Attribute zusammen bilden ein Geo-Objekt. Ein Geo-Objekt wird in einer Zeile einer Tabelle gespeichert, der Geo-Objekt-Tabelle. Das räumliche Attribut wird in einer Spalte vom Typ Geometrie abgelegt, die weiteren Attribute in zusätzlichen Spalten (s. Tab. 22).

geoobject_table		
g_id	geometry	name
	SDO_Geometry	

Tab. 22: Das Schema einer Geo-Objekt-Tabelle zeigt die mindestens notwendigen Spalten für Geo-Objekt-ID und Geometrie und als Beispiel für ein zusätzliches weiteres Attribut eine Spalte für einen Namen.

Die oben beschriebenen Informationen genügen, um Geo-Objekte zu finden und anzuzeigen. Die Geo-Objekte selbst enthalten aber noch keine für anspruchsvollere Präsentation notwendigen Informationen. D. h. die Darstellung wäre rein schwarzweiß und an die gespeicherten Geometrien gebunden. Ohne weiterführende Darstellung kommt aber kein fortgeschrittenes GIS aus.

3. Graphische Präsentation

Darum enthält das Modell weitere Informationen über mögliche Präsentationen der Geo-Objekte und ihrer Attribute. Die Geometrie eines Geo-Objekts kann als punkt-, linien- oder flächenhafte Graphik, andere Attribute des Geo-Objekts durch Texte oder Symbole dargestellt werden.

Für diese fünf grundlegenden Darstellungsmöglichkeiten stellt das Modell Präsentationsinformationen (Styles) zur Verfügung. Ein Style enthält alle Informationen, die zur Darstellung eines bestimmten Geo-Objekts notwendig sind, wie z. B. Dicke, Farbe und Muster einer Linie oder die Schriftart und -größe eines Textes. Auf Details der Präsentationsinformationen selbst soll hier nicht genauer eingegangen werden, da diese für jede Applikation unterschiedlich sein können. Stattdessen wird genauer ausgearbeitet, wie die Präsentationsinformationen mit den eigentlichen Geo-Objekten verknüpft werden.

Es gibt dafür zwei Arten von Tabellen: erstens Style-Tabellen und zweitens Tabellen, welche einen Style einem Geo-Objekt zuordnen:

style_table			
style_id	style_name	description	geostyle

Tab. 23: Das Schema einer Style-Tabelle enthält alle Eigenschaften eines Styles im Attribut geostyle.

Eine Style-Tabelle (s. Tab. 23) enthält für jede Präsentationsart alle für die Darstellung benötigten Informationen (geostyle). Dieses Attribut ist ein komplexes Objekt, dessen

Aufbau hier nicht näher beschrieben werden soll. Ein Name (style_name) und eine verständliche Beschreibung (description) dienen der Interaktion mit dem Benutzer bei der Auswahl und der Zuordnung von Styles. Systemintern stellt eine eindeutige ID (style_id) die Beziehung zwischen Styles und Geo-Objekten her.

Diese Beziehung wird in einer Verknüpfungstabelle (geoobject_style_intersection_table) abgebildet (s. Tab. 24). Sie ordnet einem Geo-Objekt (geoobject_id) aus der Geo-Objekt-

geoobject_style_intersection_table		
geoobject_id	style_id	sequence_number

Tab. 24: Das Schema einer Verknüpfungs-Tabelle zeigt die Zuordnung von Styles zu Geo-Objekten.

Tabelle einen Style aus der Style-Tabelle zu. Einem Geo-Objekt können auch mehrere Präsentationen zugeordnet werden. Sie werden durch eine fortlaufende Nummer (sequence_number) voneinander unterschieden. Dadurch ist es z. B. möglich, ein Geo-Objekt Gebäude sowohl durch die Geometrie seiner Grundfläche, als auch durch die textuelle Präsentation der Hausnummer darzustellen.

Vorteile:

Multiple Präsentation derselben Geo-Objekte

Beim Lesen der Daten werden die Informationen zum Geo-Objekt und zur Präsentation über einen Verbund (Join) in der Datenbank zusammengeführt. Die Applikation sieht eine integrierte Sicht beider Informationen. Dies erlaubt, multiple Präsentationen für den selben Geodatenbestand vorzuhalten. Die Geometrien in der Geo-Objekt-Tabelle selbst bleiben dabei unverändert. Allein die Präsentationsdaten werden um zusätzliche Style-Tabellen ergänzt, die durch Verknüpfungstabellen mit den Geo-Objekten gekoppelt werden. Ein Wechsel der Präsentation kann ohne Änderungen in der Applikation auf Datenbankebene vorgenommen werden. Der Maßstab der Präsentationen weicht jedoch in der Regel nur geringfügig vom Originalmaßstab der Geo-Objekte ab.

Default-Style

Wenn kein Style zugeordnet ist, wird ein Defaultstyle verwendet. Dies erlaubt, entweder alle Geo-Objekte einer Klasse in einem einheitlichen Style darzustellen, oder einzelnen Geo-Objekten individuelle Styles zuzuordnen.

Regelbasierte Style-Zuordnung

Ein regelbasierter Mechanismus[2] ordnet einzelnen Geo-Objekten individuelle Styles zu. Die Verknüpfungstabelle hält das Ergebnis der Auswertung dieser Regel fest. Dadurch kann per-

2. Die Idee regelbasierter Zuordnung von Styles läßt sich in noch viel deutlicherer Ausprägung in dem auf einer objektorientierten Datenbank basierenden System GODOT [Gaede & Riekert 94] finden.

formant auf ein Geo-Objekt und seine einmal zugeordnete Präsentation zugegriffen werden, erst bei einer Veränderung des Geo-Objekts wird die Regel neu ausgewertet und die Präsentation gegebenenfalls geändert.

Eigene Präsentationsobjekte

Ergänzt man die Verknüpfungstabelle (geoobject_style_intersection_table) um eine Geometriespalte, so können Generalisierungsinformationen oder generalisierte Darstellungen gespeichert werden (z. B. ein Verschiebungsvektor, eine neue gesetzte Position oder ein ganz neues Objekt), die sich auf eine bestimmte Präsentation auswirken. Die Geometrien und Positionen der Geodaten selbst bleiben dadurch unberührt. Eigentlich als Fläche gespeicherte Geo-Objekte können damit auf einen Punkt reduziert oder ganz ausgeblendet werden.

Konformität mit den Standards der OGC

Das Modell ist konform zum OGC-Standard [OGC SF SQL 98]. Im Bereich der Präsentation, wo OGC bisher noch keine Lösung erzielen konnte, wird das OGC Simple Features Modell geeignet erweitert, um auch anspruchsvollere GIS-Darstellungen zu ermöglichen.

Trennung von Koordinaten und Präsentation

Die im Geometrie-Attribut gespeicherten Koordinaten eines Geo-Objekts tragen oft amtlichen Charakter und dürfen nicht verändert werden. Die Kartenproduktion erfordert aber aus Gründen der Darstellung zusätzliche Präsentationsobjekte oder maßstabsabhängige Generalisierungen. Die Trennung von Geometrie und Präsentationsdaten erlaubt die Generierung mehrerer Kartenprodukte auf einer identischen Geodatenbasis und wird somit beiden Anforderungen gerecht.

Anwendungsunhabhängiger Rahmen

Das beschriebene Modell bildet einen anwendungsunabhängigen Rahmen für applikationsspezifische Modelle. Dieser Rahmen vereinfacht die Zusammenarbeit mehrerer Applikationen auf der selben Datenbasis oder die Verwendung einer Applikation auf mehreren nach diesem Schema gleich strukturierten Datenbasen. Wie am Beispiel der Generalisierung durch Präsentationsobjekte geschildert, ist er aber gleichzeitig flexibel genug, um noch Veränderungen und Erweiterungen zuzulassen.

Nachteil:

Datenmigration

Bestehende GIS-Daten weisen die Trennung in Geometrie, Sachdatenattribute und Darstellung nicht auf. Bevor die Daten genutzt werden können, ist eine aufwendige Neumodellierung und Migration bestehender Datenbestände notwendig.

Wertung:

Dieses Modell ist allgemein und übertragbar (Grundprinzip 1) und nutzt die Möglichkeiten der objektrelationalen Modellierung (Grundprinzip 2).

Die Geo-Objekte selbst sind einfache Tabellenzeilen. Die Geometrie ist in einer Spalte vom Typ SDO_Geometry abgelegt, wodurch die auf diesem Typ definierten Methoden direkt in Anfragen eingesetzt werden können. Die Geometrien sind nicht durch komplexere Modellierung als Attribute in Objekten versteckt. Durch die Trennung von der Präsentationsinformation bleiben die Geo-Objekte übersichtlich und anwendungsbezogen. Dadurch sind Anfragen einfach zu formulieren (Grundprinzip 3).

Das Modell verwendet ausschließlich die von dem verwendeten Datenbanksystem zur Verfügung gestellten Typen zur Speicherung von Geometrie (SDO_Geometry) und versucht nicht, weitere spezialisierte Geometrietypen abzuleiten. Auch wenn dies für eine saubere Modellierung von Vorteil wäre, bleibt so doch gesichert, daß das Modell einfach umzusetzen bleibt (Grundprinzip 4).

Durch die beschriebenen Eigenschaften und Vorteile ist dieses Modell eine gute Basis für die weitere anwendungsspezifische Modellierung. Der Anwender braucht sich nicht um eine Lösung für GIS-spezifische Dinge wie die Darstellung zu kümmern, sondern kann sich voll darauf konzentrieren, seine speziellen Anforderungen an Geo-Objekte in der Datenbank umzusetzen.

7.1.3 Entwicklung eines anwendungsspezifischen Applikationsmodells

Konzept:

Das Applikationsmodell ergänzt das oben beschriebene neutrale Modell um anwendungsspezifische Informationen.

Es erarbeitet die für eine Anwendung wichtigen raumbezogenen Objekte und modelliert sie als Geo-Objekt-Tabellen. Die Geometriespalte der Geo-Objekt-Tabelle trägt die Information über die Lage und Ausdehnung des Objekts. Weitere Attribute des Objekts werden in zusätzlichen Spalten der Geo-Objekt-Tabelle abgelegt.

Das applikationsspezifische Modell legt nach dem Muster des Basismodells eigene Präsentationstabellen an und verknüpft sie mit den Geo-Objekt-Tabellen. So können eine oder mehrere Präsentationen in Abhängigkeit vom Maßstab oder anderen, z. B. thematischen, Kriterien gespeichert werden.

Ausarbeitung am Beispiel:

Am Beispiel einer Landmanagement-Anwendung wird eine exemplarische Modellierung nach den oben genannten Konzepten erarbeitet. Für die Modellierung wurde auch auf die

z. T. noch unvollständigen ALKIS-Konzepte [ALKIS 99] zurückgegriffen. Das Modell konzentriert sich auf die raumbezogenen Objekte Flurstück, Gebäude und Bauteile, sowie Lage als ein nicht raumbezogenes Objekt, und deren Umsetzung innerhalb des oben beschriebenen Basismodells. Die Richtigkeit und Vollständigkeit des ALKIS-Ansatzes soll hier nicht diskutiert werden.

Wie schon allgemein gezeigt, werden für ein Geo-Objekt drei Arten von Informationen gespeichert: Metadaten über das Geo-Objekt, die Geo-Objekt-Tabelle selbst, sowie eine oder mehrere Präsentationen des Geo-Objekts. Diese werden hier am Beispiel des Geo-Objekts Gebäude vorgestellt.

1. Metadaten für Geo-Objekt Gebäude

   In der Tabelle ogis_geometry_columns werden alle in der Datenbank vorhandenen Geo-Objekte eingetragen. Zusätzlich speichert die Metadaten-Tabelle geoobject_presentations alle verschiedenen Darstellungen eines Geo-Objekts (s. Tab. 25). Das Beispiel zeigt wie die Geo-Objekt-Tabelle gebäude, die Style-Tabelle gebäude_styles und die Verknüpfungstabelle (geoobject_style_intersection_table) gebäude_presentations einander zugeordnet werden. Zusammen bilden sie eine mögliche Präsentation eines Geo-Objekts.

geoobject_presentations						
gp_id	descriptive_name	geoobject_table			style_table	geoobject_style_ intersection_table
		user	name	column		
1	Gebäude von München	alkis	gebäude	geometry	gebäude_styles	gebäude_presentations

Tab. 25: Die Metadaten-Tabelle geoobject_presentations mit einem Beispieleintrag.

2. Geo-Objekt-Tabelle für Geo-Objekt Gebäude

   Das Geo-Objekt Gebäude enthält die in Tab. 26 dargestellten Attribute. Das Beispiel zeigt drei Eintragungen für Wohngebäude (Gebäudefunktion 2100). Sie unterscheiden sich in der Anzahl ihrer Geschosse und ihrem Alter (Entstehungsdatum) voneinander.

Diese Unterschiede werden später noch für unterschiedliche Darstellungsarten genutzt.

gebäude									
g_id	objekt_id	objekt arten kennung	modell arten kennung	ent stehungs datum	version	gebäude funktion	name	ge schosse	geometry
1	geb1	GGB	DLKM	1.1.99	01	2100	Oly-dorf	20	...
2	geb2	GGB	DLKM	2.2.50	01	2100		3	...
3	geb3	GGB	DLKM	15.2.98	01	2100		3	...

Tab. 26: Die Geo-Objekt-Tabelle gebäude enthält die notwendigen Spalten g_id und geometry, sowie weitere beschreibende Attribute.

3. Präsentationsdaten für Geo-Objekt Gebäude

Die Tabelle gebäude_styles (s. Tab. 27) enthält einen Default-Style, der zur Darstellung der Gebäude verwendet wird, wenn kein anderer Style eingestellt ist (style_id: 1). Die Präsentationsinformation ist im Attribut geostyle, einem komplexen Datenbankobjekt vom benutzerdefinierten Typ geostyle_t enthalten. Beliebig viele weitere Styles können in die Tabelle eingefügt werden. Im Beispiel wurden zwei weitere Styles eingetragen, die zur besonderen Hervorhebung von Hochhäusern und Altbauten dienen (style_id: 2, 4).

gebäude_styles			
style_id	style_name	description	geostyle
1	default	voreingestellte Gebäudedarstellung (schwarze Umrandung, graue Fläche)	geostyle_t
2	Hochhaus	Hochhäuser (rote Umrandung, graue Fläche)	geostyle_t
3	Hausname	Hausnamen (schwarze Schrift, durchscheinender Hintergrund)	geostyle_t
4	Altbau	Altbau (rote Umrandung, blaue Fläche)	geostyle_t

Tab. 27: Die Style-Tabelle gebäude_styles mit Beispieleinträgen.

Sind den einzelnen Gebäuden in der Geo-Objekt-Tabelle keine Styles explizit durch Einträge in der Verknüpfungstabelle gebäude_presentations zugeordnet, so werden alle mit dem Default-Style dargestellt.

Die Nutzung der Verknüpfungstabelle erlaubt es aber, jedem Geo-Objekt eine oder mehrere Darstellungen zuzuordnen (s. Tab. 28). Im Beispiel wird dem Altbau (g_id = 2) der speziell dafür vorgesehene Style (style_id: 4) zugewiesen. Ebenso dem Hochhaus (g_id = 1) der Style 2. Die Eintragung des Styles 3 für das Hochhaus bewirkt außerdem die Darstellung seines Namens. Das Gebäude 3 ist weder ein Hochhaus, noch ein Altbau; es ist darum nicht eingetragen und wird im Default-Style dargestellt.

gebäude_presentations		
geoobject_id	style_id	sequence_number
2	4	1
1	2	1
1	3	2

Tab. 28: Die Verknüpfungs-Tabelle gebäude_presentations zeigt die Zuordnung von Styles zu Geo-Objekten.

Die in Tabelle gebäude_presentations gespeicherte Zuordnung der Styles zu den Geo-Objekten kann dynamisch nach vorgegebenen Regeln erzeugt werden. Die Regeln könnten in Form von SQL-Befehlen in der Style-Tabelle mit abgelegt sein. Die Regel für die Zuordnung der Altbaudarstellung (style_id = 4) könnte z. B. wie in Abb. 49 gezeigt formuliert werden.

```
1  INSERT    INTO gebäude_presentations
2  SELECT    g_id, 4, 1
3  FROM      gebäude
4  WHERE     entstehungsdatum < 1.1.1960
```

Abb. 49: Verknüpfung von Geo-Objekten und Styles mittels einer in SQL formulierten Regel

Wertung:

Die beim Basismodell genannten Vorteile bestätigen sich bei der Umsetzung am konkreten Beispiel der ALKIS-Implementierung. Die Geo-Objekt-Tabellen des Applikationsmodells lassen sich nahtlos in den Rahmen des Basismodells einfügen und zeigen die Machbarkeit des gewählten Ansatzes.

7.1.4 Migration des Datenbestands

Unter 7.1.2 war als Hauptnachteil des neuen Modells bereits die Notwendigkeit genannt worden, Daten aus bestehenden Datenbeständen migrieren zu müssen. Hier wird erörtert, warum eine Migration notwendig ist und wie sie durchgeführt wird.

Begründung:

Die Notwendigkeit einer Datenmigration begründet sich aus drei wesentlichen Unterschieden zwischen alter und neuer Modellierung (s. Tab. 29):

Erstens werden im Gegensatz zur alten bei der neuen Modellierung Objekte mit gleichartigen Attributen zu Geo-Objekten zusammengefaßt und in einer Geo-Objekt-Tabelle zusammen, aber getrennt von anderen Geo-Objekten, gespeichert.

Modellierungsvergleich		
Eigenschaft	alt	neu
Bildung und Trennung von Geo-Objekten	keine, alle Objekte zusammen in einer Tabelle	nur Objekte mit gleichen Eigenschaften zusammen in einer Geo-Objekt-Tabelle
Geometrie und Graphik	untrennbar	getrennt
Geometrie und Alpha-Attribute	getrennt	als Geo-Objekte in Tabelle zusammengefaßt

Tab. 29: Vergleich alter und neuer Modellierung

Bei der alten Modellierung erfolgt eine Trennung höchstens über verschiedene Ebenen innerhalb einer Geodatenbank, wobei sie aber nicht für das RDBMS sichtbar und somit bei Abfragen nutzbar ist. Ansonsten werden alle Geo-Objekte zusammen in einer Tabelle gehalten. Zweitens trennt das neue Modell Geometrie und Graphik voneinander, sie wird in unterschiedlichen Tabellen abgelegt und erst bei Bedarf für eine spezielle graphische Darstellung zusammengefügt. Bei der alten Modellierung sind Geometrie und Graphik untrennbar in einem Darstellungsobjekt miteinander verbunden. Und drittens ist im neuen Modell die frühere Trennung von Geometrie und alphanumerischen Attributen eines Geo-Objekts aufgehoben. Beide Attributarten werden jetzt zusammen in einer Tabelle gespeichert.

Aufgrund dieser wesentlichen Unterschiede zwischen alter und neuer Modellierung und ihrer Auswirkungen auf die Datenspeicherung ist es notwendig, alte Datenbestände in die neue Modellierung zu überführen. Dies beinhaltet folgende Schritte:

1. Geometriebildung
   Analyse der im alten GIS-Modell vorhandenen geometrischen Primitive[3] und Abbildung auf die zur Verfügung stehenden neuen Geometrie-Elemente der Datenbank, dabei Auflösung topologischer Datenstrukturen, wenn vorhanden.

2. Bildung von Geo-Objekten
   Zusammenführen der gebildeten Geometrien mit den vorher in mehreren Tabellen verstreuten Sachdaten in einer Geo-Objekt-Tabelle.

3. Präsentationszuordnung
   Analyse der vorher an den Elementen befindlichen Präsentationsinformationen und Design adäquater Styles im neuen Modell, Verknüpfen der Styles mit den Geo-Objekten.

Das Hauptproblem dabei ist: Noch liegen keine Ergebnisse darüber vor, ob und inwieweit sich diese Schritte automatisieren lassen. Ein hoher manueller Nachbearbeitungsaufwand verursacht aber hohe Kosten bei der Migration.

3. In SICAD z. B. PG, LI und LY.

Durchführung:

Die oben begründete und grob skizzierte Migration soll hier wieder am Beispiel des Geo-Objekts Gebäude vorgestellt werden. Sie wird in zwei Schritten durchgeführt. Der erste Schritt, der Export der Daten, erfolgt noch im alten GIS und hat eine Reihe von exportierten Dateien als Ergebnis. Diese Dateien sind die Verbindung und der Input für den zweiten Schritt, den Import in das neue System.

1. Umsetzung, manuelle Schritte, Export der Daten

   Geometriebildung

   Die Geometrie der neuen Gebäudeobjekte wird aus der Objektgeometrie der vorhandenen ALK-Objekte Gebäude gebildet. Dazu werden die bisher über topologische Verzeigerung über Linien und Punkte gespeicherten Flächen zu einem einzigen Polygon umgeformt, das alle seine Stützpunkte selbst enthält. Flächen mit Inseln erfordern zum Teil manuelle Nachbearbeitung.

   Bildung der Geo-Objekte

   Die weiteren Attribute des neu zu bildenden Geo-Objekts gebäude werden aus dem ALK-Objekt abgeleitet: Die Gebäudefunktion (gfk) läßt sich aus der Objektart des ALK-Objektes mit einer Zuordnungstabelle, die Anzahl der Vollgeschosse (gsz) aus der freien Beschriftung des ALK-Objekts bestimmen. Probleme treten hier jedoch bei eventuell mehrfachen unterschiedlichen Vollgeschoßbeschriftungen auf. Auch die Zuordnung des bisher in der Graphik gespeicherten Gebäudenamens erfordert eine manuelle Bearbeitung, da die Namen oft über mehrere Textstücke verstreut sind.

   Präsentationsanalyse

   Die an jedem einzelnen ALK-Gebäude-Objekt vorhandenen Darstellungsparameter werden analysiert und klassifiziert. Für die spätere Abbildung im neuen Modell ist eine Beschreibung der Klassen und ihrer Parameter wichtig. Dazu gehört, nach welchen Attributen (z. B. Nutzungsart) die Klassenbildung erfolgt und wie die einzelnen Klassen dargestellt werden (Liniendicke, -muster und -farbe, Flächenfüllung, etc). Die Präsentationsanalyse erfolgt im Moment manuell mit den Mitteln des alten GIS. Das Ergebnis ist eine tabellarische Übersicht der Klassen.

   Export

   Die neugebildeten Geometrien wurden im SQD-Format exportiert. Die anderen Attribute des Geo-Objekts nimmt zeitweise eine temporäre Tabelle mit den benötigten Spalten auf. Diese Tabelle wird mit Datenbankmitteln exportiert.

2. Import und Verknüpfung der Daten im neuen Datenmodell

Import

Die SQD-Datei wird mit einer modifizierten auf dem ORDBMS aufsetzenden Geodatenbank-Middleware importiert. Die exportierte Hilfstabelle wird mit Datenbankmitteln (SQL*Loader) importiert.

Geometriebildung, 2. Teil

Beim Import werden die alten GIS-Elemente in die neuen Geometrie-Elemente der Datenbank (SDO_Geometry) umgewandelt.

Bildung der Geo-Objekte, 2. Teil

Die Attribute in der importierten Hilfstabelle und die neuen Geometrie-Elemente werden mit SQL-Befehlen in eine Geo-Objekt-Tabelle zusammenkopiert. Damit ist das Geo-Objekt komplett.

Präsentationszuordnung

Die analysierten Klassen des alten Modells werden auf Styles des neuen Modells abgebildet. Dazu werden die erfaßten Parameter in Tabellen eingetragen. Die Verknüpfung der Styles mit den einzelnen Objekten in der Geo-Objekt-Tabelle erfolgt durch Eintragung der entsprechenden Schlüssel in die Verknüpfungstabelle.

7.2 Design und Implementierung

Nach der Transformation der Modellierung und dem Transfer der Daten aus dem bestehenden GIS in die OR-Datenbank liegen die Daten vor. Was noch fehlt, ist ein Werkzeug für den Anwender, um die Daten abfragen und die Ergebnisse anzeigen zu können. Das ORDBMS kann Ergebnisse nur tabellarisch, aber nicht graphisch anzeigen. Das ist bei Geodaten nicht ausreichend. Es wird ein GIS-Frontend gebraucht, das auf der Datenbank aufsetzt.

7.2.1 Architektur

Die Datenbank-Integrationsschicht verbindet das GIS-Frontend mit der Datenbank (s. Abb. 50). Sie stellt eine graphische Benutzerschnittstelle für die Eingabe der räumlichen Anfragen (Query-Builder) zur Verfügung, reicht die Anfragen an die Datenbank weiter und liefert das Ergebnis der Datenbank in alphanumerischen und geometrischen Teil getrennt an das GIS-Frontend. Das GIS-Frontend stellt die geometrischen Daten graphisch dar und zeigt die alphanumerischen Daten tabellarisch in einem weiteren Fenster an.

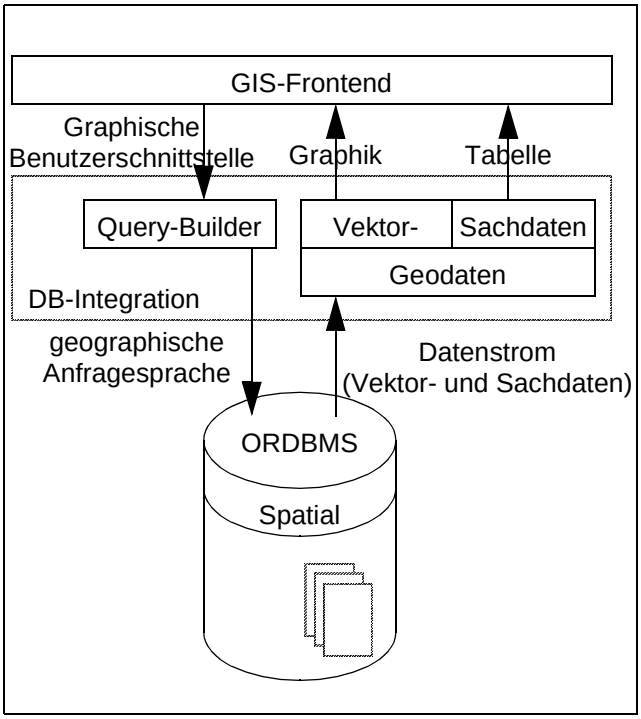


Abb. 50: Architektur einer GIS-Anwendung auf der Basis einer objektrelationalen Datenbank mit geographischem Erweiterungsmodul.

7.2.2 Datenbankdesign

Das Datenbankdesign setzt die in Kapitel 7.1.2 und 7.1.3 erarbeiteten Modelle in Tabellen der Datenbank um. Hier wird kurz und unter Auslassung von Details auf die wesentlichen Tabellen eingegangen. Die Listings sind in Anhang A.5.

Das applikationsunabhängige Basismodell besteht wie geschildert aus den drei Teilen Metadaten, Styles und Geo-Objekten. Die Metadatentabelle sicad_geoobject_types beschreibt die Arten von Geo-Objekten, welche im Modell verwendet werden dürfen. Auf diese Geo-Objekt-Typen wird in Tabelle sicad_geoobject_tables (s. Abb. 51) mit Attribut geoobject_type verwiesen. Der Constraint sic_geoo_tab_typ stellt sicher, daß nur definierte Typen Verwendung finden. Auf gleiche Weise sorgt der Constraint sic_geoo_tab_geo_col dafür, daß nur solche Geo-Objekte in sicad_geoobject_tables definiert sind, die auch wirklich im Datenbanksystem bekannt sind (d. h. eingetragen in ogis_geometry_columns). Die dritte Metadatentabelle sicad_geoobject_presentations ordnet jedem durch den Fremdschlüssel g_table_schema, g_table_name und g_geometry_column eindeutig referenzierten Geo-Objekt eine Style-Tabelle (style_table) und eine Verknüpfungstabelle (geoobject_style_inter_table) zu. Sie definiert auch einen voreingestellten Darstellungsmodus (default_style).

```
CREATE TABLE SICAD.SICAD_GEOOBJECT_TYPES (
   GEOOBJECT_TYPE                  NUMBER NOT NULL UNIQUE,
   DESCRIPTION                     VARCHAR2(256) NOT NULL,
   CONSTRAINT SIC_GEOO_TYP_IDX PRIMARY KEY (GEOOBJECT_TYPE));

CREATE TABLE SICAD.SICAD_GEOOBJECT_TABLES (
   G_TABLE_SCHEMA                  VARCHAR2(64),
   G_TABLE_NAME                    VARCHAR2(64),
   G_GEOMETRY_COLUMN               VARCHAR2(64),
   GEOOBJECT_TYPE                  NUMBER,
   DESCRIPTIVE_NAME                VARCHAR2(256),
   CONSTRAINT SIC_GEOO_TAB_IDX PRIMARY KEY

(G_TABLE_SCHEMA,G_TABLE_NAME,G_GEOMETRY_COLUMN),
   CONSTRAINT SIC_GEOO_TAB_TYP FOREIGN KEY(GEOOBJECT_TYPE)
         REFERENCES SICAD.SICAD_GEOOBJECT_TYPES(GEOOBJECT_TYPE),
   CONSTRAINT SIC_GEOO_TAB_GEO_COL
         FOREIGN KEY(G_TABLE_SCHEMA,G_TABLE_NAME,G_GEOMETRY_COLUMN)
         REFERENCES MDSYS.OGIS_GEOMETRY_COLUMNS
                  (G_TABLE_SCHEMA,G_TABLE_NAME,G_GEOMETRY_COLUMN));

CREATE TABLE SICAD.SICAD_GEOOBJECT_PRESENTATIONS (
   SGP_ID                  NUMBER NOT NULL UNIQUE,
   G_TABLE_SCHEMA                  VARCHAR2(64),
   G_TABLE_NAME                    VARCHAR2(64),
   G_GEOMETRY_COLUMN               VARCHAR2(64),
   STYLE_TABLE                     VARCHAR2(64) NOT NULL,
   GEOOBJECT_STYLE_INTER_TABLEVARCHAR2(64),
   DESCRIPTIVE_NAME                VARCHAR2(256),
   DEFAULT_STYLE                   REF SICAD.SICAD_STYLE_T,
   CONSTRAINT SIC_GEOO_PRE_IDX PRIMARY KEY (SGP_ID),
   CONSTRAINT SIC_PRE_GEOO_TAB
         FOREIGN KEY(G_GEOMETRY_COLUMN,G_TABLE_NAME,G_TABLE_SCHEMA)
         REFERENCES SICAD.SICAD_GEOOBJECT_TABLES
                  (G_GEOMETRY_COLUMN,G_TABLE_NAME,G_TABLE_SCHEMA));
```

Abb. 51: Erzeugung der Metadatentabellen des Basismodells

Zu den Metadatentabellen kommen Definitionen von abstrakten Datentypen für die Speicherung von Styles (s. Abb. 52), die später im Applikationsmodell Verwendung finden. Das Objekt sicad_geo_style_t definiert, welche Informationen ein Style beinhaltet. Es hat die fünf Bestandteile pointstyle, linestyle, areastyle, textstyle und symbolstyle, die ihrerseits wieder Objekte referenzieren, welche die eigentliche Darstellungsinformationen tragen. Sie werden hier nicht genauer ausgeführt. Der Typ sicad_style_t faßt die im sicad_geo_style definierten Darstellungsinformationen mit einem Namen, einer ID und einer Beschreibung zu einem Style-Objekt zusammen, das dann einem Geo-Objekt zugeordnet werden kann. Wie eine solche Zuordnung auszusehen hat, legt der abstrakte Datentyp sicad_geoobject_presentation_t fest: Die Geo-Objekt-ID (g_id) bestimmt eindeutig ein Geo-Objekt, das Attribut style referenziert ein Objekt des eben beschriebenen Typs sicad_style_t. Die so erstellte Zuordnung kann noch durch weitere Präsentationen, die durch die sequence_number durchnumeriert sind, er-

gänzt werden. Bringen manche Präsentationen eigene Darstellungskoordinaten mit, können diese im Geometrie-Objekt presentation_geometries abgelegt werden.

```
CREATE OR REPLACE TYPE SICAD.SICAD_GEO_STYLE_T AS OBJECT (
   POINTSTYLE                 REF SICAD.SICAD_POINT_STYLE_T,
   LINESTYLE                  REF SICAD.SICAD_LINE_STYLE_T,
   AREASTYLE                  REF SICAD.SICAD_AREA_STYLE_T,
   TEXTSTYLE                  REF SICAD.SICAD_TEXT_STYLE_T,
   SYMBOLSTYLE                REF SICAD.SICAD_SYMBOL_STYLE_T);

CREATE OR REPLACE TYPE SICAD.SICAD_STYLE_T AS OBJECT (
   STYLE_ID                   NUMBER,
   STYLE_NAME                 VARCHAR2(256),
   DESCRIPTION                VARCHAR2(256),
   GEOSTYLE                   SICAD.SICAD_GEO_STYLE_T);

CREATE OR REPLACE TYPE SICAD.SICAD_GEOOBJECT_PRESENTATION_T AS OBJECT (
   SGP_ID                     NUMBER,
   G_ID                       NUMBER,
   STYLE                      REF SICAD.SICAD_STYLE_T,
   SEQUENCE_NUMBER            NUMBER,
   PRESENTATION_GEOMETRIES    MDSYS.SDO_GEOMETRY);
```

Abb. 52: Erzeugung der abstrakten Datentypen für Styles

Die Geo-Objekte nutzen die bisher definierten Objekte und Tabellen, sind aber bereits Bestandteil des Applikationsmodells.

Die Umsetzung des Applikationsmodells in der Datenbank läuft für jedes einzelne Geo-Objekt nach dem gleichen Schema ab, das am Beispiel des Geo-Objekts Gebäude dargestellt wird:

Zuerst wird die Geo-Objekt-Tabelle erzeugt und für diese Geo-Objekt-Tabelle eine Style-Tabelle vom abstrakten Datentyp sicad_style_t abgeleitet (s. Abb. 53). Ebenso wird die Verknüpfungstabelle von sicad_geoobject_presentation_t abgeleitet. Dann werden die Geo-Objekte mit den Styles verknüpft, indem die jeweils zusammengehörigen Geo-Objekt-IDs und Style-IDs in die Verknüpfungstabelle eingetragen werden (siehe Abb. 49 oder auch Anhang A.5).

```
CREATE TABLE ALKIS.AL_GEBAEUDE
(
  G_ID                          NUMBER NOT NULL UNIQUE,
  OBJEKTID                      CHAR(7),
  OBJEKTARTENKENNUNG            CHAR(3),
  MODELLARTENKENNUNG            CHAR(8),
  ENTSTEHUNGSDATUM              CHAR(6),
  VERSION                       CHAR(2),
  GEBAEUDEFUNKTION              INT,
  NAME                          VARCHAR(255),
  GESCHOSSE                     INT,
  GEOMETRY                      MDSYS.SDO_GEOMETRY
);

CREATE TABLE ALKIS.AL_GEBAEUDE_STYLES OF SICAD.SICAD_STYLE_T;
CREATE TABLE ALKIS.AL_GEBAEUDE_PRESENTATIONS OF
SICAD_GEOOBJECT_PRESENTATION_T;
```

Abb. 53: Erzeugung eines Geo-Objekts des Applikationsmodells

Zum Schluß werden die Metadaten in die entsprechenden Tabellen des Basismodells eingetragen. Den Kern des Metadatenmodells bildet die Tabelle sicad_geoobject_presentations. Abbildung 54 zeigt, wie die Namen der Geo-Objekt-Tabelle und der Style- und Verknüpfungs-Tabelle (g_presentation_table) eingetragen werden und damit die Verknüpfung zwischen Styles und Geo-Objekten hergestellt wird. In Anhang A.5 ist aufgeführt, wie die anderen Tabellen gefüllt werden.

```
 DECLARE
          G_SCHEMA                VARCHAR2(64) DEFAULT USER;
          G_TABLE                 VARCHAR2(64) DEFAULT 'AL_GEBAEUDE';
          G_COLUMN                VARCHAR2(64) DEFAULT 'GEOMETRY';
          G_STYLE_TABLE           VARCHAR2(64) DEFAULT 'AL_GEBAEUDE_STYLES';
          G_PRESENTATION_TABLE  VARCHAR2(64) DEFAULT 'AL_GEBAEUDE_PRESENTATIONS';
          STYLE                   REF SICAD.SICAD_STYLE_T;
BEGIN
SELECT REF(S) INTO STYLE FROM ALKIS.AL_GEBAEUDE_STYLES S
   WHERE STYLE_NAME = 'DEFAULT';

 INSERT INTO SICAD.SICAD_GEOOBJECT_PRESENTATIONS VALUES
 (
   SICAD.SICAD_GEOOBJECT_PRESENTATION_IDS.NEXTVAL,
   G_SCHEMA,
   G_TABLE,
   G_COLUMN,
   G_STYLE_TABLE,
   G_PRESENTATION_TABLE,
   'Gebaeude DEFAULT',
   STYLE
 );
END
```

Abb. 54: Eintragung der Metadaten des Applikationsmodells

Außerdem können die einzelnen Geo-Objekt-Tabellen, wie andere relationale Tabellen auch, miteinander oder mit anderen Tabellen in der Datenbank durch Fremdschlüsselbeziehungen verknüpft werden. Diese Modellierung ist jedoch für jede Anwendung applikationsspezifisch umzusetzen und darum nicht Bestandteil dieser Arbeit.

7.2.3 Datenströme

Sind die im vorangegangenen Kapitel kennengelernten Tabellen einmal mit Daten gefüllt, so interessiert vor allem, wie sich das darauf aufbauende Gesamtsystem bei Anfragen verhält, d.h. wie es die Daten zusammenführt, auswertet und aufbereitet an den Client sendet.

Wichtig an diesem Datenfluß ist die klare Aufgabenteilung zwischen Datenbank und GIS. Während das GIS sich rein auf die Darstellung der Daten und die Interaktion mit dem Benutzer konzentriert, berechnet das DBMS die darzustellende Ergebnismenge räumlicher Objekte. Es benutzt dabei die Funktionen der räumlichen Erweiterung. Dem GIS fällt außerdem die Aufgabe zu, die Anfragen der Benutzer entgegenzunehmen und als SQL-Abfrage an das DBMS weiterzugeben.

Außerdem werden die in der Datenbank separat gespeicherte Geometrie und Präsentation erst zusammengeführt, wenn das Ergebnis von der Datenbank an das GIS weitergegeben wird.

8. Tests und Ergebnisse

Um das auf der objektrelationalen Datenbank basierende System mit dem auf einer relationalen Datenbank basierenden GQL vergleichen zu können, werden hier die in Kapitel 6 durchgeführten Tests auszugsweise wiederholt.

8.1 Sequoia 2000 Benchmark

Durchführung

Die Anfragen 5, 6, 7, 8 und 10 des Sequoia 2000 Benchmarks werden mit dem auf Oracle Spatial basierenden System durchgeführt. Rasteranfragen und Rekursion werden wieder ausgeklammert. Die wesentlichen Ausschnitte der SQL-Anfragen werden im folgenden kurz beschrieben. Die kompletten SQL-Skripts können in Anhang A.3 eingesehen werden.

ad 1. Erzeugen der Datenbank, Laden der Daten, Erzeugen der Indizes

Die Daten des S2K-Benchmarks werden mit AWK[4]-Skripts aus dem vorliegenden textuellen Format in Oracle Bulk Load Format[5] umgesetzt und mit dem SQL*Loader (sqlldr) in Tabellen geladen. Indizes werden entsprechend den Empfehlungen der Oracle Spatial Tuning Utilities (sdo_tune) aufgebaut, wobei hier zu berücksichtigen ist, daß die gleiche Indizierungstiefe für Tabellen verwendet wird, die später im räumlichen Verbund verschnitten werden sollen. Datenbankschema und Skripts finden sich in Anhang A.3.

ad 5. nicht-räumliche Selektion: Suche einen Punkt mit einem Namen n

Gesucht ist ein Punkt mit einem beliebig gewählten Namen (s. Abb. 55). Das ist eine reine Alphadaten-Abfrage, die noch keine besonderen Oracle Spatial Eigenschaften nutzt. Die Geo-Objekt-ID (g_id) wird in einer Ergebnis-Tabelle zwischengespeichert. Sie wird verwendet, wenn das Ergebnis auch graphisch angezeigt werden soll.

```
1  INSERT   INTO loc_result
2  SELECT   g_id
3  FROM     loc
4  WHERE    nam= ‚Boca Hill‘
```

Abb. 55: OSP-Lösung für Anfrage 5 aus Tabelle 15

4. Von Unix-Plattformen bekanntes Werkzeug, das einfache Bearbeitung von Textdateien erlaubt.
5. Ein textuelles proprietäres Austauschformat von Oracle, das schnelles Laden von Daten unterstützt.

ad 6. räumliche Selektion (mit Erzeugung einer neuen Tabelle): Suche alle Polygone, die sich mit einem Rechteck r überschneiden.

Gesucht sind alle Polygone, die sich mit einem Suchrechteck überschneiden. Das Suchrechteck kann in Oracle Spatial direkt als Geometrie in einer proprietären Syntax angegeben werden (s. Abb. 56). Der auszuführende Egenhofer-Operator, hier anyinteract, wird durch den Parameter mask der allgemeinen Funktion sdo_relate bestimmt. Die Geo-Objekt-IDs der gefundenen Landuse-Polygone werden in die Tabelle use_result eingefügt und stehen dort zur Steuerung der graphischen Ergebnisanzeige zur Verfügung.

```
1   INSERT    INTO use_result
2   SELECT    g_id
3   FROM      use u
4   WHERE     mdsys.sdo_relate(
5                 u.geometry,
6                 mdsys.sdo_geometry(3, null, null
7                     mdsys.sdo_elem_info_array(1,3,3),
8                     mdsys.sdo_ordinate_info_array(
9                         -1720000, -425000, -1707000, -416000)),
10                ,mask=anyinteract querytype=window') = ,TRUE';
```

Abb. 56: OSP-Lösung für Anfrage 6 aus Tabelle 15

ad 7. Kombination räumlicher Kriterien: Suche alle Polygone, innerhalb des Kreises k und einer Fläche größer x

Gesucht sind alle Polygone aus Tabelle use, die sowohl innerhalb eines durch drei Punkte definierten Kreises (Z. 6-11) liegen, als auch einen Flächeninhalt von mehr als 1 km^2 haben (s. Abb. 57). Ähnlich wie in der vorherigen Abfrage wird der Oracle Spatial Konstruktor sdo_geometry verwendet, um den Kreis für Oracle Spatial zur Verfügung zu stellen. Da die Flächen vollständig enthalten sein sollen, wird der Operator inside verwendet. Der Flächeninhalt wird mit der PL/SQL-Funktion sdo_area aus dem Package sdo_geom ermittelt.

```
1   INSERT    INTO use_result
2   SELECT    g_id
3   FROM      use u, sdo_geom_metadata md
4   WHERE     mdsys.sdo_relate(
5                 u.geometry,
6                 mdsys.sdo_geometry(3, null, null
7                     mdsys.sdo_elem_info_array(1,3,4),
8                     mdsys.sdo_ordinate_info_array(
9                         -1720000, -425000,
10                        -1707000, -416000,
11                        -1720000, -416000)),
12                ,mask=inside querytype=window') = ,TRUE'
13  AND       sdo_geom.sdo_area(u.geometry,md.diminfo)) > 1000000
14  AND       md.table_name = ,USE';
```

Abb. 57: OSP-Lösung für Anfrage 7 aus Tabelle 15

ad 8. Verbund zwischen Punkt- und Polygondaten: Gib die Landnutzung (landuse) aller Polygone innerhalb eines Rechtecks von 50 km um einen Punkt p aus

Gesucht sind alle Polygone, die innerhalb eines bestimmten Rechtecks um einen durch seinen Namen ausgewählten Punkt liegen. Ein solches Rechteck kann mit den Mitteln von Ora-

cle Spatial nicht innerhalb der SQL-Anfrage direkt berechnet werden, sondern muß in einer PL/SQL-Prozedur zuerst bestimmt und dann verwendet werden.

Abbildung 58 zeigt eine PL/SQL-Prozedur, die zuerst ein Rechteck der Größe rect_size um den Punkt mit dem Namen name berechnet (Z. 24-28). Die Größe und der Name werden durch Eingabe-Parameter der Prozedur festgelegt. Die Funktion sdo_buffer liefert das Rechteck als SDO_Geometry in die Variable BUFFER, die dann im zweiten Schritt verwendet wird, um alle Landuse-Polygone in diesem Rechteck zu finden (Z. 32-24).

```
15  CREATE OR REPLACE PROCEDURE „S2K“.QUERY_8_PROC ( name in varchar2, rect_size IN NUMBER)
16  IS
17                  BUFFER      MDSYS.SDO_GEOMETRY;
18                  result_size number;
19  BEGIN
20                  DBMS_OUTPUT.PUT_LINE('polygons within a rectangle of size',
21                        || rect_size || , around , || name || ,.');
22
23                  SELECT      sdo_geom.sdo_buffer(l.geometry, md.diminfo, rect_size)
24                  INTO        BUFFER
25                  FROM        loc l, sdo_geom_metadata md
26                  WHERE       l.nam = name
27                  AND         md.table_name = ,LOC‘;
28
29                  DELETE FROM use_result;
30
31                  INSERT      INTO use_result
32                  SELECT      u.g_id from use u
33                  WHERE       mdsys.sdo_relate(u.geometry, BUFFER,
34                                    ,MASK=INSIDE QUERYTYPE=WINDOW‘) = ,TRUE‘;
35
36                  commit work;
37
38                  SELECT count(*) INTO result_size
39                  FROM use_result;
40
41                  DBMS_OUTPUT.PUT_LINE('Found , || result_size || , polygons.');
42
43  END;
44  /
```

Abb. 58: PL/SQL-Prozedur für Anfrage 8 aus Tabelle 15

ad 10. Verbund zwischen Punkt- und Polygondaten (mit Erzeugen einer Tabelle): Ermittle die Namen aller Punkte, die innerhalb von Polygonen mit Landnutzung (landuse) liegen und lege sie in einer Tabelle ab

Gesucht sind alle Punkte innerhalb eines Polygons mit einer bestimmten Landnutzung (hier fkt = 83). Die Punkte werden mit dem Operator inside bestimmt (s. Abb. 59). Dazu wird ein räumlicher Verbund (spatial join) über die Tabellen der Punkte (loc) und der Landuse-Polygone (use) gerechnet.

```
1  INSERT    INTO loc_result
2  SELECT    l.g_id
3  FROM      loc l, use u
4  WHERE     mdsys.sdo_relate(
5                 l.geometry, u.geometry,
6                 ,mask=inside querytype=join') = ,TRUE'
7  AND       u.fkt = 83;
8
```

Abb. 59: OSP-Lösung für Anfrage 10 aus Tabelle 15

Ergebnisse und Wertung

Die Laufzeit der Anfragen faßt Tabelle 30 zusammen. Das direkte Laden in die Datenbank einschließlich Erzeugen der Indizes liegt mit einer Laufzeit im Bereich von Minuten im unkritischen Bereich.

Einfache nicht-räumliche Selektionen (Anfrage 5) stellen, wie für ein datenbankbasiertes System erwartet, kein Problem dar.

Räumliche Selektionen nutzen den räumlichen Index gut aus und skalieren in ihrer Laufzeit linear zur Größe des Selektionsbereichs. Probleme kann es hier bei zu großen Suchbereichen geben, da dann Oracle 8.1.5 Spatial beim Aufbau eines Index für den Suchbereich abstürzt. Die Verknüpfung mit weiteren nicht-räumlichen Kriterien ist ohne Probleme möglich.

Gruppe	Nr.	Anfrage	Laufzeit/sec
			OSP
Daten laden	1	Laden	1450[a]
Einfache Punkt- und Polygonanfragen	5	nicht-räumliche Selektion	1
	6	räumliche Selektion	172[b]
	7	kombinierte Selektion	303[c]
Räumlicher Verbund (spatial join)	8	Punkt- und Polygondaten mit oder ohne Erzeugen einer Tabelle	2-10[d]
	10		110[e]

Tab. 30: Die Ergebnisse des Sequoia 2000 Benchmarks mit OSP

a. Laden loc+use + Index erzeugen loc+use = 509+557 + 96+288 sec
b. Wert bei 10175 Flächen in der Treffermenge, sonst bei kleinen Rechtecken < 1 sec, bei größeren linear proportional zur Anzahl der Flächen in der Ergebnismenge ansteigend, ca. 60 El/sec
c. Wert bei 2578 Flächen in der Treffermenge, sonst bei kleinen Rechtecken < 5 sec, bei größeren linear proportional zur Anzahl der Flächen in der Ergebnismenge ansteigend, ca. 10 El/sec
d. je nach Größe des gewählten Kreises
e. Wert bei 909 zu überprüfenden Polygonen, sonst überproportionaler Anstieg bei mehr zu prüfenden Polygonen

Oracle Spatial meistert in der Regel auch den räumlichen Verbund auf den einfachen S2K-Benchmark-Daten mit guter Performance.

Die Anwendung des S2K-Benchmarks erlaubt bereits Aussagen über die Performance von Oracle Spatial bei einfach strukturierten Daten und Anfragen. Genauere Aussagen über die Benutzbarkeit mit realen Anwenderdaten zeigt das im nächsten Abschnitt wiederholte Anwenderszenario.

8.2 Digitale Stadtgrundkarte

Die in Kapitel 6.2.2 vorgestellte Anfrage nach einem für einen Kindergarten geeigneten Grundstück wird hier wiederholt, um die Eignung von Oracle Spatial für komplexere Anfragen mit mehreren Bedingungen zu prüfen. Die Anfrage wird auf Landmanagementdaten der Stadt Berlin durchgeführt, die an ALKIS angelehnt in Oracle Spatial modelliert und in Ausdehnung und Datendichte den in Kapitel 6.2.2 verwendeten Daten der Stadt München vergleichbar sind. Die Bedingungen

1. größer als 2000 m^2 zu sein,
2. der Stadt zu gehören und
3. mehr als 100 m entfernt
4. von Bundesstraßen zu sein

werden wie folgt umgesetzt.

Die Bestimmung des Flächeninhalts kann direkt im Bedingungsteil der Hauptanfrage (siehe Z. 11-20 in Abb. 60) durch die Funktion sdo_area des Oracle Spatial Pakets sdo_geom abgefragt werden. In Ermangelung eines vorhandenen Eigentümerdatenbestands wurde die Frage nach dem Eigentumsverhältnis auf die tatsächliche Nutzung (fkt = 2920) abgebildet, was nach dem ALKIS-Objektartenkatalog Grundstücken entspricht, die unbebaut sind oder leerstehende Gebäude tragen. Die im Testdatenbestand vorhandene Eisenbahntrasse wurde anstelle einer Bundesstraße benutzt. Dazu wird mit der Funktion sdo_buffer ein Puffer von 100 m um die Bahnlinie (fkt = 5400) gelegt und in Tabelle bahn_buffer abgelegt (Z. 4-7). Für die effiziente Durchführung der Hauptanfrage wird ein Index auf dieser Tabelle angelegt (Z. 8-10). Die Puffer-Tabelle wird dann in der Hauptanfrage über eine Unteranfrage eingebunden (Z. 17-20). Als Gesamtergebnis werden die g_ids der gefundenen Flächen in einer Tabelle geoobject_set abgelegt und das GIS zeigt die dort eingetragenen Flächen an.

```
1   DEFINE      dim_n = „mdsys.sdo_dim_array(
2                   mdsys.sdo_dim_element('X',10000,40000,0),
3                   mdsys.sdo_dim_element('Y',10000,40000,0))“

4   CREATE      TABLE bahn_buffer AS
5   SELECT      g_id, sdo_geom.sdo_buffer(n.geometry, &dim_n, 100) geometry
6   FROM        ntn n
7   WHERE       n.fkt = 5400;

8   CREATE          INDEX bahn_buf_spt_idx ON bahn_buffer(geometry)
9   INDEXTYPE       IS MDSYS.SPATIAL_INDEX
10  PARAMETERS  ('SDO_LEVEL=10, SDO_NUMTILES=12');

11  INSERT      INTO geoobject_set
12  SELECT      n1.g_id
13  FROM        ntn n1
14  WHERE       n1.fkt = 2920
15  AND         sdo_geom.sdo_area(n1.geometry, &dim_n) BETWEEN 1000 AND 5000
16  AND         n1.g_id NOT IN (
17                  SELECT      n2.g_id
18                  FROM        ntn n2, bahn_buffer b
19                  WHERE       mdsys.sdo_relate(b.geometry, n2.geometry,
20                                      'mask=anyinteract querytype=join' = 'TRUE');
```

Abb. 60: Oracle Spatial Lösung für eine Standortanalyse

Randbedingungen

Wie oben gezeigt, ist die Anfrage mit den Mitteln von Oracle Spatial durchführbar, wenn auch nicht so direkt und einfach, wie ursprünglich vorgestellt. Die Ursachen dafür liegen in einer Reihe von durch Oracle Spatial bestimmten technischen Randbedingungen, die im folgenden angesprochen werden sollen:

1. Die Funktionen sdo_area und sdo_buffer benötigen als Eingabeparameter das sdo_dim_array, das den Wertebereich der Koordinaten der berechneten Geometrie in X- und Y-Richtung festlegt. Diese Informationen liegen in der Tabelle sdo_geom_metadata vor und könnten eigentlich durch einen Join direkt in die jeweiligen Abfragen mit eingebunden werden. Enthalten diese Abfragen aber noch mehr Bedingungen, so wird dadurch der Oracle Optimierer so verwirrt, daß um Klassen schlechtere Antwortzeiten die Folge sind. Darum wurde das oben abgebildete Vorgehen gewählt und das benötigte sdo_dim_array in einer Variable definiert, die dann in den Anfragen verwendet wird.
2. In Oracle Spatial fehlt die eigentlich vom OGC vorgesehene Funktion distance, welche die Entfernung zweier Geometrien voneinander berechnet. Die als Ersatz vorgesehene Funktion within_distance kann nur bestimmen, welche Geometrien **weniger** als eine gegebene Entfernung von einer festgelegten Geometrie entfernt sind. In der Aufgabenstel-

lung ist aber gefragt, welche Geometrien **mehr** von einer festgelegten Geometrie entfernt sind. Der einfache Schluß, die Bedingung mit NOT zu negieren, ist aus Gründen der internen Bearbeitung in Oracle nicht zulässig und führt zu einem Fehler.
Außerdem ist within_distance nicht geeignet, einen räumlichen Verbund über zwei Tabellen mit jeweils variablen Geometrien zu berechnen, da die von Spatial verwendete Methode, über einen Nested-Loop für jeweils eine Geometrie der einen Tabelle die Entfernungen zu den anderen Geo-Objekten zu bestimmen, die bestehenden Indizes nicht ausnutzt bzw. nicht von geeigneten Indexstrukturen unterstützt wird.
Abhilfe läßt sich dadurch schaffen, daß um alle Geometrien der einen Tabelle (am besten der kleineren) ein Puffer der gewünschten Entfernung gelegt wird und diese neuen Geometrien in einer Tabelle zwischengespeichert werden. Das ermöglicht, einen Join über diese Hilfstabelle und die andere Tabelle zu rechnen und dabei das distance-Problem auf den Operator anyinteract zu reduzieren, der von Spatial gut unterstützt wird, wenn geeignete Indizes vorliegen.

3. Darum muß in der Folge auf der Hilfstabelle, im obigen Beispiel bahn_buffer, ein Index erzeugt werden. Die Werkzeuge des sdo_tune-Package von Spatial berechnen empfohlene Werte für Parameter sdo_level und sdo_numtiles. Mit diesen Werten kann ein optimal auf die jeweiligen Geometrien zugeschnittener Index erzeugt werden. Das Problem liegt darin, daß zum einen für Tabellen mit unterschiedlichen Geometrien **unterschiedliche** Werte für einen optimalen Index empfohlen werden, zum anderen aber **gleiche** Werte für die optimale Abarbeitung eines räumlichen Verbunds benötigt werden. Aus diesem Grund wurden der Index im obigen Beispiel mit den Werten der zu verknüpfenden Tabelle erzeugt und nicht mit den eigentlich von sdo_tune vorgeschlagenen.
4. In der eigentlichen Hauptabfrage wird in Z. 16 als Ersatz für die eigentliche distance-Bedingung eine mit NOT negierte Unteranfrage verwendet. Die selbe Bedingung hätte auch als join formuliert werden können, was die Performance aber negativ beeinflußt.

Laufzeit

Verwendet man die Folge von optimal aufeinander abgestimmten Befehlen, wie im angegebenen Beispiel liegt das Ergebnis nach weniger als 10 Sekunden vor. Experimente mit geringfügigen Abweichungen, wie z. B. eines höheren Wertes für die gewählte Entfernung oder eines Joins an Stelle einer Subquery, zeigten Laufzeiten von 1000 Sekunden oder führten gar zu einem Abbruch mit Fehler.

Vorteile

Die gestellte Aufgabe kann mit Oracle Spatial mit akzeptabler Performance gelöst werden.

Nachteile

Bei der Bearbeitung der etwas komplexeren Anfrage zeigen sich folgende Nachteile:

- Die Anfrage ist nicht in einem SQL-Befehl zu formulieren, sondern muß auf mehrere Teilbefehle aufgeteilt werden, daß entspricht einer Optimierung von Hand durch den Anwender.
- Die Berechnung und Zwischenspeicherung von explizit erzeugten Puffer-Geometrien kostet Zeit und Speicherplatz.
- Einer der beiden bei dem räumlichen Verbund benutzten Indizes ist suboptimal.
- Vom Anwender bzw. Applikationsentwickler nicht optimal gestellte und getunte Anfragen resultieren in erheblichen Performanceeinbrüchen.

Wertung

Komplexere Anfragen mit mehreren räumlichen Bedingungen zeigen die Grenzen der aktuellen Technologie von Oracle Spatial. Obwohl die zum Endergebnis führenden Einzelschritte schnell Zwischenergebnisse liefern, kann doch die Anfrage nicht in einem einzigen SQL-Befehl abgesetzt werden. Das legt die Folgerung nah, daß die relativ neue Technologie von Oracle Spatial noch nicht optimal in die gesamte Datenbank eingebunden ist. Insbesondere die Optimierung komplexer Anfragen ist noch nicht zufriedenstellend gelöst und bleibt damit dem Anwender überlassen. Dies setzt beim Anwender tiefgehendes Verständnis der Vorgänge in der Datenbank voraus.

8.3 Zusammenfassung

Auf der Basis einer um eine räumliche Komponente erweiterten objektrelationalen Datenbank wurde ein allgemeines Datenmodell entworfen, das Geo-Objekt-Tabellen gemäß der OGC Simple Features Spezifikation und zusätzlich geeignet verknüpfte Präsentationsinformationen zur Verfügung stellt. Das allgemeine Modell wurde zu einem anwendungsspezifischen Applikationsmodell verfeinert und in einem Prototypen umgesetzt.

Design, Entwicklung und Test des Prototypen bestätigten die beschriebenen Vorteile des gewählten Ansatzes: die saubere Trennung von Geometrie und Präsentation und die Zusammenführung der Geometrie und der weiteren Attribute eines Geo-Objekts in einer Tabelle. Die Tests mit dem Sequoia 2000 Benchmark ergaben, daß sich einfache Anfragen auf einer Geo-Objekt-Tabelle leicht und performant umsetzen lassen.

Die Erfahrungen mit komplexeren Anfragen zeigen die Einschränkungen der verwendeten Datenbank auf und liefern wertvolle Erkenntnisse für die Entwicklung von GIS-Applikationen und die Modellierung von GIS-Daten mit objektrelationalen Datenbanken: die Beschränkung auf einfache Geometrien erlaubt den Einsatz einer erweiterten Standarddatenbank, die Bedeutung der Graphik wird gegenüber jener der Datenbank zurückgedrängt, Präsentationsinformationen werden den Daten nur noch bei Bedarf hinzugefügt.

Darüber hinaus liefert die Umsetzung bestehender Daten in das neue Datenmodell wertvolle Erfahrungen über die Migration alter Datenbestände beim Einsatz objektrelationaler Datenbanken.

9. Vergleich und Bewertung

Vergleich

In der Arbeit geht es darum, eine GIS-Applikation mit einer integrierten räumlichen Anfragesprache aufzubauen. Sie beschreitet dazu zwei unterschiedliche Wege: Während der erste eine Abfragesprache direkt in ein GIS einbaut, nutzt der zweite die Abfragesprache eines erweiterten DBMS. Die beiden Lösungen lassen sich auf der Basis nachfolgender Kriterien miteinander vergleichen. Dabei geht es um eine umfassende Bewertung der beiden Ansätze aus der Sicht eines Anwenders und Anwendungsentwicklers von GIS, die über den reinen Vergleich von Performance-Meßdaten hinausgeht. Folgende Liste beschreibt verschiedene Kriterien. Zu jedem Kriterium gibt es vier Abschnitte: der erste Abschnitt definiert das Kriterium, die beiden folgenden beschreiben, in wie weit die beiden Lösungsansätze GQL und ORDBMS das Kriterium erfüllen, der letzte Abschnitt vergleicht und bewertet die Lösungsansätze.

1. Implementierung

   Definition: Die Implementierung umfaßt den Aufwand zur ersten Erstellung des Gesamtsystems aus GIS und Datenbank und dessen Weiterentwicklung.
   GQL: Der Einbau von GQL in das GIS erfordert die komplette Definition und den Test der räumlichen Operatoren. Darüber hinaus werden räumlichen Anfragen im Client bearbeitet. Das erlaubt aber auch die optimale Anpassung an das bestehende System und Speziallösungen.
   ORDBMS: Das erweiterte DBMS stellt die Anfragesprache zur Verfügung. Sie muß einerseits nicht selbst entwickelt und definiert werden, andererseits ist der Anwender aber auch auf den Funktionsumfang der Sprache beschränkt. Spezielle Anforderungen lassen sich nicht so einfach einbauen, wie bei der selbst entwickelten und ins GIS integrierten Sprache. Der Client ist völlig von der Anfrageverarbeitung entlastet, sie läuft völlig im Server ab. Setzt ein neues Release des Servers weiterentwickelte Methoden zur Verfügung, wie z. B. einen R-Baum als weitere Indizierungsmethode, so steht diese fast ohne eigene Aufwendungen zur Verfügung.
   Wertung: Für einen GIS-Hersteller kann GQL nur eine Übergangslösung sein. Auf lange Sicht kann so aber nicht mit der erwarteten rasanten Weiterentwicklung objekt-relationaler DBMS mitgehalten werden.

2. Anfrageformulierung

   Definition: Wie einfach und anwendernah lassen sich Anfragen formulieren.
   GQL: GQL hält sich bei den verwendeten Operatoren eng an den OGC-Standard und unterstützt z. B. den bei Oracle Spatial fehlenden distance-Operator. Außerdem erlauben einfache Konstruktoren wie circle oder rectangle die komfortable und anwendernahe Ein-

gabe von Anfragen.

ORDBMS: Die Anfragen auf der Basis des erweiterten ORDBMS sind z. T. nicht so anwendernah. Benutzerdefinierte Objekte lassen sich über den SDO_Geometry-Konstruktor zwar definieren, diese sind aber eher technik-, als anwenderorientiert. Auch die Verwendung des allgemeinen Relate-Operators anstelle von ausformulierten Operatoren für jeden Egenhofer-Operator erleichtert zwar den Datenbankdesignern die Arbeit, kommt dem Anwender aber nicht entgegen. Die Schnittstelle ist hier wohl auch weniger für den Endanwender gedacht, sondern eher als Basis für Applikationsprogrammierer.

Wertung: Den einfachen Formulierungsmöglichkeiten in der Anfragesprache GQL sieht man an, daß bei ihrer Entwicklung Anwenderinteressen eine große Rolle gespielt haben. Die zum Teil eher technisch orientierten Sprachkonstrukte des ORDBMS sind schwieriger zu merken und einzugeben, sind aber für ein Anwendungsprogramm vielfältiger parametrisierbar. Da die Sprache in der Regel unter der graphischen Benutzeroberfläche des Applikationsprogramms verborgen ist, wiegt der Vorteil der besseren Parametrisierbarkeit des ORDBMS schwerer als die leichtere Formulierung von GQL.

Stimmt allerdings die Modellierung nicht, so sind in beiden Sprachen Anfragen sehr komplex, wenn nicht sogar unmöglich.

3. Speicherung von und Zugriff auf Objekte

Definition: Wie unterstützt die Modellierung den Zugriff auf einzelne Objekte.

GQL: Bei dem GQL zu Grunde liegenden Datenbankschema werden alle Objekte in einer einzigen Tabelle gespeichert. Dort sind sie in BLOBs versteckt. Zum Zugriff auf einzelnen Objekte muß immer die komplette Tabelle durchsucht werden. Dazu werden die BLOBs in den Client geladen, interpretiert und ggf. wieder verworfen bis alle abgearbeitet sind

ORDBMS: Im ORDBMS werden gleichartige Objekte in einer Geo-Objekt-Tabelle zusammengefaßt. Ihre Attribute liegen als Spalte dieser Tabelle für jeden zugänglich offen. Zum Zugriff auf ein Objekt muß nur diese Tabelle durchsucht werden. Dies geschieht auf dem Server. Nur die Ergebnisse werden auf den Client übertragen.

Wertung: Die Daten gemäß ihrer Modellierung auch getrennt in Tabellen zu speichern erlaubt effizienteren, stärker selektiven Zugriff und ist dem alten Ansatz deutlich überlegen.

4. Anfragebearbeitung und Optimierung

Definition: Was passiert bei der Bearbeitung einer Anfrage im System.

GQL: Die Anfrage wird vom Client auf räumliche Anteile hin untersucht. Die gefundenen räumlichen Operatoren werden getrennt vom Client in Datenbankoperationen umgesetzt und bearbeitet. Das Gesamtergebnis wird durch einen Join über alle Teilergebnisse berechnet.

GQL selbst verfügt über einfache heuristische Optimierungsverfahren, welche die Rei-

henfolge der Abarbeitung der einzelnen Unteranfragen bestimmen. Eine Optimierung über die gesamte Anfragen kann auf Grund der Aufteilung der Bearbeitung zwischen Client und Server nicht erfolgen.

ORDBMS: Die Anfrage wird als Ganzes an den Server weitergereicht. Der Server verwendet die räumliche Erweiterung und ihre Möglichkeiten und berechnet das Gesamtergebnis.

Für die Zerlegung in einzelne Teilschritte ist eine Optimierung über die gesamte Anfrage möglich, wenn auch zum Teil noch eingeschränkt, wie einzelne Experimente zeigen.

Wertung: Auch wenn die Optimierung im ORDBMS zum Teil noch Mängel hat und nicht immer zu optimalen Ausführungsplänen führt, ist dieser Ansatz der Aufteilung der Bearbeitung zwischen Client und Server deutlich überlegen.

5. Integration der Anfragesprache

   Definition: Bezeichnet die Art und Weise, wie die Anfragesprache integriert ist.

   GQL: Die Anfragesprache ist im GIS integriert. Sie kann über die Variablen der Prozedursprache gut mit anderen Teilen des GIS zusammenarbeiten.

   ORDBMS: Die Anfragesprache ist Teil der Datenbank und wird auf der Applikationsebene im GIS verfügbar gemacht. Dadurch ist es schwerer, mit anderen Prozeduren im GIS zu interagieren. Diese müssen entweder ihre Ergebnisse in eine temporäre Tabelle in der Datenbank übertragen um sie für die Anfragesprache verfügbar zu machen oder sie als Parameter übergeben.

   Wertung: Eine gute Integration sowohl mit dem GIS als auch mit der Datenbank könnte mit der Integration eines ORDBMS in den Geodatenserver erreicht werden.

6. Datenkonvertierung und -transfer

   Definition: Hier wird berücksichtigt, ob für die Nutzung der Anfragesprache Daten konvertiert und evtl. in ein anders System transferiert werden müssen.

   GQL: Die Anfragesprache GQL ist so in das GIS integriert, daß sie direkt auf die Daten des GIS zugreifen kann. Dafür müssen keine Daten umgesetzt werden. Eine Bedingung ist allerdings, daß der Detaillierungsgrad der Daten und die Denkebene des Benutzers zusammenpassen, damit die Sprache optimal eingesetzt werden kann. Sonst ist auch bei diesem Ansatz zumindest teilweise eine Neumodellierung und damit Konvertierung notwendig.

   ORDBMS: Der zweite Ansatz integriert ein erweitertes Datenbanksystem in das GIS. Das DBMS stellt die Anfragesprache zur Verfügung. Die Daten liegen im alten System vor. Es gibt noch keine ausgereifte GIS auf der Basis der neuen Technologie. Das macht es notwendig, das alte System zur Erfassung und Fortführung beizubehalten. Die Daten werden für Analysen oder Präsentationszwecke in das neue System übertragen. Hier ist eine Anpassung des Datenmodells an das erweiterte DBMS und seine Möglichkeiten zwin-

gend notwendig.

Wertung: Die Notwendigkeit Daten konvertieren und in das neue System transferieren zu müssen, stellt wohl die größte Hemmschwelle für seinen Einsatz dar. Erst nach einer Lösung dieses Problems werden auch Altkunden mit großen GIS-Datenbeständen die neue Technologie nutzen.

7. Modellierung

   Definition: Hierzu gehört, ob zur Lösung der Anfragen eine Neumodellierung der Daten notwendig ist.

   GQL: Die Integration der Anfragesprache in das GIS erlaubt, die Modellierung der bestehenden Daten weitgehend beizubehalten. Die Anfragesprache greift über die GeoDB-Middleware auf die im GIS vorhandenen Datenstrukturen zu und berechnet unter ihrer Nutzung die Ergebnisse.

   ORDBMS: Setzt man dagegen auf einem erweiterten DBMS auf, so erfordert dies, um es sinnvoll einzusetzen, auch die bestehende Datenmodellierung an die Möglichkeiten des DBMS und die Datentypen und Operatoren der räumlichen Erweiterung anzupassen.

   Wertung: Manche Modellierungsmöglichkeiten objektrelationaler Datenbanken wie z. B. sets erschweren den Zugriff mit der Anfragesprache und machen ihn ohne den weiteren Ausbau der Datenbankerweiterung unmöglich. Die Erweiterung des DBMS kann dem Anwender aufgrund der hohen Komplexität nicht empfohlen werden. Stattdessen bietet ein direkter Einsatz der vorgefertigten Typen auf oberster Ebene für Tabellenspalten die besten Möglichkeiten. Er erlaubt, mit den vorgesehenen Operatoren auf die Daten zuzugreifen und stellt damit guten Nutzen für nur geringen Aufwand.

8. Interne Datenspeicherung: Proprietäres vs. offenes vs. standardisiertes Format

   Definition: Wie werden die Geodaten und damit zusammenhängende Attribute gespeichert.

   GQL: Bei GQL sind die Daten in den Blobs in einem proprietären, nicht offengelegten Format versteckt. Die Sachdaten sind im RDBMS modelliert und über Schlüssel (location, elementid) mit den graphischen Daten verknüpft. Auf die Elemente kann mit den Operatoren der Sprache zugegriffen werden, die über das GIS realisiert ist. Das RDBMS selbst kennt aber die Daten, ihren Typ und ihre Eigenschaften nicht und kann deshalb auch nicht optimal darauf zugreifen oder über die Daten Indizes bilden.

   ORDBMS: Beim zweiten Ansatz sind die Geodaten mit den Alphadaten in einer Tabelle in einem proprietären, aber offengelegten Format gespeichert. Die Verknüpfung entfällt dadurch. Auf die Geodaten kann mit den Operatoren der Sprache zugegriffen werden, die hier Bestandteil des erweiterten DBMS ist. Das DBMS kennt also die Datentypen und kann mit ihnen umgehen. Die Geometrie ist für externe Applikationen offen zugänglich, für auf Java basierende Anwendungen auch in einem OGC-konformen Interface, anson-

sten mit dem proprietären Format von Oracle Spatial.

Wertung: In einer Zeit, in der GIS immer stärker mit anderen Anwendungen vernetzt werden, spielen offene Formate und Schnittstellen eine entscheidende Rolle. Standardisierte offene Systeme werden sich gegenüber proprietären Systemen durchsetzen.

9. Datenströme

   Definition: Welche Datenströme fließen bei der Bearbeitung einer Anfrage.

   GQL: Die Abfrage wird aus Datenbanksicht auf der Ebene der Applikation ausgewertet. Das erfordert teure Transfers von Daten aus der Datenbank in die Applikation, die dann von den Filterschritten wieder verworfen werden. Dieser Ansatz ist nur bei guter Vorselektion durch den Anwender oder sehr geringen Datenmengen tragfähig.

   ORDBMS: Die Operatoren werden innerhalb der Datenbank ausgewertet. An das DBMS wird die Anfrage weitergeleitet. Nur die Ergebnisse werden an die Applikation zurückgereicht.

   Wertung: Da bei Anfragen, die keine Sachdateneinschränkung haben (ganzes Plangebiet) potentiell die ganze Datenbank durchsucht wird, ist das ein wesentlicher Unterschied.

10. Aufgabe des Anwendungsentwicklers

    Definition: Welche Aufgabe hat der Anwendungsentwickler.

    GQL: Der GIS-Anwendungsentwickler kann auf die vertrauten Werkzeuge des GIS zurückgreifen, die durch GQL ergänzt werden.

    ORDBMS: Der GIS-Anwendungsentwickler muß umdenken und seine Sicht von einer GIS-zentrischen Sicht zu einer datenbank-orientierten verändern.

    Wertung: Ein Anwendungsentwickler, der keine volle GIS-Funktionalität, wie z. B. Elemente zu erzeugen und zu verändern, braucht, sondern eine reine Analyse und Anzeige-Applikation aufbauen will, ist mit der Funktionalität eines erweiterten DBMS gut bedient. Das Problem der Neumodellierung und Übertragung der Daten bleibt allerdings bestehen, es muß in diesem Fall vom Anwender bewältigt werden. Ist dagegen die volle GIS-Funktionalität notwendig und sollen nur gelegentlich Analysen getätigt werden, so ist es vorzuziehen, eine Applikation auf einem GIS aufzusetzen. Die Anfragesprache sollte dann, wie das bei GQL der Fall ist, auch voll im GIS integriert sein, um z. B. mit ihm Variablenwerte problemlos austauschen zu können.

Zieht man alle diese Kriterien zu Rate, so läßt sich bewerten, wann welche Architektur ihre Vorteile hat. GIS mit den traditionellen Aufgaben der Erfassung und Fortführung können weiterhin auf den bewährten Architekturen aufsetzen, die das effiziente Laden und Speichern von lokalen Ausschnitten unterstützen. Die Verarbeitung findet hauptsächlich im GIS statt, eine Anfragesprache kann ergänzend von Nutzen sein.

Treten dagegen Analysen über den gesamten Datenbestand oder die Integration mit anderen Unternehmensdaten in den Vordergrund, bieten sich erweiterte DBMS mit einer räumlichen Anfragesprache als Grundlage an. Modelle mit einfachen Geometrien und geringen topologischen Beziehungen zwischen den Elementen lassen sich mit ihnen schnell realisieren. Bei komplexeren Anwendungen, wie z. B. der Umsetzung eines ALKIS-Modells fehlen noch komplexere Datentypen und GIS-spezifische Modellierungsmöglichkeiten wie z. B. für Topologie. Hierfür gibt es eventuell aber auch alternative Lösungen mit Datenbankmitteln, wie z. B. Objektreferenzen, die navigierenden Zugriff erlauben. Dazu liegen aber noch keine GIS-spezifischen Untersuchungen vor.

Zusammenfassend lassen sich die beiden Ansätze wie in Tabelle 31 dargestellt vergleichen und bewerten.

Kriterien	Beurteilung	
	erweitertes DBMS	GQL auf RDBMS
1. Implementierung	setzt auf ORDBMS auf, höherer Startaufwand, profitiert danach von Fortschritten der DB	setzt auf RDBMS und GeoDB-Middleware auf, niedriger Startaufwand, aber Folgekosten für Performance
2. Anfragen	leicht umzusetzen, performant	leicht umzusetzen, z. T. unperformant
3. Integration	DB-nah, die Sprache wird von der DB zur Verfügung gestellt, besser bei globalen Anfragen	GIS-nah, die Sprache ist ins GIS eingebaut, evtl. besser bei lokalen Anfragen,
4. Datentransfer	aufwendiger Transfer der Daten vom GIS in die erweiterte DB	nicht notwendig
5. Modellierung	immer notwendig	in Abhängigkeit von den Anfragen nur sehr eingeschränkt notwendig
Gesamturteil	gute Lösung für einfache, wenig verknüpfte Daten, Anfragen über das ganze Gebiet und geringen Visualisierungsansprüchen	bessere Lösung für komplexe stark verknüpfte Daten, lokale Anfragen und hohe Visualisierungsansprüche

Tab. 31: Vergleich eines erweiterten DBMS mit GQL auf RDBMS

Bewertung

Die Fähigkeiten räumlicher DBMS-Erweiterungen GIS-Analyseaufgaben zu übernehmen wachsen. Doch nicht bei allen Aufgaben ist es auch sinnvoll, sie in der Datenbank bearbeiten zu lassen. Analysen können nach zwei Kriterien unterschieden werden: Erstens anhand der Anzahl der von der Anfrage betroffenen Elemente und zweitens anhand der Anforderungen an die Visualisierung der Ergebnisse (s. Tab. 32).

Ansprüche an die Visualisierung	Anteil betroffener Elemente	
	klein (lokale Anfragen)	groß (globale Anfragen)
komplexe Graphik	GIS	noch offen
einfache Graphik oder nur tabellarisch	GIS und ORDBMS	ORDBMS

Tab. 32: Eignung verschiedener Arten von Analysen für GIS oder ORDBMS

Lokale Analyseaufgaben mit komplexer Graphik und hohen Ansprüchen an die Visualisierung sind nach wie vor eine Domäne der GIS. Die lokalen Daten können ohne weiteres auf einmal geladen und im Hauptspeicher gehalten werden. Die dann erforderlichen Analyseaufgaben können im Hauptspeicher mit den vorhandenen Werkzeugen des GIS sehr schnell und effizient durchgeführt werden. Die Ergebnisse stehen sofort in der Graphik zur Verfügung und können zur Visualisierung weiter bearbeitet werden.

Werden im Gegensatz dazu einfache graphische Ergebnisse oder tabellarische Ausgaben gefordert, dafür aber Analysen über das ganze geographische Gebiet der Datenbank gefahren, so empfiehlt es sich, erweiterte DBMS einzusetzen. Die graphischen Ergebnisse können von einem GIS-Frontend dargestellt werden, das auf dem DBMS aufsetzt. Tabellarische Übersichten können mit den Standard-Werkzeugen des DBMS erstellt werden. Durch die in die Anfrageverarbeitung einbezogenen räumlichen Indizes behalten erweiterte DBMS ihre Leistung auch bei großen Datenmengen. Nur die Ergebnisse müssen aus der Datenbank in das GIS zur Anzeige übertragen werden, alle anderen Berechnungen laufen innerhalb des DBMS ab.

Kommen die Anforderungen nach einer qualitativ hochwertigen graphischen Präsentation der Ergebnisse mit globalen Anfragen zusammen, so haben sowohl GIS als auch erweiterte DBMS ihre Schwierigkeiten. GIS können die großen Datenmengen nur schwer nach gemischten Sach- und Geokriterien analytisch bearbeiten, für die DBMS gibt es noch keine geeigneten integrierten Visualisierungswerkzeuge. Eine Lösung dieses Problems kann darin liegen, ein auf RDBMS basierendes GIS auf ein erweitertes DBMS zu portieren und dessen Möglichkeiten so zu nutzen, daß sie im GIS zur Verfügung stehen.

10.Zusammenfassung und Ausblick

Zusammenfassung

Das Ziel der vorliegenden Arbeit ist, eine geographische Anfragesprache in einem GIS zur Verfügung zu stellen. Sie beschreitet dabei zwei unterschiedliche Wege: Zum einen definiert sie die Sprache GQL und implementiert sie in einem GIS, das auf einem relationalen DBMS basiert. Zum anderen verwendet sie die Sprache und die Modellierungsmöglichkeiten eines erweiterten ORDBMS und integriert dieses mit einem GIS-Frontend zu einer GIS-Applikation, die geographische Anfragen erlaubt und die Ergebnisse darstellen kann.

Sie evaluiert die Eignung beider Ansätze durch die Entwicklung eines Prototypen für jeden der beiden Wege. Sie untersucht die Prototypen sowohl anhand von systematischen Tests mit generierten Testdaten, als auch anhand von realen Testdaten und -anfragen, die aus dem laufenden Produktionsprozeß zweier beteiligter Projektpartner gewonnen wurden. Sie vergleicht die gewonnenen Ergebnisse und leitet Erfahrungen für die GIS-Entwicklung daraus ab.

Die Arbeit zeigt, daß man je nach der Anzahl der durchsuchten Elemente und den Ansprüchen an die Visualisierung unterschiedliche Lösungen bevorzugen sollte. Weiterhin kommt sie zu dem Schluß, daß eine *räumliche Erweiterung* einer objektrelationalen Datenbank wie Oracle Spatial noch keine *GIS-Erweiterung* ist. Dafür sind die angebotenen Datentypen und insbesondere die Funktionen auf einer zu niedrigen, rein auf die Geometrie bezogenen Ebene angesiedelt. Was fehlt, sind typische GIS-Funktionen, wie z. B. die Unterstützung anspruchsvoller Präsentation der Ergebnisse. Diese müssen weiterhin in einer Zwischenschicht zwischen GIS und Geodatenbank vom GIS-Hersteller oder Anwender selbst implementiert werden. Die Arbeit zeigt hier, wie basierend auf dem Standard OGC Simple Features, Präsentations-Daten in der Datenbank gehalten und mit Geo-Objekten zusammengefügt werden können.

Die Arbeit bestätigt die Vorteile objektrelationaler Datenbanken gegenüber rein relationalen Datenbanken als Basis für den Aufbau eines Geodatenservers für ein GIS, da die GIS-Implementierung auf einer höheren Ebene aufsetzen kann, als bei relationalen Datenbanken.

Ausblick und weitere Forschungsmöglichkeiten

Die Datenhaltung für GIS wird ihren Spezialistenstatus mit proprietären Lösungen in den nächsten Jahren verlieren. Die für die GIS-Datenhaltung benötigte Technologie wird Schritt für Schritt von den Herstellern von Standarddatenbanken in Ihre Systeme integriert werden, die verstärkt Analyseaufgaben mit übernehmen können (siehe Abb. 61).

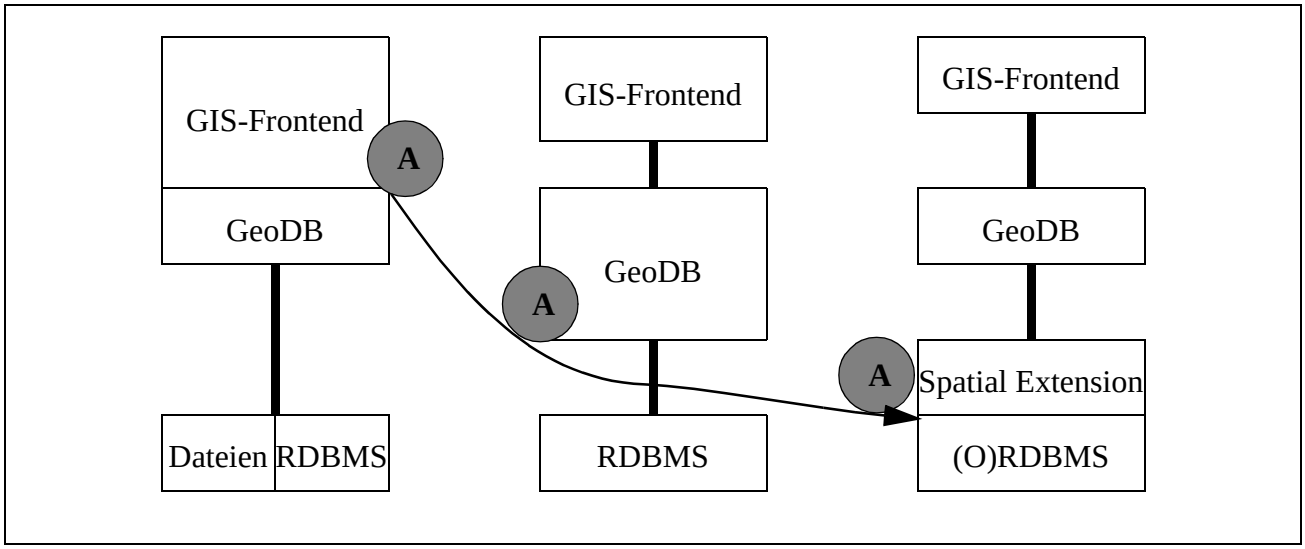


Abb. 61: Geodatenbanken übernehmen zunehmend GIS-Analyse-Aufgaben (A)

Auf dieser Erkenntnis aufbauend lassen sich die Ergebnisse dieser Arbeit auf drei Arten nutzen und weiterentwickeln:

1. GQL auf der Basis von ORDBMS erweitern
   Hier werden die Vorteile der GIS-basierten Anfragesprache GQL beibehalten und der Nachteil der schlechteren Performance dadurch ausgeglichen, daß ORDBMS zum Beantworten der Anfragen verwendet werden. Dazu sind keine Veränderungen im GIS-Frontend und keine Migration bestehender Daten notwendig. Nur die Middleware der GDB-X nutzt die Möglichkeiten der räumlichen Datenbankerweiterung, indem sie z. B. einen zusätzlichen Index für Anfragen aufbaut. Dieser Ansatz ist einfach, in seinen Auswirkungen im GIS begrenzt und somit leicht umzusetzen, aber er behält auch die bisherigen proprietären Datenstrukturen bei.

2. RDBMS-basierte GIS auf ORDBMS umstellen
   Entwickelt man die im zweiten Ansatz vorgestellten Konzepte zur Nutzung von ORDBMS für GIS weiter, so muß man sich darauf konzentrieren, wie alte Datenbestände neu modelliert und migriert werden können. Der Wechsel von RDBMS und BLOBs zu ORDBMS beschreitet bereits den Weg in Richtung offener Systeme. Die Präsentations-Informationen sind allerdings bis jetzt noch herstellerabhängig gespeichert. Hier besteht noch Bedarf an Forschung, Entwicklung und Standardisierung. Dieser zweite Punkt könnte Zwischenschritt zu einer neuen GIS-Architektur sein, wie sie im nächsten Punkt vorgestellt wird.

3. Entwurf einer neuen GIS-Architektur
   Fügt man die Ergebnisse dieser Arbeit und die aktuellen Bestrebungen nach offenen, komponentenbasierten Systemen, wie sie auch vom OGC unterstützt werden, zusam-

men, so ergibt sich als konsequenteste Umsetzung eine neue GIS-Architektur. Sie beinhaltet folgende Punkte:

Präsentations-Informationen

Das in der Arbeit vorgestellte Konzept zur Speicherung von Styles und ihrer Verknüpfung mit Geo-Objekten könnte verallgemeinert, weiterentwickelt und als Standardisierungsvorschlag bei OGC eingereicht werden.

Komplexe Geometrien

Die im Rahmen der Arbeit notwendige Migration von Geodaten ergab, daß einfache Geometrien für in Europa übliche GIS-Anwendungen nicht immer ausreichen. Hier besteht Bedarf, die Standardisierung der komplexen Geometrien weiter voranzutreiben.

Definition einer auf Diensten basierten Gesamtarchitektur

Die Ergebnisse dieser Arbeit können als Basis für eine Gesamtarchitektur von Geo-Diensten dienen, die Geodaten mit oder ohne Präsentationsinformation an verschiedene Anwendungen liefern (s. Abb. 62).

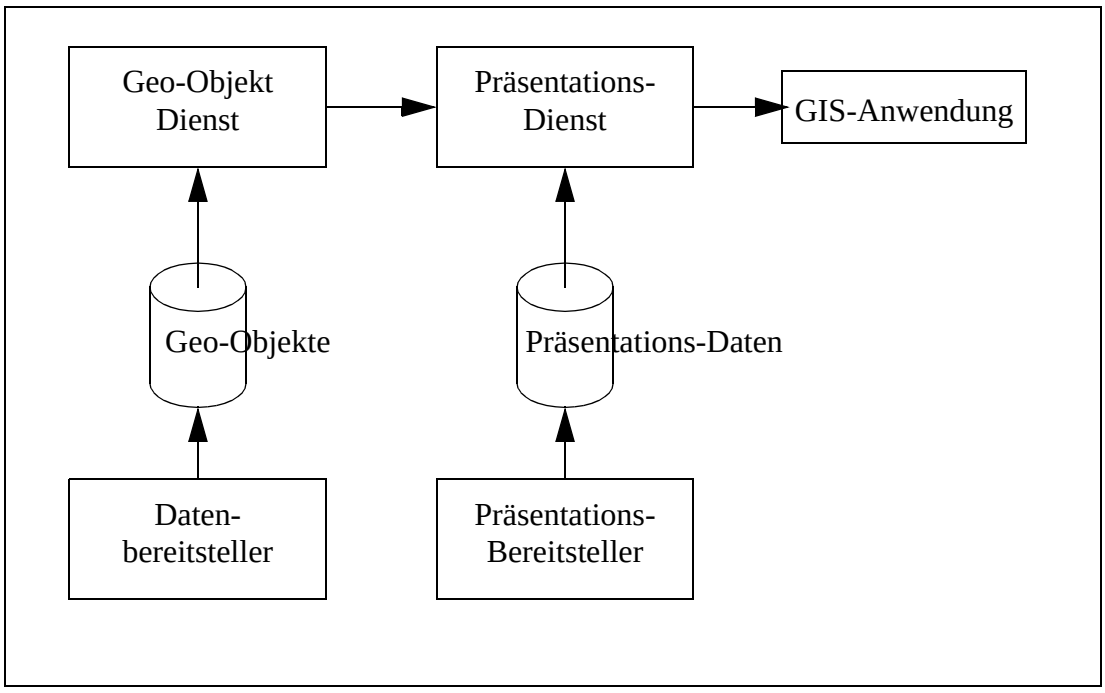


Abb. 62: Ausblick auf eine Gesamtarchitektur von GIS-Diensten

Ein Geo-Objekt-Dienst könnte leicht mit dem getesteten Oracle Spatial aufgebaut werden. Der Präsentations-Dienst bekommt Geo-Objekte vom Geo-Objekt-Dienst, fügt Darstellungsinformation hinzu und leitet beides an die GIS-Anwendung weiter.
Hinterfragt man diesen groben Architekturansatz, so ergeben sich eine Reihe interessanter Forschungsfragen, z. B. wie geeignete Interfaces zwischen den einzelnen Komponenten aussehen sollen oder wie diese Architektur um weitere Dienstleister oder Dienste z. B. für Metadaten ergänzt werden kann.

Diese Lösung verknüpft voneinander unabhängige Komponenten über offene, standardisierte Schnittstellen und wird sich auf lange Sicht durchsetzen.

Die Welt der Geoinformationssysteme wird sich durch die im ersten Kapitel genannten Trends verändern und die GIS-Hersteller vor neue Aufgaben stellen: Eine davon wird sein, GIS und Datenbanken unter Nutzung objektrelationaler Technologie besser als bisher zu integrieren und damit der Vision von GIS als einer in unternehmensweite Informationssysteme nahtlos eingebundenen Komponente einen Schritt näher zu kommen. Die Verwertung der Ergebnisse dieser Arbeit möge dazu beitragen.

11.Glossar

Das Glossar erklärt wichtige Begriffe basierend auf dem Glossar des GIS-Lehrstuhls der TU München [Schilcher et. al. 00].

Abfragesprache (query language)	Eine A. ist eine textbasierende Sprache, um alle Funktionen eines *DBMS* anzusprechen. Die A. ist eine Schnittstelle zwischen dem Applikationsprogrammierer oder Endbenutzer und dem DBMS. Die A. dient zur Datenmanipulation, Datendefinition und zur Administration einer *Datenbank*. Ein typischer Vertreter einer A. ist *SQL*.
AdV	ist die “Arbeitsgemeinschaft der Vermessungsverwaltungen der Länder der BRD”. Die AdV hat das *ALK/ATKIS* - Model und das Austauschformat *EDBS* konzipiert.
ALB	= Automatisches LiegenschaftsBuch. In ihm sind beschreibende Daten von Flurstücken (z.B. Gemarkung, Eigentümer, etc.) hinterlegt. Es hat urkundlichen Charakter und bildet zusammen mit der *ALK* den Kern des Liegenschaftskatasters.
ALK	“Automatisierte Liegenschaftskarte”. Digitale Grafikkomponente des amtlichen *Liegenschaftskatasters*. Das ALK ist als Modell der *AdV* in den meisten Bundesländern der BRD eingeführt. In Bayern übernimmt die *Digitale Flurkarte (DFK)* die Aufgaben der ALK. Wie das *ALB* hat die ALK urkundlichen Charakter.
ALKIS	“Amtliches Liegenschaftskataster-Informations-System”. Modell der *AdV* zur Integration von *ALB* und *ALK* in einheitliches Datenmodell, derzeit in Konzeption.
Alphadaten	Kurzform für *alphanumerische Daten*.
Alphanumerische Daten	Unter a.D. versteht man Datenobjekte, die ausschließlich durch numerische und alphabetische Datentypen repräsentiert sind. Datenfelder von a.D. können ganzzahlige Zahlen, Fließkommazahlen oder Texte sein. A.D. in einem *GIS* sind zum Beispiel beschreibende und administrative Daten von Betriebsmitteln (Hersteller, Alter. Wartungsdatum) oder von Liegenschaften (Eigentümer, Flurstücksnummer).
API	(Application Programming Interface) ist die Schnittstelle für Anwendungsprogramme, die von Programmentwicklern als vorprogrammierte Routinen genutzt werden.
API	Application Programming Interface
Applikation	Für einen spezifischen Anwendungszweck entworfene und implementierte Menge bzw. Folge von Operationen, die auf der generischen Funktionalität einer GIS-Software basiert. Meist unmittelbar mit einer Bedienungsoberfläche verbunden. Allgemeiner: Als EDV-Lösung konzipierte und realisierte Aufstellung, die damit für die Bedienung durch den Endbenutzer aufbereitet wird.
ATKIS	“Amtliches Topographisch Kartographisches Informationssystem”. Digitales geotopographisches Informationssystem der Deutschen Landesvermessung. Stellt amtliche *Geobasisinformationen* über die Erdoberfläche für private und öffentliche Anwender zur Verfügung. Das ATKIS-Konzept der *AdV* von 1989 sieht auf der Grundlage eines hierarchischen, objektorientierten Datenmodell *digitale, objektstrukturierte Landschaftsmodelle (DLM) und digitale kartographische Modelle (DKM)* vor. Ihr Aufbau und Inhalt sind in Objektartenkatalogen uns Signaturenkatalogen beschrieben. Für den Austausch von ATKIS-Vektordaten gilt grundsätzlich die *Einheitliche Datenbankschnittstelle (EDBS)*. Das ATKIS-Datenmodell wird derzeit durch die *AdV* überarbeitet und auf das in Konzeption befindliche *ALKIS*-Modell abgestimmt.
Attribut	Das Merkmal für eine geographische Eigenschaft beschrieben durch Nummern, Buchstaben Bilder oder CAD-Zeichnungen, Typischerweise in Tabellenform gespeichert und durch eine vom Benutzer zugewiesene Kennung an die Eigenschaft gebunden (siehe *Attributverknüpfung*) Spalte in einer Tabelle einer *Datenbank*.

Benchmark	Verfahren zum Vergleich von Computern und Anwendungsprogrammen zur Beurteilung des Leitungsumfanges, der Durchsatzrate etc.
Benutzeroberfläche (BOF)	Schnittstelle zur Bedienung eines Computers durch den Anwender; abhängig von den Möglichkeiten der Hardware und vom Komfort der Software; Bsp.: Einfache Kommandosprache oder ausgefeilte Menütechnik, vgl. GUI.
Betriebssystem	Eine Reihe von Computerprogrammen, die den Betrieb des Computers selber steuern. Anwenderprogramme, wie auch GIS-Software, laufen unter einem Betriebsystem. Beispiele für Betriebssysteme sind: Windows95/98, UNIX, DOS und OS/2, Windows NT.
Binary Large Object (BLOB)	Als BLOB werden Datenfelder bezeichnet, in denen große Mengen, für ein DBMS unstrukturierte, binäre Daten gespeichert werden können. Die Interpretation des Inhalts eines BLOB wird dabei vom Anwenderprogramm übernommen. Die Datentypen, die zur Definition solcher Datenfelder verwendet werden, sind von System zu System unterschiedlich.
BLOB	Binary Large Object Datentyp in RDBMS, er kann beliebig formatierte binär gespeicherte Daten aufnehmen, dem RDBMS ist der Sinn und Inhalt der Daten nicht bekannt, es speichert nur die Daten, nur die Anwendung kennt den Sinn der Daten und kann mit ihnen umgehen.
Datenbank	Elektronisches Archiv, Basis für anwendungsspezifische, integrierte Datenorganisation zur Vermeidung von Datenredundanzen. Grundlage zur Informationswiedergewinnung nach bestimmten Suchkriterien (Deskriptoren).
DBMS	(DataBase Management System, DatenBank-Management-System) Ein DBMS verwaltet eine zusammenhängende Menge von Daten - die Datenbank -, durch die den unterschiedlichen Informationsbedürfnissen eines Unternehmens Rechnung getragen werden sollen. Mit einem DBMS wird die Sicherheit, Integrität und Konsistenz der Daten bei minimaler, kontrollierter Redundanz gewährleistet. Durch die Integration der Daten und die Kontrolle des Zugriffs auf die Daten durch ein DBMS können die Daten durch viele Benutzer gleichzeitig genutzt werden. Dadurch kommt es zu weniger Zugriffskonflikten zwischen den einzelnen Bereichen eines Unternehmens. Für die Datendefinition und -manipulation steht eine einheitliche Schnittstelle, die Abfragesprache, zur Verfügung. Dadurch kann von verschiedenen Anwendungen nach dem gleichen Verfahren auf die Datenbank zugegriffen werden.
Deskriptoren	in CAD-Systemen mit grafischen Informationen verknüpfte zusätzliche alphanumerische Informationen; sie bilden zusätzliche Abfrage- und Auswerteparameter; in Datenbanken Suchwörter und -begriffe.
Diskretisierung	Räumliche Kontinua (Oberflächen, ‘rund’ verlaufende Linien müssen zur digitalen Handhabung je nach angestrebter Auflösung bzw. Maßstab diskretisiert - in kleine Abschnitte zerlegt - werden.
Embedded SQL	Unter E.SQL versteht man die Einbettung der Abfragesprache SQL in eine Wirtssprache. Die Wirtssprache (host language) ist dabei in der Regel eine Programmiersprache der 3. (C, Fortran Cobol) oder 4. Generation (4GL).
Fachdaten	Anwendungsspezifische Daten eines Fachanwenders. z.B. Leitungsdaten eines Versorgungsunternehmens oder Sachdaten. Zu den Fachdaten können auch Ergänzungen der amtlichen Geobasisdaten zählen, die durch den Anwender selbst erfaßt werden.
Filter	Selektion von Teil-Datenbeständen auf der Basis logischer und/oder arithmetischer Bedingungen.
Flächenverschneidung	(Area Intersection) ist ein Verfahren, bei dem aus zwei oder mehreren räumlich überlappenden Geometriemengen (Flächen, Linien oder Punkten) durch Zerlegung eine neue Menge gebildet wird. Aus dieser lassen sich wiederum Vereinigungs-, Schnitt- und Komplementärmengen ableiten.

Flurkarte	Auch *Katasterkarte*, stellt unter anderem alle Flurstücke mit ihren Nummern und Grenzen dar. In analoger Form auf Papier oder Karton meist im Maßstab 1:1000 geführt, in digitaler Form in verschiedenen Formaten erhältlich.
Fortführung	Aktualisierung von GIS-Daten. Vorgang zur Anpassung von Daten an laufende Veränderungen der abgebildeten Objekte. Fortführungen lassen sich auf das Neuentstehen, Wegfallen oder die Veränderung von Eigenschaften und Beschreibungen eines Objekts zurückführen. Fortführungen können periodisch, d.h. in festgelegten Zeitintervallen oder kontinuierlich, d.h. laufend erfolgen. Die Fortführung von Fremdsystemen (z.B. Fortführung von amtlichen Geobasisdaten in Fachinformationssystemen) kann über den Transfer von Komplettdaten oder Differenzdaten erfolgen und stellt besondere Anforderungen an Datenschnittstellen.
GDB-X	(auch SICAD-GDB-X) alte Bezeichnung des SICAD Geodaten-Servers von SNI; ein eigenständiges Programm für das Geodatenmanagement des GIS SICAD/open von SNI.
Genauigkeit	Grad der Reproduzierbarkeit bzw. Exaktheit von (räumlicher) Information (siehe *Auflösung*). Meist als Lagegenauigkeit verstanden, gibt an, wie groß die Abweichung der digital gespeicherten Lagekoordinaten eines Objektes von der Realität ist. Gemessen entweder als (statistische) mittlere Abweichung oder Maximalwert (Auflösung). Auch: numerische Genauigkeit der Speicherung von Werten (z.B. Koordinaten) als 16bit, 32bit usw. Daten.
Generalisierung	kartographischer Bearbeitungsschritt im Zuge einer Maßstabsverkleinerung, der zur Verbesserung der Kartenlesbarkeit und zum Entfernen von im Zielmaßstab überflüssigen bzw. Störenden Details notwendig ist. Umsetzung auf digitale Techniken ist bis heute noch kaum zufriedenstellend gelungen.
Geobasisdaten	Geodaten, die für viele GIS-Anwendungen benötigt werden und deren Basis bilden. Dies sind u.a. Bezugssysteme und Grundlagennetze, Höhendaten, Topographiedaten, Verwaltungsgrenzen auf nationaler, regionaler und lokaler (z.B. Flurstücks-) Ebene, Luftbilder. Unter Amtlichen Geobasisdaten wird der Datenbestand verstanden, der von den Vermessungsverwaltungen der Länder der BRD erfaßt und geführt wird und in den amtlichen Geoinformationssystemen ALK, ALB und ATKIS enthalten ist.
Geodaten	Unter G. oder raumbezogenen Daten versteht man Datenobjekte, die durch eine Position im Raum direkt oder indirekt referenzierbar sind. Der Raum ist dabei definiert durch ein Koordinatensystem, das den Bezug zur Erdoberfläche herstellt. G. Werden in der Regel grafisch in Papierform oder an graphikfähigen Bildschirmen präsentiert. Aus informationstechnischer Sicht kann man die Daten, die wiederum zu Geodaten gehören, einteilen in: Geometrie (Position und geometrische Ausprägung) Topologie (explizit gespeicherte geometrisch-topologische Beziehungen) Präsentation (graphische Ausprägungen wie Signaturen, Farbe, Typographie) Sachdaten (alphanumerische Daten zur Beschreibung der Semantik) Geodaten stellen in der Informationsverarbeitung eine besondere Herausforderung dar. Die Gründe dafür sind: der hohe Aufwand für die Erfassung die große Menge der anfallenden Daten die geforderten Antwortzeiten beim Zugriff auf G. die Verarbeitung nach räumlichen Kriterien sowie die Komplexität der Beziehungen.
Geodaten Server	Der Geodaten S. ist ein eigenständiges Programm, das Nachrichten von Client-Programmen empfangen und dann entsprechende Funktionen ausführen kann.

Geodaten-Management	(GDM) Mittlerweile wurden umfangreiche Geodatenbestände von verschiedenen Organisationen aufgebaut. Dazu gehören vor allem die Stadt- und Landesvermessungsämter, Umweltorganisationen, Versorgungsunternehmen, aber auch andere private Organisationen. In den Bereichen Verkehr, Transport, Straßenbau, Bauwesen, Land- und Forstwirtschaft, Energieversorgung und vielen anderen besteht ein großer Bedarf zur Nutzung von Geodaten. Die Vermarktung von Geodaten und die Kommunikation zwischen Datenanbietern und Datennutzern kann nur durch gezieltes GDM bewältigt werden. Bei GDM-Systemen steht vor allem die Übernahme, Pflege, Verwaltung und Bereitstellung von Geodaten und *Metadaten* im Vordergrund. Sie können sogar ohne graphische Ein- und Ausgabegeräte betrieben werden.
Geodaten-Server Datenbank	(Geodaten-Server Database) Bei einer Geodaten-Server D. handelt es sich um einen Satz von Tabellen in einem RDBMS, in denen geometrische, topologische, graphische und alphanumerische Daten gespeichert werden können. Beim Erstellen einer Geodaten-Server D. werden diese Tabellen angelegt und mit Anfangswerten belegt. Die Tabellen gehören dem Benutzer, der sie erstellt hat. Andere Benutzer können auf die Geodaten-Server D. zugreifen, wenn sie vom Eigentümer entsprechende Zugriffsrechte erhalten haben. Mit Ausnahme der alphanumerischen Daten kann auf die Tabellen einer Geodaten-Server D. nur das Geodaten-Server System sinnvoll zugreifen. Ein unkontrollierter Zugriff mit anderen Programmen würde die Datenintegrität der Geodaten-Server D. gefährden.
Geographic Query Language	(GQL) ist die kompatible Erweiterung der Standard-Abfragesprache SQL um räumliche Funktionen und Operatoren. Während mit SQL standardmäßig nur alphanumerische Daten verarbeitet werden können, bietet GQL die Möglichkeit, Geodaten mit ihren spezifischen Merkmalen abzufragen.
Geographische Datenbank	ist eine Datenbank zur Speicherung von Geodaten. Sie wird durch ein geographisches Datenbank-Managementsystem wie die Geodaten-Server verwaltet und besitzt alle Eigenschaften im Sinne der Datenbanktechnologie. Dazu gehört vor allem die strikte Trennung von Anwendung und Daten. Auf die Daten in einer Geodatenbank muß genauso unabhängig von einem spezifischen Verarbeitungssystem (hier: GIS) zugegriffen werden können, wie dies bei rein alphanumerischen Datenbanken der Fall ist.
Geographische Zugriffsrechte	Ein RDBMS besitzt standardmäßig die Möglichkeit, Zugriffsrechte auf Datenbankobjekte wie Tabellen und Datenbanksichten zu vergeben. Durch das Geodaten-Server System besteht zusätzlich die Möglichkeit, auch Zugriffsrechte nach räumlichen Kriterien zu vergeben. So kann zum Beispiel definiert werden, daß ein bestimmter Benutzer nur in einem räumlich abgegrenzten Gebiet, also innerhalb einer Fläche, Manipulationen durchführen darf, während er im restlichen Plangebiet nur lesen darf.
Geoinformation	Raumbezogene Information zur Beschreibung von Gegebenheiten eines Landes
Georeferenzierung	Die Festsetzung der Beziehung zwischen den ebenen Koordinaten einer Karte und den bekannten Koordinaten eines geodätischen Bezugssystems.
Georelationale Konzeption	Anwendung des relationalen Datenbankmodells auf ortsbezogene Daten.
GIS	Geographisches Informationssystem; System zur Erfassung, Speicherung, Prüfung, Manipulation, Integration, Analyse und Darstellung von Daten, die sich auf räumliche Objekte beziehen, Nach gängigem Verständnis besteht ein GIS aus einer räumlich adressierbaren Datenbank und geeigneter, darauf abgestimmter Anwendungssoftware. Spezielle Ausprägungen von GIS sind: KIS (Kommunales IS), LIS (Land-IS), NIS (Netz-IS), UIS (Umwelt-IS), RIS (Raum-IS). Welche Komponenten jeweils dazugehören, ist nicht eindeutig festgelegt, da sich viele Bereiche überlappen.

GUI	(Graphical User Interface/grafische Benutzeroberfläche). Methode zur Bedienung des Computers unter Verwendung von bildlichen Schaltflächen (Symbolen) und Befehlslisten, die mit der Maus gesteuert werden. Diese Methode wird als leichter erlernbar angesehen als die Befehlszeilen-Oberfläche, in der Befehle über die Tastatur eingegeben werden müssen. Bsp. für GUIs ist Microsoft Windows für PCs. Vgl. *Benützeroberfläche*
Interaktiv	Wechselspiel zwischen Benutzer und Computer, d.h. schrittweises Vorgehen. Der Computer antwortet auf jede vom Benutzer initiierte Eingabe; jederzeit unterbrechbar.
Interface	Schnittstelle, Verbindung Hardwareseitig: Definition physikalischer(Spannungspegel) und geometrischer (Stecker) Daten an einer Übergabestelle, z.B. Rechner und Bildschirm, Drucker, Plattenspeicher u.a. Softwareseitig: Definition von Programmschnittstellen.
Join	Verbindungsaufbau und physisches Verschmelzen zweier Attributtabellen mittels gemeinsamer Eigenschaften
Koordinaten	Zahlen, die den Standpunkt eines Punktes im Verhältnis zu einem Ursprungspunkt angeben. Kartesische Koordinaten drücken den Standort zwei- oder dreidimensional als senkrechte Entfernung von zwei oder drei orthogonalen Achsen aus.
Koordinatensysteme	polare, geographische und kartesische Systeme. Eindeutig kann die Erdoberfläche nur mittels geographischer Koordinaten abgebildet werden, lokal angepaßte (Projektions-) Systeme bieten jedoch den Vorteil rechtwinkliger Achsen mit einheitlich metrischer Teilung, so daß die euklidische Metrik angewandt werden kann. Die gebräuchlichsten K. sind Gauß-Krüger und *UTM* (Universelles Transversales Mercatorsystem).
Legende	dokumentiert die Zuordnung von Attributkategorien räumlicher Objekte zu graphischen Darstellungen, also etwa die Entsprechung unterschiedlicher Schraffurdichten in Flächen zu den Klassen des kartierten Flächenattributs.
Liegenschaftskataster	Das L. ist amtliches Verzeichnis der Grundstücke als Bestandteil des Grundbuchs. Es enthält grafische und beschreibende Informationen zu Grundstücken und Eigentumsverhältnissen, die u.a. in Liegenschaftsbuch und Liegenschaftskarte nachgewiesen sind. In digitaler Form sind dies *ALB* und *ALK* (in Bayern *DFK*)
Makro	eine Reihe von Programmbefehlen oder Anweisungen, die in einer Datei gespeichert werden und auf Wunsch wieder aufgerufen werden können. Makros werden zum individuellen Anpassen von einzelnen GIS Anwendungen verwendet. Eine Makroaufzeichnung über Handlungen ist möglich.
Menu	Auswahl von Optionen, die dem Benutzer graphisch und/oder textlich am Bildschirm dargeboten wird. Als Benutzerschnittstelle alternativ zu Kommandosprache.
ODBMS	(Objektorientierte DatenBank-ManagementSystem) Ein O. muß in der Lage sein, objektorientierte Programmiersysteme um Datenbankeigenschaften zu erweitern. Deshalb basiert eine der ersten Definitionen der Eigenschaften von O. auf der Aussage, daß O. mindestens das objektorientierte Paradigma (siehe objektorientierte Softwaretechnologie) und zusätzlich bestimmte Datenbankeigenschaften beinhalten müssen. Zu den zusätzlichen Eigenschaften gehören Dinge wie die Verwaltung komplexer Objekte, Versionierung oder geschachtelte und lange Transaktionen.
OGC	Open GIS Consortium, Internationaler Zusammenschluß von Herstellern und Anbietern von Geoinformationssystemen und Geodaten.
OGIS	Open GIS, “Offene” bzw. interoperable Geoinformationssysteme; die Schaffung interoperabler GIS ist Ziel des Open GIS Consortiums (OGC).
ORDBMS	Objekt relationales Datenbankmanagementsystem (*DBMS*);
PROBE	GIS der Computer Corporation of America, verbindet Vektor und Rasterdaten.
Quadtree	Datenstruktur, die eine beliebige Fläche in vier Quadranten unterteilt und die Quadranten solange in der gleichen Weise unterteilt, bis sie einheitlich sind oder bis die Grundauflösung der Daten erreicht ist. Wird am häufigsten zur Komprimierung von Rasterdaten angewendet.

Raster	Datenstruktur, die aus quadratischen Zellen besteht. Zellengruppen stellen geographische Merkmale dar; der Wert in der Zelle steht für ein Attribut des Merkmals.
Rasterdaten	(raster data) Mit R. bezeichnet man in Matrixform (Zeilen*Spalten) vorliegende digitale Daten. In der GIS-Welt sind das in der Regel Bilddaten (Rasterbilder) mit einem (geo-) graphischen Bezug. Die einzelnen Bildelemente heißen Pixel. Es kann sich jedoch auch um beliebige andere numerische Informationen handeln (z.B. Meßwerte), die einer Rasterfläche zugeordnet sind. In diesem Fall nennt man die einzelnen Elemente Rastermaschen.
Raumbezogener Zugriff	(spatial access) bezeichnet den Zugriff auf Geodaten über Koordinaten oder Adressen.
Räumliche Analyse	ist ein Prozeß, in dem analytische Techniken an Datensätzen mit geographisch bezogenen Informationen angewendet werden, um neue geographische Informationen zu extrahieren oder zu erstellen. Die räumliche Analyse kann zur Darstellung von komplexen geographischen Wechselwirkungen angewendet werden und dient auch zur Bestimmung der Eignung eines Standortes und der Prognose zukünftiger Ereignisse. Obwohl die gesamte analytische Technik sehr kompliziert erscheint, besteht sie im allgemeinen nur aus einer Kombination von einfachen Techniken, die in der entsprechenden Reihenfolge angewendet werden müssen.
Räumliches Bezugssystem	Summe von Definitionen (Koordinatensystem, Paßpunkte, Blattschnitte), welche die Systematik des Lagebezugs räumlicher Objekte in einem GIS ordnen.
Räumliches Objekt	(räumliche Einheit) spezielle Klasse von Objekten, die zusätzlich zu anderen Attributen immer auch solche mit Lageinformationen besitzen. Zu unterscheiden sind die Kategorien Punkt, Linie, Fläche und Raster bzw. deren Aggregate (z.B. Netzwerk).
RDBMS	Relationales Datenbankmanagementsystem Die relationale Datenbanktechnologie wurde durch erste Veröffentlichungen von E. F. Codd im Jahre 1970 begründet. Das relationale Modell beruht auf der Speicherung der Daten in einer zweidimensionalen Tabelle (relation). Eine Tabelle hat folgende Eigenschaften: Jede Spalte (column) enthält Werte des gleichen Datentyps. Es sind nur einfache Datentypen erlaubt. Jede Spalte hat einen eindeutigen Namen. Die Reihenfolge der Spalten ist irrelevant. Jeder Satz (row) einer Spalte ist unterschiedlich. D.h. ein Satz kann mit den gleichen Werten nur einmal in der Tabelle vorkommen. Die Reihenfolge der Sätze ist irrelevant. Basierend auf diesen Eigenschaften ist es möglich, einen formalen Satz mengentheoretischer Operation auf Tabellen zu definieren. RDBMS haben heute die größte Marktverbreitung. Zu den gängigsten Produkten gehören INFORMIX, INGRES, ORACLE, SYBASE und IBM. RDBMS bieten heute eine ausgereifte Technologie für große Datenmengen mit einer großen Anzahl gleichzeitiger Nutzer.
Sachdaten	(alphanumeric data) siehe Alphanumerische Daten
Sachsatz	Ein S. entspricht einem Eintrag in einer relationalen Tabelle und enthält alphanumerische Daten (keine Geodaten) einer realen Welt.
SEQUEL	Structured English Query Language Vorläufer von SQL.
SQL	(Structured Query Language) ist die Standardabfragesprache zur Benutzung von RDBMS. Der Sprachumfang beinhaltet sowohl DDL als auch DML Befehle. SQL ist standardisiert. Der derzeitig gültige Standard ist ISO 9075 SQL2 Entry Level von 1992. Durch den Einsatz von SQL wird ein hohes Maß an Übertragbarkeit von Anwendungen erreicht. Die Verfügbarkeit weiterer Schnittstellen wie ODBC (Open Data-Base Connectivity, basierend auf SQL), ermöglichen den leichten Zugriff auf Datenbanken von Microsoft-Werkzeugen aus.

Tool	(engl. Werkzeug); Sammelbegriff für kleinere Softwareprogramme für Nebenaufgaben (vgl. Utilities); auch für die Bezeichnung von Bearbeitungsoperationen bei der Grafikverarbeitung verwendet (hier meist als Befehle in Menüleisten abrufbar).
Topologie	T. beschreibt die nichtmetrischen räumlichen Verbindungen zwischen Objekten auf beliebig geformten Körpern.
Transaktion	besteht aus mehreren logisch zusammenhängenden Einzelaktionen. Die T. setzt sich technisch aus einer ununterbrechbaren Folge aus DML-Befehlen, welche die Datenbank von einem logisch konsistenten Zustand in einen neuen, logisch konsistenten Zusatz überführt. Dabei müssen alle Befehle oder keiner der Befehle wirksam werden. Eine erfolgreiche T. erhält die Datenintegrität (Einhaltung der Menge aller Integritätsbedingungen). Alle Aktionen einer T. müssen vor anderen, parallel laufenden T. verborgen bleiben. Sobald die Änderungen einer T. freigegeben werden, müssen sie permanent in der Datenbank verfügbar sein, egal, welche Fehlerfälle auch eintreten. Das Transaktionsparadigma befreit den Anwendungsprogrammierer von allen Aspekten des Fehlerfalls und der Nebenhäufigkeit (konkurrierender Zugriff im Mehrbenutzerbetrieb).
Vektordaten	Datenmodell, das auf der Darstellung von geographischen Objekten durch kartesische Koordinaten basiert und im allgemeinen zur Darstellung linearer Merkmale verwendet wird. Jedes Merkmal wird durch eine Reihe von Koordinaten dargestellt, die seine Form definieren und verknüpfte Informationen besitzen können. Hoch entwickelte Vektordatenmodelle schließen Topologie mit ein.
Verschneidung	(overlay); Gruppe grundlegender GIS-Funktionen, die ein digitales Zusammenführen von Lage- und Attributinformationen mehrerer Schichten ermöglicht. Durch Boole'sche Operatoren wird die Art der Zusammenführung genauer spezifiziert.

12.Literatur

Verwendete Abkürzungen im Literaturverzeichnis:

IJGIS	International Journal of Geographical Information Systems, Taylor & Francis
TODS	Transactions on Database Systems
ACM	Association for Computing Machinery
ACM CS	ACM Computing Surveys
ACM SIGMOD	Proceedings of the ACM SIGMOD International Conference on Management of Data
AutoCarto	Proceedings of the International Symposium on Computer-Assisted Cartography
BTW	Datenbanksysteme in Büro, Technik und Wissenschaft
CACM	Communications of the ACM, ACM Press
GI	Gesellschaft für Informatik
ICDE	IEEE International Conference on Data Engineering
IEEE	Institute of Electrical and Electronics Engineers
IEEE DEB	IEEE Bulletin of the Technical Committee on Data Engineering
IFB	Informatik Fachberichte, Springer Verlag
IJGIS	International Journal of Geographical Information Systems
ISO	International Standards Organisation
LNCS	Lecture Notes in Computer Science, Springer Verlag
NaKaVerm	Nachrichten aus Kartographie und Vermessungswesen
SDH	International Symposium on Spatial Data Handling
SIGMOD	ACM Special Interest Group on Management of Data
SIGMOD Record	A Quarterly publication of the ACM Special Interest Group on Management of Data
SSD	Advances in Spatial Databases, Proceedings of the International Symposium on Large Spatial Databases
TKDE	IEEE Transactions on Knowledge and Data Engineering
TODS	ACM Transactions on Database Systems
TOIS	ACM Transactions on Information Systems
TOSE	IEEE Transactions on Software Engineering
URISA	Urban and Regional Information Systems Association
VLDB	Proceedings of the International Conference on Very Large Databases
ZfV	Zeitschrift für Vermessungswesen

Referenzen:

Abel 88 Abel, D. J., *Relational Data Management Facilities for Spatial Information Systems*, 3rd SDH, Sydney, pp. 9-18, 1988.

Abel 89 Abel, D. J., *SIRO-DBMS: A Database Tool Kit for Geographical Information Systems*, IJGIS, 3(2), pp. 103-116, 1989.

Abel 96 Abel, D. J., *What's special about spatial?*, 7th Australasian Database Conference, Melbourne, 18(2), pp. 72-81, 1996.

Abel & Smith 83 Abel, D. J., Smith, J. L., *A Data Structure and Algorithm Based on a Linear Key for a Rectangle Retrieval Problem*, Int. Journal of Computer Vision, Graphics and Image Processing, 24(1), pp. 1-13, 1983.

Abel & Smith 86 Abel, D. J., Smith, J. L., *A Relational GIS Database Accommodating Independent Partitionings of the Region*, 2nd SDH, Seattle, pp. 213-224, 1986.

Abel & Smith 87 Abel, D. J., Smith, J. L., *A Kernel-Shell Approach to an Extended Relational Spatial Database Management System*, Internal Report, CSIRO, Canberra, 1987.

Abiteboul & Beeri 88 Abiteboul, S., Beeri, C., *On the Power of Languages for the Manipulation of Complex Objects*, Technical Report 846, Paris: INRIA, 1988.

Abler 87 Abler, R., *The National Science Foundation National Center for Geographic Information and Analysis*, IJGIS, 1(4), pp. 303-326, 1987.

ALKIS 99 AdV, *ALKIS-Konzeptpapiere*, Stand vom 1.7.1999, AdV, 1999.

Amann & Scholl 92 Amann, B., Scholl, M., *GRAM: a Graph Data Model and Query Language*, Proc. of ECHT '92, Milano, Italy, pp. 201-211, 1992.

Anselin 89 Anselin, L., *What is special about spatial data? Alternative perspectives on spatial data analysis*, Technical Report 89-4, NCGIA, Santa Barbara, CA,1989.

Aref & Samet 90 Aref, W., Samet, H., *An Approach to Information Management in Geographical Applications*, 4th SDH, Zürich, pp. 589-598, 1990.

Aref & Samet 91a Aref, W., Samet, H., *Extending a DBMS with Spatial Operations*, 2nd SSD, Zürich, LNCS 525, pp. 299-318, 1991.

AS 91b Aref, W., Samet, H., *Optimization Strategies for Spatial Query Processing*, 17th VLDB, Barcelona, 1991.

Arnborg 80 Arnborg, S., *SAL - A Simple Query Language Based on Set Algebra*, BIT, 20, pp. 266-278, 1980.

Autometric 98 Autometric, *Autometric Corporation Homepage*, http://www.autometric.com/, 1998.

Bär et. al. 94 Bähr, U., Singer, C., Kiessling, W., *Zur Systematik räumlicher Operatoren in Geo-Datenbanken*, GIS, 7(4), Wichmann, Karlsruhe, pp. 13-21, 1994.

Balownew & Breunig 97 Balownew, O., Breunig, M., *Erste Versuche zur Modellierung der Zeit als Anwendung eines GeoToolKits*, Zeit als weitere Dimension in Geo-Informationssystemen, Workshop an der Universität Rostock, pp. 107-114, 1997.

Balownew et. al. 97 Balovnev, O., Breunig, M., Cremers, A. B., *From GeoStore to GeoToolKit: The Second Step*, 5th SSD, Berlin, LNCS 1262, pp. 223-237, 1997.

Barrera & Buchmann 81 Barrera, R., Buchmann, A., *Schema Definition and Query Language for a Geographical Database System*, Proceedings IEEE Conference on Computer Architecture for Pattern Analysis and Image Database Management, 11, pp. 250-256, 1981.

Batory et. al. 88 Batory, D. S., Barnett, J. R., Garza, J. F., Smith, K. P., Tsukuda, K., Twichell, B. C., Wise, T. E., *GENESIS: An Extensible Database Management System System*, IEEE TOSE, 14(11), pp. 1711-1730, 1988.

Bayer & McCreight 72 Bayer, R., McCreight, C., *Organization and Maintenance of Large Ordered Indexes*, Acta Informatica, 1(3), pp. 173-189, 1972.

Beckmann et. al. 90 Beckmann, N., Kriegel, H. P., Schneider, R., Seeger, B., *The R*-tree: An Efficient and Robust Access Method for Points and Rectangles*, ACM SIGMOD, Atlantic City, NJ, pp. 322-331, 1990.

Berman & Stonebraker 77 Berman, R. R., Stonebraker, M., *GEO-QUEL: A System for the Manipulation and Display of Geographic Data*, ACM Computer Graphics, 11(2), pp. 186-191, 1977.

Bill & Fritsch 94 Bill, R., Fritsch, D., *Grundlagen der Geo-Informationssysteme, Band 1: Hardware, Software und Daten*, 2. Auflage, Wichmann Verlag, Karlsruhe, 1994.

Boursier & Mainguenaud 92 Boursier, P., Mainguenaud, M., *Spatial Query Languages: Extended SQL vs. Visual Languages vs. Hypermaps*, 5th SDH, Charleston, SC, 1992.

Breunig 96 Breunig, M., *Integration of Spatial Information for Geo-Information Systems*, Lecture Notes in Earth Sciences, 61, Springer, 1996.

Brinkhoff 94 Brinkhoff, T., *Der Spatial Join in Geoinformationssystemen*, Dissertation Uni München, Shaker-Verlag, 1994.

Bringkhoff et. al. 94a Brinkhoff, T., Kriegel, H. P., Schneider, R., Seeger, R., *Multi-Step Processing of Spatial Joins*, ACM SIGMOD, Minneapolis, MN, pp. 197-208, 1994.

Calcinelli & Mainguenaud 91 Calcinelli, D., Mainguenaud, M., *The Management of the Ambiguities in a Graphical Query Language for Geographical Information Systems*, 2nd SSD, Zürich, LNCS 525, pp. 99-118, 1991.

Carey et. al. 86a Carey, M., DeWitt, D., Richardson, J., Shetika, E., *Object and File Management in the EXODUS Extensible Database System*, 12th VLDB, pp. 91-100, 1986.

Carey et. al. 86b Carey, J. M., DeWitt, D. J., Frank, D., Graefe, G., Muralikrishna, M., Richardson, J. E., Shekita, E. J., *The Architecture of the EXODUS Extensible DBMS*, Proc. of the IEEE, ACM International Workshop on Object-Oriented Database Systems, Pacific Grove, CA, pp. 52-65, 1986.

Carey et. al. 88 Carey, M., DeWitt, D., Vandenberg, S., *A Data Model and Query Langauge for EXODUS*, ACM SIGMOD, Chicago, IL, 17(3), pp. 413-423, 1988.

Carey et. al. 88 Carey, M., DeWitt, D., Graefe, G., Haight, D., Richardson, J., Schuh, D., Shekita, E., Vandenberg, S., *The EXODUS Extensible DBMS Project: An Overview*, Technical Report 808, University of Wisconsin, Madison, WI, 1988.

Chang & Fu 80 Chang, N. S., Fu, K. S., *Query-by-Pictorial-Example*, IEEE TOSE, SE-6(6), pp. 519-524, 1980.

Chang & Fu 81 Chang, N. S., Fu, K. S., *Picture Query Languages for Pictorial Data-Base Systems*, Computer, 14(11), pp. 23-33, 1981.

Chang & Schek 89 Chang, W. W., Schek, H.-J., *A Signature Access Method for the Starburst Database System*, 15th VLDB, Amsterdam, 1989.

Chamberlin et. al. 76 Chamberlin, D., Astrahan, M., Eswaran, K., Griffiths, P., Lorie, R., Mehl, J., Reisner, P., Wade, B., *Sequel 2: A unified approach to data definition, manipulation, and control*, IBM Journal of Research and Development, 20(6), pp. 560-575, 1976.

Charlwood et. al. 87 Charlwood, G., Moon, G., Tulip, J., *Developing a DBMS for Geographic Informations: a Review*, 8th Auto-Carto, Baltimore, MD, pp. 302-315, 1987.

Clementini & Di Felice 96 Clementini, E., Di Felice, P., *A Model for Representing Topological Relationships Between Complex Geometric Features in Spatial Databases*, Information Sciences, 90, pp. 121-136, 1996.

Clementini & Di Felice 97 Clementini, E., Di Felice, P., *Approximate Topological Relations*, Int. J. Approx. Reason. (USA), 16(2), pp. 173-204, Elsevier, 1997.

Clementini et. al. 95 Clementini, E., Di Felice, P., Califano, G., *Composite Regions in Topological Queries*, Information Systems, 20(6), pp. 33-48, 1995.

Codd 72 Codd, E. F., *Relational Completeness of Data Base Sublanguages*, Courant Computer Science Symposia, 6, Data Base Systems, Prentice Hall, pp. 67-101, 1972.

Costagliola et. al. 95 Costagliola, G., Tortora, G., Tucci, M., Busillo, M., *GISQL - A Query Language Interpreter for Geographical Information Systems*, Visual Database Systems 3, Chapmann & Hall, 1995.

Cruz et. al. 87 Cruz, I. F., Mendelzon, A. O., Wood, P. T., *A Graphical Query Language Supporting Recursion*, ACM SIGMOD, 1987.

David et. al. 93 David, B., Raynal, L., Schorter, G., V. Mansart, *GeO2: Why Objects in a geographical DBMS?*, 3rd SSD, Singapore, LNCS 692, pp. 264-276, 1993.

Dayal & Smith 86 Dayal, U., Smith, J. M., *PROBE: A Knowledge-Oriented Database Management System* , In: Brodie, M. L., Mylopoulos, J. (editors), On Knowledge Base management Systems: Integrating Artificial Inelligence and Database Technologies, Springer, pp. 227-257, 1986.

Dayal et. al. 87 Dayal, U., Manola, F., Buchman, A., Chakravarthy, U., Goldhirsch, D., Heiler, S., Orenstein, J., Rosenthal, A., *Symplifying Complex Objects: The PROBE Approach to Modelling and Querying them*, In: Schek, H. J., Schlageter, G. (editors), Proc. of BTW 87, Springer, pp. 17-37, 1987.

Egenhofer 87 Egenhofer, M. J., *Towards an Extended SQL for Treating Spatial Objects*, Proc. 2nd Int. Seminar on Trends and Concerns of Spatial Sciences, pp. 83-95, Fredericton, New Brunswick, Canada, 1987.

Egenhofer 89 Egenhofer, M. J., *Spatial SQL: A Spatial Query Language*, Report 103, Departement of Surveying Engineering, University of Maine, 1989.

Egenhofer 90 Egenhofer, M. J., *Interaction with Geographic Information Systems via Spatial Queries*, Journal of Visual Languages and Computing, 1(4), pp. 389-413, 1990.

Egenhofer 91a Egenhofer, M. J., *Categorizing Topological Relationships - Quantitative Refinements of Qualitative Spatial Information*, Technical Report, Departement of Surveying Engineering, University of Maine, Orono, ME, 1991.

Egenhofer 91b Egenhofer, M. J., *Extending SQL for Cartographic Display*, Cartography and Geographic Informations Systems, 18(4), pp. 230-245, 1991.

Egenhofer 91c Egenhofer, M. J., *Reasoning about Binary Topological Relations*, 2nd SSD, Zürich, LNCS 525, pp. 143-160, 1991.

Egenhofer 91d Egenhofer, M. J., *Deficiencies of SQL as a GIS Query Language*, Cognitive and Linguistic Aspects of Geographic Space, Kluwer Academic Publishers, pp. 477-491, 1991.

Egenhofer 92 Egenhofer, M., *Why not SQL!*, IJGIS, 6(2), pp. 71-85, 1992.

Egenhofer 93 Egenhofer, M., *Definition of Line-Line Relations for Geographic Databases*, IEEE DEB, 16(3), pp. 40-45, 1993.

Egenhofer 94 Egenhofer, M. J., *Spatial SQL: A Query and Presentation Language*, IEEE TKDE, 6(1), pp. 86-95, 1994.

Egenhofer 97 Egenhofer, M. J., *Spatial Relations: Models and Inferences*, 5th SSD, Berlin, Tutorial Workbook, 1997.

Egenhofer et. al. 94 Egenhofer, M. J., Clementini, E., Di Felice, P., *Topological Relations between Regions with Holes*, IJGIS, 8(2), pp. 129-142, 1994.

Egenhofer & Frank 87 Egenhofer, M., Frank, A., *Object-Oriented Databases: Database Requirements for GIS*, Proc. Int. GIS Symposium: The Research Agenda, Vol. II, US Government Printing Office, Washington D.C., 1987.

Egenhofer & Frank 89a Egenhofer, M., Frank, A., *Object-Oriented Modeling in GIS: Inheritance and Propagation*, 9th Auto-Carto, Baltimore, MD, pp. 588-589, 1989.

Egenhofer & Frank 89b Egenhofer, M., Frank, A., *Why Object-Oriented Software Engineering Techniques are Necessary for GIS*, International Geographic Information Systems (IGIS) Symposium, Baltimore, MD, 1989.

Egenhofer & Frank 91 Egenhofer, M., Franzosa, R., *Point-Set Topological Spatial Relations*, IJGIS, 5(2), pp. 161-174, 1991.

Egenhofer & Herring 90 Egenhofer, M., Herring, J., *A Mathematical Framework for the Definition of Topological Relationships*, 4th SDH, Zürich, pp. 803-813, 1990.

Egenhofer & Herring 91 Egenhofer, M., Herring, J., *Categorizing Binary Topological Relationships Between Regions, Lines and Points in Geographic Databases*, In: A Framework for the Definition of Topological Relationships and an Algebraic Approach to Spatial Reasoning within this Framework, NCGIA Technical Report 91-7, Departement of Surveying Engineering, University of Maine, Orono, ME, 1991.

Egenhofer & Mark 95 Egenhofer, M. J., Mark, D. M., *Modelling Conceptual Neighbourhoods of Topological Line-Region Relations*, IJGIS, 9(5), pp. 555-565, 1995.

Erwig & Schneider 97 Erwig, M., Schneider, M., *Vague Regions*, 5th SSD, Berlin, LNCS 1262, pp. 298-320, 1997.

ESRI 98 ESRI, *Environmental Systems Research Institute Homepage*, http://www.esri.com/, 1998.

Findeisen 90 Findeisen, D., *Datenstruktur und Abfragesprachen für raumbezogene Informationen*, Dissertation Uni Bonn, Schriftenreihe des Instituts für Kartographie, 19, Bonn, Kirschbaum, 1990.

Finkel & Bentley 74 Finkel, R. A., Bentley, J. L., *Quad Trees: A Data Structure for Retrieval on Composite Keys*, Acta Informatica, 4, pp. 1-9, 1974.

Frank 82a Frank, A., *MAPQUERY: Data Base Query Language for Retrieval of Geometric Data and its Graphical Representation*, In: Bergeron, D. (editor), SIGGRAPH'82, ACM Computer Graphics, 16(3), pp. 199-207, 1982.

Frank 82b Frank, A., *MAPQUERY: Design and Implementation*, Problems of Realization of Land Iinformation Systems, Part 2, Report 56, Institut für Geodäsie und Photogrammetrie, Swiss Federal Institute of Technology , Zürich, Switzerland, 1982.

Frank 82c Frank, A., *Data Structure and Data Description for Geo Data Systems*, Problems of Realization of Land Iinformation Systems, Part 3, Report 56, Institut für Geodäsie und Photogrammetrie, Swiss Federal Institute of Technology , Zürich, Switzerland, 1982.

Frank 84a Frank, A., *Requirements for Database Systems Suitable to Manage Large Spatial Databases*, 1st SDH, Zürich, 1984.

Franzosa & Egenhofer 92 Franzosa, R., Egenhofer, M., *Topological Spatial Relations Based on Components and Dimensions of Set Intersections*, Melter, R., Wu, A. (editors), International Society for Optical Engineering, SPIE's OE/Technology '92 - Vision Geometry, vol. 1832, pp. 236-246, 1992.

Gardarin et. al. 87 Gardarin, G., Jean-Noel, M., Kerherve, B., Pasquer, F., Pastre, D., Simon, E., Valduriez, P., Verlaine, L., Viemont, Y., *Sabrina * : a Relational Database System Developed in a Research Environment*, TSI, 6(3), 1987.

Gaede 96 Gaede, V., *Extending Query Optimization for Spatial Database Systems*, Dissertation, Humboldt-Universität, Berlin, 1996.

Gaede & Günther 95 Gaede, V., Günther, O., *Survey on Multidimensional Access Methods*, Technical Report, ISS-16, Institut für Wirtschaftsinformatik, Humboldt Universität, Berlin, 1995.

Gaede & Riekert 94 Gaede, V., Riekert, W.-F., *Spatial Access Methods and Query Processing in the in the Object-Oriented GIS GODOT*, Proc. of the AGDM'94 Workshop, Netherlands Geodetic Commission, 1994.

Goh 89 Goh, P.-C., *A Graphic Query Language for Cartographic and Land Information Systems*, IJGIS, 3(3), pp. 245-255, 1989.

Goodchild 89 Goodchild, M. F., *Tiling Large Geographical Databases*, 1st SSD, Santa Barbara, LNCS 409, pp. 137-146, 1989.

Goodchild 90 Goodchild, M. F., *Spatial Information Science*, Keynote Address, 4th SDH, Zürich, pp. 3-12, 1990.

Gould et. al. 96 Gould, M., Brand, M., Craglia, M., Fotheringham, S., Frank, A., *What‘s so special about spatial?*, GIS Europe, 10, pp. 22-26, 1996.

Gradwell 90a Gradwell, D., *Analysis of Requirements for Database Software for Constructing a GIS*, The Association of Geographic Information, Proc. of AGI, Brighton, UK, 1990.

Gradwell 90b Gradwell, D., *Can SQL Handle Geographic Data?*, AGI Standards Comitee - SQL Working Party, Proc. of AGI, Brighton, UK, 1990.

Günther 98 Graefe, G., DeWitt, D. J., *Environmental Information Systems*, Springer, 1998.

Günther 87 Günther, O., *The EXODUS Optimizer Generator*, ACM SIGMOD, San Francisco, CA, pp. 160-172, 1987.

Güting 88a Güting, R. H., *Modeling Non-Standard Database Systems by Many-Sorted Algebras*, Uni Dortmund, Fachbereich Informatik, Report 255, 1988.

Güting 88b Güting, R. H., *Geo-Relational Algebra: A Model and Query Language for Geometric Database Systems*, In: Noltemeier, H. (editor), Computational Geometry and its Applications - Int. Workshop on Computational Geometry CG'88, Würzburg, Springer, LNCS 333, 1988.

Güting 88c Güting, R. H., *Geo-Relational Algebra: A Model and Query Language for Geometric Database Systems*, In: Schmidt, J. W., Ceri, S., Missikoff, M. (editors), Proceedings of the International Conference on Extending Database Technology, Venice, LNCS 303, pp. 506-527, 1988.

Güting 89 Güting, R. H., *Gral: An Extensible Relational Database System for Geometric Applications*, 15th VLDB, Amsterdam, pp. 33-44, 1989.

Güting 94a Güting, R. H., *GraphDB: Modeling and Querying Graphs in Databases*, 20th VLDB, Santiago, Chile, pp. 297-308, 1994.

Güting 94b Güting, R. H., *GraphDB: A Data Model and Query Language for Graphs in Databases*, FernUni Hagen, Informatik-Report, 155, 1994.

Güting 94c Güting, R. H., *An Introduction to Spatial Database Systems*, VLDB Journal, 3(4), pp. 357-399, 1994.

Güting & Schneider 93a Güting, R. H., Schneider, M., *Realm-Based Spatial Data Types: The ROSE Algebra*, FernUni Hagen, Report 141, 1993.

Güting & Schneider 93b Güting, R. H., Schneider, M., *Realms: A Foundation for Spatial Data Types in Database Systems*, 3rd SSD, Singapore, LNCS 692, pp. 14-35, 1993.

Güting & Schneider 95 Güting, R. H., Schneider, M., *Realm-Based Spatial Data Types: The ROSE Algebra*, VLDB Journal, 4(2), pp. 100-143, 1995.

Güting et. al. 95 Güting, R. H., Ridder, T. de, Schneider, M., *Implementation of the ROSE Algebra: Efficient Algorithms for Realm-Based Spatial Data Types*, 4th SSD, Portland, LNCS 951, pp. 216-239, 1995.

Guttman 84 Guttman, A., *R-Trees: A Dynamic Index Structure for Spatial Searching*, ACM SIGMOD, Boston, MA, 14(2), pp. 47-57, 1984.

Haas et. al. 89 Haas, L. M., Freytag, J. C., Lohman, G. M., Pirahesh, H., *Extensible Query Processing in Starburst*, ACM SIGMOD, Portland, OR, pp. 377-388, 1989.

Haas et. al. 90 Haas, L. M., Chang, W., Lohman, G. M., McPherson, J., Wilms, P. F., Lapis, G., Lindsay, B., Pirahesh, H., Carey, M. J., Shekita, E., *Starburst Midflight: As the Dust Clears*, IEEE TKDE, 2(1), pp. 143-160, 1990.

Hasan & Pirahesh 88 Hasan, W., Pirahesh, H., *Query Rewrite Optimization in Starburst*, IBM Almaden Research Center, Report RJ 6367, 1988.

Härder & Reuter 85 Härder, T., Reuter, A., *Architektur von Datenbanksystemen für Non-Standard-Anwendungen*, Blaser, A., Pistor, P. (editors), Datenbank-Systeme für Büro, Technik und Wissenschaft (BTW) - GI-Fachtagung, Karlsruhe, Informatik-Fachberichte, pp. 253-286, Springer, LNCS 94, 1985.

Herring et. al. 88 Herring, J. R., Larsen, R. C., Shivakumar, J., *Extensions to the SQL Language to Support Spatial Analysis in a Topological Data Base*, Proc. GIS/LIS'88 2, San Antonio, TX, pp. 741-750, 1988.

Hoop & Oosterom 92 Hoop, S. de, Oosterom, P. van, *Storage and Manipulation of Topology in Postgres*, In: Harts, J., Ottens, O., Scholten, H. (editors), 3rd European Conf. on Geographical Information Systems, EGIS'92, München, Verlag EGIS Foundation, Faculty of Geographical Sciences, Utrecht, NL, 1992.

Hoop et. al. 93 Hoop, S. de, Oosterom, P. van, Molenaar, M., *Topological Querying of Multiple Map Layers*, COSIT 93, LNCS 716, pp. 139-157, 1993.

Huang 93 Huang, Z., *Design of GeoSAL, a Database Language for Spatial Data Analysis*, PhD Thesis, Royal Institute of Technology, Environmental and Natural Resources Information Systems, Stockholm, Sweden, 1993.

Huang & Svensson 93 Huang, Z., Svensson, P., *Neighborhood Query and Analysis with GeoSAL, a Spatial Database Language*, 3rd SSD, Singapore, LNCS 692, pp. 413-436, 1993.

Huang et. al. 92 Huang, Z., Svensson, P., Hauska, H., *Solving Spatial Analysis Problem with GeoSAL, A Spatial Query Language*, Proc. of the 6th Int. Working Conf. on Scientific and Statistical Database Management, Schweiz, 1992.

IBM 98 IBM, *IBM Corporation Homepage*, http://www.ibm.com/, 1998.

Informix 98 Informix, *Informix Corporation Homepage*, http://www.informix.com/, 1998.

Ingram & Phillips 87 Ingram, K. J., Phillips, W. W., *Geographic Information Processing Using a SQL-Based Query Language*, 8th Auto-Carto, Baltimore, MD, pp. 326-335, 1987.

Jarke & Vassiliou 85 Jarke, M., Vassiliou, Y., *A Framework for Choosing a Database Query Language*, In: Mylopoulos, J., Brodie, M. L. (editors), Readings in Artificial Intelligence & Databases, Morgan Kaufmann, San Mateo, California, pp. 363-376, 1985.

Joseph & Cardenas 88 Joseph, T., Cardenas, A. F., *PICQUERY: a High Level Query Language for Pictorial Database Management*, IEEE TOSE, 14(5), pp. 630-638, 1988.

JTC 1 SC 32 ISO/IEC JTC 1 SC 32 WG 4, http://www.iso.ch/meme/JTC1SC32.html, 1998.

Karimi 96 Karimi, H., *Open Computing GIS - An Effective, Affordable Approach to Solving Spatial Problems*, GIS World, 9(3), pp. 48-51, 1996.

Kasturi et. al. 89 Kasturi, R., Fernandez, R., Amlani, M. L., Feng, W., *Map Data Processing in Geographic Information Systems*, IEEE Computer, 22, pp. 10-20, 1989.

Kemp 90 Kemp, Z., *An Object-Oriented Model for Spatial Data*, 4th SDH, Zürich, 1990.

Koch 97 Koch, H., *Anforderungen an GQL zum Einsatz mit ATKIS-Objekten*, persönliches Schreiben, unveröffentlicht, 1997.

Kottman 98 Kottman, C., *Geoprocessing Standards Connections: OGC and TC 211, TC 204, SQL MM, CORBAgis, W3C*, OpenGIS Newsletter, 3(2), pp. 11-11, OGC, Wayland, MA, USA, 1998.

Kramer 97 Kramer, H., *Modellierung, Transfer und Visualisierung von Geodaten in objekt-relationalen Datenbanksystemen am Beispiel von SICAD/GDB-X und Informix' Universal Server*, unveröffentlichte Diplomarbeit, FHO Emden, SNI, 1997.

Larue et. al. 93 Larue, T., Pastre, D., Viemont, Y., *Strong Integration of Spatial Domains and Operators in a Relational Database System*, 3rd SSD, Singapore, LNCS 692, pp. 53-72, 1993.

Laurini 87 Laurini, R., *Manipulation of Spatial Objects with a Peano Tuple Algebra*, University of Maryland, Center for Automation Research, Technical Report, CAR TR 311, 1987.

Laurini & Milleret 87 Laurini, R., Milleret, F., *Peano Relations in CADCAM Databases*, Int. Conf. IEEE Data and Knowledge Systems for Manufacturing and Engineering, Hartford, Connecticut, 1987.

Laurini & Milleret 89 Laurini, R., Milleret, F., *Spatial Data Base Queries: Relational Algebra versus Computational Geometry*, In: Rafanelli, M., Klensin, J., Svensson, P. (editors), LNCS 339, 1989.

Lee & Chin 95 Lee, Y. C., Chin, F. L., *An Iconic Query Language for Topological Relationships in GIS*, IJGIS, 9(1), pp. 25-46, 1995.

Lorie 91 Lorie, R., *The Use of a Complex Object Language in Geographic Data Management*, 2nd SSD, Zürich, LNCS 525, pp. 319-337, 1991.

Mainguenaud & Portier 90 Mainguenaud, M., Portier, M., *Cigales: A Graphical Query Language for Geographical Information Systems*, 4th SDH, Zürich, 1990.

Manola & Orenstein 86 Manola, F., Orenstein, J., *Toward a General Spatial Data Model for an Object-Oriented DBMS*, 12th VLDB, Kyoto, pp. 328-335, 1986.

Manola et. al. 87 Manola, F., Orenstein, J., Dayal, U., *Geographic Information Processing in the PROBE Database System*, 8th International Symposium on Computer-Assisted Cartography, Baltimore, Maryland, pp. 316-325, 1987.

McDonell 86 McDonell, K. J., *An Overview of the Relational Test Bed (RTB)*, Technical Report, 81, CS, Monash University, Australia, 1986.

Meng 02 Meng, L., *Generalisierung von Geodaten - Notwendigkeiten, Möglichkeiten und Hemmnisse*, KN 1/2002, Kirsch Baum Verlag, pp. 7-13, 2002.

Menon & Smith 89 Menon, S., Smith, T. R., *A Declarative Spatial Query Processor for Geographic Information Systems*, Photogrammetric Engineering and Remote Sensing, 55(11), pp. 1593-1600, 1989.

Meyer 92 Meyer, B., *Beyond Icons: Towards New Metaphors for Visual Query Languages for Spatial Information Systems*, In: Cooper, R. (editor), Interfaces to Database Systems, Berlin, Springer, 1992.

Mitschang 95 Mitschang, B., *Anfrageverarbeitung in Datenbanksystemen - Entwurfs- und Implementierunskonzepte*, Vieweg, 1995.

Mohan & Kashyap 88 Mohan, L., Kashyap, R., *An Object-Oriented Knowledge Representation for Spatial Information*, IEEE TOSE, 14, pp. 675-681, 1988.

Morton 66 Morton, G. M., *A Computer Oriented Geodetic Data Base and a New Technique in File Sequencing*, IBM Ltd., Ottawa, Canada, 1966.

SQL/MM CWB-038 Scarponcini, P., *Resolving OGIS and SQL/MM Spatial Differences*, ISO/IEC JTC 1 SC 32 WG 4 SQL/MM:CWB-038, March 9, 1998.

SQL/MM FRA-004 98 ISO/IEC JTC 1 SC 32 WG 4 SQL/MM:FRA-004, *SQL Multimedia and Application Packages (SQL/MM) - Part 3: Spatial*, ISO, September 6, 1998.

Neumann 88 Neumann, K.-H., *Eine geowissenschaftliche Datenbanksprache mit benutzerdefinierbaren geometrischen Datentypen*, Dissertation, TU Braunschweig, 1988.

Oesterreich 97 Oesterreich, B., *Objektorientierte Softwareentwicklung mit der UML*, Oldenbourg, 1997, 3. Aufl., 1997.

OGC 98 OGC, *OpenGIS Consortium Homepage*, http://www.opengis.org/, 1998.

OGC AS 98 OGC, *OpenGIS Abstract Specification*, http://www.opengis.org/techno/specs.htm#abstract, 1998.

OGC SF SQL 98 OGC, *OGC Simple Features Specification for SQL*, http://www.opengis.org/techno/specs.htm#implementation, 1998.

Ooi 88 Ooi, B. C., *Efficient Query Processing for Geographic Information Systems*, PhD Thesis, Monash University, Victoria, Australia, 1988.

Ooi 90 Ooi, B. C., *Efficient Query Processing in Geographic Information Systems*, LNCS 471, 1990.

Ooi & Sacks-Davis 89 Ooi, B. C., Sacks-Davis, R., *Query Optimization in an Extended DBMS*, In: Litwin, W., Schek, H.-J. (editors), Foundations of Data Organization and Algorithms - 3rd Int. Conf. FODO'89, Paris, June 1989, LNCS 367, pp. 247-258, 1989.

Ooi et. al. 89 Ooi, B. C., Sacks-Davis, R., McDonell, K. J., *Extending a DBMS for Geographic Applications*, Proceedings of the IEEE 5th International Conference on Data Engeneering, Los Angeles, pp. 590-597, 1989.

Oosterom & Vijlbrief 91 Oosterom, P. van, Vijlbrief, T., *Building a GIS on Top of the Open DBMS POSTGRES*, 2nd European Conference on Geographical Information Systems EGIS '91, Brussels, 1991.

Oracle 98 Oracle, *Oracle Corporation Homepage*, http://www.oracle.com/, 1998.

Orenstein & Manola 86 Orenstein, J. A., Manola, F. A., *Spatial Data Modeling and Query Processing in PROBE*, Technical Report CCA-86-05, Computer Corporation of America, Cambridge, MA, 1986.

Orenstein et. al. 88 Orenstein, J. A., Frank, A., Manola, F. A., *PROBE Spatial Data Modeling and Query Processing in an Image Database Application*, IEEE TOSE, 14(5), pp. 611-629, 1988.

Orenstein & Merrett 84 Orenstein, J. A., Merrett, T., *A Class of Data Structures for Associative Searching*, In: Proceedings of SIGART-SIGMOD 3rd Symposium on Principles of Database Systems, pp. 181-190, Waterloo, Canada, 1984.

Paiva & Egenhofer 95 Paiva, J., Egenhofer, M., *Topological Equivalence of Regions with Holes: The Concepts and an Incremental Algorithm*, Technical Report, NCGIA, University of Maine, Orono, ME, 1995.

Paredaens et. al. 94 Paredaens, J., Bussche, J. Van den, Gucht, D. Van, *Towards a Theory of Spatial Database Queries*, Proc. ACM Symp. on Principles of Database Systems, pp. 279-288, 1994.

Paredaens & Kuijpers 98 Paredaens, J. Kuijpers, B., *Data Models and Query Languages for Spatial Databases*, Data & Knowledge Engineering, 25, pp. 29-53, Elsevier, 1998.

Radhakrishan et. al. 81 Radhakrishan, T., Barrera, R., Buchmann, A. P., *An Architecture for Geographical Data Base Systems*, INFORMATICS '81, New Delhi, India, IIMAS Tech. Rep. No. 241, 1981.

Robinson 90 Robinson, V. B., *Spatial Query Languages*, In: J.-C. Muller (editor), Advances in Cartography, Elsevier Applied Science on behalf of International Cartographic Association, pp. 89-111, 1990.

Roschlaub 98 Roschlaub, R., *Klassifikation und Interpolation mittels affin invarianter Voronoidiagramme auf der Basis eines Wahrscheinlichkeitsmaßes in großmaßstäbigen Geoinformationssystemen*, Doktorarbeit, TU München, Fachgebiet GIS, 1998.

Rowe & Stonebraker 87 Rowe, L., Stonebraker, M., *The POSTGRES Data Model*, 13th VLDB, Brighton, England, pp. 83-96, 1987.

Rumbaugh et. al. 93 Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., Lorensen, W., *Objektorientiertes Modellieren und Entwerfen*, Hanser, 1993.

Sacks-Davis et. al. 87 Sacks-Davis, R., McDonell, K. J., Ooi, B. C., *GEOQL - a Query Language for Geographic Information Systems*, Technical Report 87/2, Monash University, Victoria, Australia, 1987.

Samet 81 Samet, J. (Editor), *Query Languages - a Unified Approach*, Report of the British Computer Society Query Languages Group, Heyden University Press, Cambridge, England, 1981.

Samet 84 Samet, J., *The Quadtree and Related Hierarchical Data Structures*, ACM CS, 16(2), pp. 187-260, 1984.

Samet 89a Samet, J., *Applications of Spatial Data Structures: Computer Graphics, Image Processing, and GIS*, Addison Wesley, MA, 1989.

Samet 89b Samet, J., *Hierarchical Spatial Data Structures*, 1st SSD, Santa Barbara, LNCS 409, pp. 193-212, 1989.

Samet 89c Samet, J., *The Design and Analysis of Spatial Data Structures*, Addison Wesley, MA, 1989.

Sicad 98 Sicad, *Sicad Geomatics Homepage*, http://www.sicad.com/, http://www.sicad.de/, 1998.

Schäfer 97 Schäfer, S., *Anbindung von Wincat an IUS*, unveröffentlichter Praktikumsbericht, Universität der Bundeswehr, SNI, 1997.

Schek 87 Schek, H. J., *Ein Datenbank-Kernsystem für anwendungsspezifische Schichten - Architektur der DASDBS-Familie*, Informationstechnik, 3, pp. 153-164, 1987.

Schek & Weikum 86 Schek, H. J., Weikum, G., *DASDBS: Concepts and Architecture of a Database System for Advanced Applications*, TU Darmstadt, West Germany, Report DVSI-1986-T1, 1986.

Schek et. al. 90 Schek, H.-J., Paul, H. B., Scholl, M. H., Weikum, G., *The DASDBS Project: Objectives, Experiences, and Future Prospects*, IEEE TKDE, 2(1), pp. 25-43, 1990.

Schilcher 97 Schilcher, M., *Der GIS-Markt im Umbruch*, GIS 96, Das Praxisforum für Anwender, Unterlagen zum Vortrag, 1996.

Schilcher et. al. 00 Schilcher, M., Pichelmann, M., Plabst, S., *GIS Glossar*, TU München, Geodätisches Institut, http://www.gis.bv.tum.de/, Oktober 2000.

Scholl & Voisard 89 Scholl, M., Voisard, A., *Thematic Map Modeling*, 1st SSD, Santa Barbara, LNCS 409, pp. 167-190, 1989.

Schwarz et. al. 86 Schwarz, P., Chang, W., Freytag, J., Lohman, G., McPherson, J., Mohan, C., Pirahesh, H., *Extensibility in the Starburst Database System*, Proc. of the IEEE-ACM Int. Workshop on Object-Oriented Database Systems, Pacific Grove, CA, pp. 85-92, 1986.

Sellis et. al. 87 Sellis, T., Roussopoulos, N., Faloutsos, C., *The R+-Tree: A Dynamical Index for Multi-Dimensional Objects*, 13th VLDB, Brighton, pp. 507-518, 1987.

Shariff et. al. 98 Shariff, A. R. B. M., Egenhofer, M. J., Mark, D. M., *Natural-Language Spatial Relations between Linear and Areal Objects: the Topology and Metric of English-Language Terms*, IJGIS, 12(3), pp. 215-245, 1998.

Singer 93 Singer, C., *Relationale Datenbanksysteme als Basis Geographischer Datenbanken*, GIS, 6(6), Wichmann, Karlsruhe, 1993.

Smith et. al. 87 Smith, T. R., Menon, S., Star, J. L., Estes, J. E., *Requirements and Principles for the Implementation and Construction of Large Scale Geographic Information Systems*, IJGIS, 1(1), pp. 13-31, 1987.

Stonebraker 92 Stonebraker, M., *An Overview of the Sequoia 2000 Project*, Sequoia 2000 Technical Report, 91/5, Computer Science Division, EECS Departement, UCB, 1992.

Stonebraker 96 Stonebraker, M., Moore, D., *Object-relational DBMSs - the next great wave*, Morgan Kaufmann Publishers, San Francisco, CA, 1996.

Stonebraker & Rowe 86 Stonebraker, M., Rowe, L. A., *The Design of POSTGRES*, ACM SIGMOD, Washington, D.C., pp. 340-355, 1986.

Stonebraker et. al. 90 Stonebraker, M., Rowe, L., Hirohama, M., *The Implementation of POSTGRES*, ACMIEEE TKDE, 2(1), pp. 125-142, 1990.

Stonebraker et. al. 93 Stonebraker, M., Frew, J., Gardels, K., Meredith, J., *The Sequoia 2000 storage benchmark*, ACM SIGMOD, Washington, D.C., 1993.

SSD 91 Günther, O., Schek, H.-J. (editors), Proceedings of the Second International Symposium on Large Spatial Databases, Zürich, LNCS 525, 1991.

SSD 95 Egenhofer, M. J., Herring, R. (editors), Proceedings of the Forth International Symposium on Large Spatial Databases, Portland, LNCS 951, 1995.

Svensson & Huang 91 Svensson, P., Huang, Z., *GeoSAL: A Query Language for Spatial Data Analysis*, 2nd SSD, Zürich, LNCS 525, pp. 119-140, 1991.

TC 211 N 549 ISO TC 211 WG 2, *1st combinded working draft of ISO 15046-7 Geographic Information - Spatial Schema and ISO 15046-20 Geographic Information - Spatial Operators*, ISO TC 211 Geographic Information/Geomatics, 1998.

TC 211 Scope 98 ISO TC 211, *TC 211 Scope*, http://www.statkart.no/isotc211/scope.htm, 1998.

Vassiliou & Jarke 84 Vassiliou, Y., Jarke, M., *Query Languages - a Taxonomy*, In: Vassiliou, Y. (editor), Human Factors and Interactive Computer Systems, Ablex, Norwood, New Jersey, pp. 47-81, 1984.

Vijlbrief & Oosterom 92 Vijlbrief, T., Oosterom, P. van, *The GEO++ System: An Extensible GIS*, 5th SDH, Charleston, SC, pp. 40-51, 1992.

Ubeda & Egenhofer 97 Ubeda, T., Egenhofer, M. J., *Topological Error Correcting in GIS*, 5th SSD, Berlin, LNCS 1262, pp. 283-297, 1997.

Vossen 94 Vossen, G., *Datenmodelle, Datenbanksprachen und Datenbankmanagement-Systeme*, Addison-Wesley, Bonn, 1994.

Waugh & Healey 86 Waugh, T. C., Healey, R. H., *The GEOVIEW Design: a Relational Database Approach to Geographical Data Handling*, 2nd SDH, Seattle, pp. 193-212, 1986.

Waugh & Healey 87 Waugh, T. C., Healey, R. H., *The GEOVIEW Design: a Relational Database Approach to Geographical Data Handling*, IJGIS, 1(2), pp. 101-118, 1987.

Wellar 95 Wellar, B., *Evaluating Information Systems Performance Using Informational Activity Criteria*, interner Bericht, 33rd URISA, pp. 97-111, 1995.

Westwood & Brinkman 88 Westwood, K. A., Brinkman, J. P., *Toward the Successful Integration of Relational and Quadtree Structures in a Geographic Information System*, Urban and Regional Information Systems Association, pp. 181-189, 1995.

Widmayer 91 Widmayer, P., *Datenstrukturen für Geodatenbanken (data structures for spatial databases)*, In: Vossen, G., Witt (editors), Entwicklungstendenzen bei Datenbanksystemen (future trends in database systems), pp. 317-361, Oldenburg, München, 1991.

Wolf 89 Wolf, A., *The DASDBS Geo-Kernel: Concepts, Experiences, and the Second Step*, 1st SSD, Santa Barbara, LNCS 409, pp. 67-88, 1989.

Wolf 90 Wolf, A., *How to Fit Geo-Objects Into Databases - An Extensibility Approach*, Proc. 1st European Conf. on Geographical Information Systems EGIS'90, Amsterdam, 1990.

Worboys 95 Worboys, M., *GIS: a Computing Perspective*, Taylor & Francis, London, 1995.

Worboys et. al. 90a Worboys, M., Hearnshaw, H., Maguire, D., *Object-Oriented Data Modelling for Spatial Databases*, IJGIS, 4(4), pp. 369-383, 1990.

Worboys et. al. 90b Worboys, M., Hearnshaw, H., Maguire, D., *Object-Oriented Data and Query Modelling for Spatial Databases*, 4th SDH, Zürich, pp. 679-688, 1990.

Wu 88 Wu, J.-K., *Design ofa Knowledge-Based Geographic Information System*, Technical Report, University of Science and Technology of China, Hefei, People´s Republic of China, 1988.

Wu et. al. 89 Wu, J.-K., Chen, T., Yang, L., *QPF - a Versatile Query Language for a Knowledge-Based Geographical Information System*, IJGIS, 3(1), pp. 51-59, 1989.

Ziegler 97 Ziegler, M., *Analyse der Anforderungen von Anwendern an geographische Anfragesprachen am Beispiel der Kartographischen Anstalt*, interner Bericht, unveröffentlicht, 1997.

Zloof 75 Zloof, M. M., *Query-by-Example*, AFIPS Conference Proceedings, 44, pp. 431-438, 1975.

Zwart & Greener 94 Zwart, P., Greener, S., *Some Results and Implications of Performance Tests on Large Nationwide Spatial Databases*, Proc. AURISA, pp. 135-142, 1994.

A. Anhang

A.1Ablauf des systematischen Funktionalitätstests

Der Aufbau der Testdatenbank für den systematischen Funtionalitätstest von GQL wird in Kapitel 6.1.1 beschrieben. Dieses Kapitel gibt den Ablauf seiner Ausführung wieder.

Erzeugung der Testdaten

Die Tabelle der GQL-Testfälle (Tabelle 33) steuert das automatische Erzeugen der Testdaten. Ein Testfall ist eindeutig definiert durch einen Operator (Operatorname) und die als erster

GqlTestFälle										
Testfall Nr.	Operator Name	Parameter1	Parameter2	Element1	Element2	Status	unten	links	oben	rechts

Tab. 33: Tabelle der GQL-Testfälle

und zweiter Parameter angegebenen Elementtypen (Element1, Element2), z. B. Test des Operators contains mit Flächen (FL) und Linien (LI). Jeder Testfall entspricht einem 1000 x 1000 Testfeld und hat eine eindeutige Testfall-Nr.. Aus den jeweiligen Elementtypen lassen sich nach Tabelle 36 die zugehörigen Parametertypen ableiten, sie sind hier der Übersicht halber mit aufgeführt.

Abhängig vom Wert der Spalte status erzeugen die Prozeduren die Testdaten, führen den Test durch und/oder löschen die Daten. Die Spalten haben die folgenden Wertebereiche:

- Operatorname ist einer von {contains, cross, disjoint, equals, intersect, overlaps, touch, within};
- Parameter1, Parameter2 ist einer von{Fläche, Linie, Punkt};
- Werte für Element1, Element2 sind z. B. Kreis (KR) oder Linienzug (LY) (siehe Tabellen 6 und 36);
- Status ist einer von {Einzellauf, Testen, Füllen, Löschen, nicht Testen}; die Aktionen derTestprozeduren in Abhängigkeit vom Status und wie sie den Status ändern sind in Abbildung 63 erläutert.

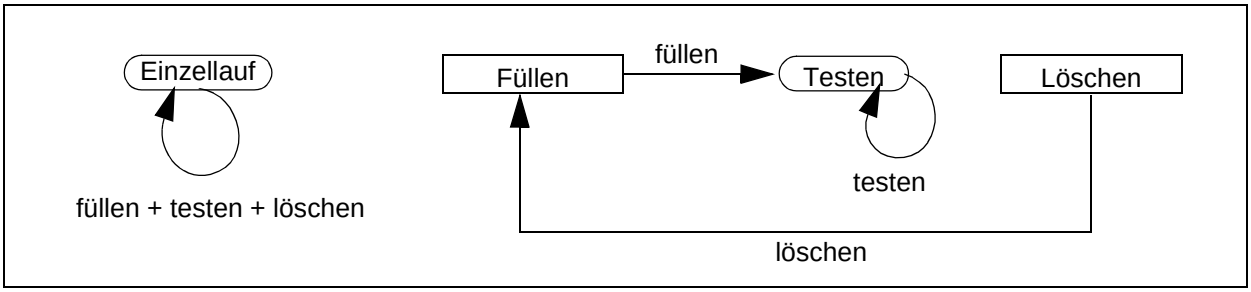


Abb. 63: Zustände von Testfällen und zugehörige Aktionen

- unten, links, oben, rechts können bei folgendem Algoritmus berechnet werden

```
for operators op from 0 to 7
    for parameter_type_1 pt1 from 0 to 4
        for parameter_type_2 pt2 from 0 to 4
            for elements first param el(pt1) from 0 to max(el(pt1))
                for elements second param el(pt2) from 0 to max(el(pt2))
                    if supported[op, pt1, pt2] = TRUE
                        links   =    0   + 2500 pt1    + 100 el(pt1)
                        unten   = 10000 op + 2500 pt2    + 100 el(pt2)
                        rechts  =    0   + 2500 (pt1+1) + 100 ((elpt1)+1)
                        oben    = 10000 op + 2500 (pt2+1) + 100 ((elpt2)+1)
```

Ablauf der Testroutinen

Aufbauend auf den oben beschriebenen Testdaten werden Tests durchgeführt. Elemente für die der Operator zutrifft, müssen in der Ergebnismenge sein (positiver Test), umgekehrt dürfen Elemente nicht fälschlich in der Ergebnismenge sein, wenn der Operator nicht zutrifft (negativer Test). Der Einfluß verschiedener Werte von Toleranz auf das Testergebnis ist zu überprüfen. Nicht unterstützte Elemente müssen korrekt behandelt werden.

Es gibt drei Ebenen von funktionalen Tests. Sie überprüfen ob die Ergebnismenge die korrekte Anzahl von Elementen enthält, ob exakt die erwarteten Elemente durch die Bedingung im WHERE-Teil ausgewählt worden sind und ob exakt die erwartete Ausgabe von der Projektion im SELECT-Teil ausgegeben worden ist. Sie werden im folgenden erläutert.

1. Korrekte Anzahl von Elementen in der Ergebnismenge

   Die Anzahl der Elemente in der Ergebnismenge wird mit einem SQL-Befehl der Art

   ```
   SQL SELECT count(*) INTO <GIS-Variable> FROM <gqlquery>
   ```

   in eine GIS-Variable eingelesen. Stimmt ihr Wert nicht mit dem erwarteten überein, wird ein Prozedur zur Dokumentation des Fehlers aufgerufen:

   ```
   IF <GIS-Variable> NE <expected> THEN document_error();
   ```

   Bei dieser Art von Test wird das erwartete Ergebnis direkt in der Testprozedur abgelegt. Dies hat den Vorteil einfacherer Handhabung. Es werden keine zusätzlichen Tabellen benötigt.

2. Exakt die erwarteten Elemente ausgewählt

   Zur Vorbereitung dieses Testlaufs werden noch zusätzliche Informationen benötigt. Diese werden in drei Tabellen gehalten.

   Die Tabelle der erwarteten Testergebnisse (GqlTestErgebnisse, Tabelle 33) zeigt zu jedem Element identifiziert durch seinen Namen und den Testfall ob es im Testlauf j zum Ergebnis gehören soll (IN) oder nicht (NOT IN). In der Tabelle der getesteten Elemente

(GqlTestElemente, Tabelle 35) werden jedem Element in der Datenbank (location) ein innerhalb des Testfalls eindeutiger Name zugeordnet.

GqlTestErgebnisse			
Testfall-Nr.	Objektname	Testlauf	Ergebnis
n	Linie_17	i	IN
n	Linie_17	j	NOT IN

Tab. 34: Tabelle der erwarteten Testergebnisse

GqlTestElemente		
Testfall-Nr.	Objektname	location
n	Linie_17	
n	Punkt_18	

Tab. 35: Tabelle der getesteten Elemente

Das Objekt ‘Linie_17’ wird dynamisch von der Testprozedur erzeugt. Seine Elementnummer (location) und sein Name werden in GqlTestElemente eingefügt. Für jeden durchzuführenden Testlauf wird das erwartete Ergebnis in GqlTestErgebnisse eingetragen. Das folgende Beispiel zeigt die Erzeugung und das Füllen der Testtabellen in einer Testprozedur:

- Anlegen der Tabellen GqlTestErgebnisse, GqlTestElemente und GqlTestFälle, falls nicht schon vorhanden

```
gbneu test2 0 0 1000 1000 spt=XXS

sql create table GqlTestErgebnisse (
        TestfallNr.             number(10),
        Objektname              char(20),
        Testlauf                number(10),
        Ergebnis                char(5)
        )

sql create table GqlTestElemente (
        TestfallNr              number(10),
        Objektname              char(20),
        location                char(20)
        )

sql create table GqlTestFälle(
        TestfallNr              number(10),
        Operatorname            char(20),
        Parameter1              char(5),
        Parameter2              char(5),
        Element1                char(2),
        Element2                char(2),
        Status                  char(10),
        unten                   number(10),
        links                   number(10),
        oben                    number(10),
        rechts                  number(10)
        )
```

- Verbindung zur Datenbank herstellen, Erzeugen und Aktivieren der Tabellen- und Verknüpfungsdefinitionen

```
gbnam test2 ziegler 0
GBCRETD GqlTestElemente 'TestfallNr Objektname' L
GBCRELD GqlTestErgebnisse TestfallNr TestfallNr DESC WRITE
GBCRELD GqlTestErgebnisse Objektname objname DESC WRITE
gbnam test2 ziegler 0
```

- Auschecken des Gebiets für einen Testfall, Erzeugen der Elemente, Speichern der Elemente, Verknüpfen der Elemente mit den erwarteten Ergebnissen durch ihren Namen und die Element-ID (location).

```
SQL
SELECT    TestfallNr, Status, unten, links, oben, rechts
INTO      %I2, %T ...
FROM      GqlTestFälle
WHERE     Operatorname='cross'
AND       Element1='LI'
AND       Element2='LY'


xxx

if status = empty then continue else end xxx

gbler links unten rechts oben mit kuerzungswerten? xxx

li 10 10 100 100
pas li TestfallNr=%I2
pas li objname='Linie 17'
GBSBEF li ESGqlTestElemente

sql insert into GqlTestErgebnisse values(%I2, 'Linie 17', 1, 'IN')
sql insert into GqlTestErgebnisse values(%I2, 'Linie 17', 2, 'NOT IN')
sql insert into GqlTestErgebnisse values(%I2, 'Linie 17', 3, 'NOT IN')


li 100 10 10 100
pas li TestfallNr=%I2
pas li objname='Linie 18'
GBSBEF li ESTestErgebnisse

sql insert into GqlTestErgebnisse values(%I2, 'Linie 18', 1, 'NOT IN')
sql insert into GqlTestErgebnisse values(%I2, 'Linie 18', 2, 'IN')
sql insert into GqlTestErgebnisse values(%I2, 'Linie 18', 3, 'NOT IN')


gbsav
```

Starten von Test 1 und Einfügen der der Ergebnisse in eine Elementmenge.

```
sql insert into actual
select location from ... <gql query>
```

z. B.

```
SQL     insert into actual
SELECT  t1.location
FROM    GqlTestElemente e1, GqlTestElemente e2
WHERE   cross(t1.location, t2.location)
AND     e1.testcase=%TESTCASE
AND     e2.testcase=%TESTCASE
```

Anschließend wird mit folgendem Befehl überprüft werden, ob das Ergebnis korrekt ist:

```
SELECT  count(*) into %I1 from GqlTestElemente e, GqlTestErgebnisse r
WHERE   e.TestfallNr=%TESTCASE
AND     e.TestfallNr=r.TestfallNr
AND     e.Objektname=r.Objektname
AND     (   ( Ergebnis='NOT IN' AND location in (select location from actual)
            OR( Ergebnis='IN' AND location NOT in (select location from actual)
            )

IF (%I1.NE.0) dop check error=.T. result='OK'
IF (%GL0) GOTO end
```

3. Exakt die erwartete Ausgabe ausgegeben

   Mit den bisher beschriebenen Testverfahren lassen sich bereits die Bedingungen im WHERE-Teil effektiv überprüfen. Um zu testen, ob im SELECT-Teil verwendete Funktionen das richtige Ergebnis zurückliefern, wurde auf Testmethoden der GDB-X-Entwicklung zuückgegriffen. Sie werden nicht weiter erläutert.

Ergebnis

Alle Operatoren und Funktionen arbeiten korrekt entsprechend den Spezifikationen.

A.2Prozeduren des S2K-Benchmarks mit GQL

ad 1. Erzeugen der Datenbank, Laden der Daten, Erzeugen der Indizes

```
PROCN

DOCBEG
KOM '*************************************************************'
KOM 'Query 1 - Load data into DBMS and make all neccessary indexes'
KOM '*************************************************************'
DOCEND

PARBEG

IN VER %i6 PTX='Version (1=plain 2=ld+td 3=ld+td+atw)'
IN KILL %i7 PTX='Kill existing data (1=yes, 0=NO)'

PAREND

***************
******************** Delete all existing data **************
*************

*
* For all versions the same procedure
*

KOM 'Prepare query 1'
if (%i7.eq.0) goto nokill

KOM '-- delete current db'; dataout
msgsup; gbdeldb current

KOM '-- clear table gbaudit and perf'; dataout
msgsup; sql delete from gbaudit
msgsup; sql delete from perf

KOM '-- clear table luse'; dataout
msgsup; sql delete from luse

KOM '-- clear table point'; dataout
msgsup; sql delete from point

KOM '-- clear table poly'; dataout
msgsup; sql delete from poly

.nokill *
KOM '-- clear all definitions like table definition'; dataout
msgsup; gbdeltd point
msgsup; gbdeltd poly
msgsup; gbdelld point name txt
msgsup; gbdelld poly name nam
msgsup; gbatw 0 0 0
gbnam bench holger 2

************************************************************
* If not version 1 then try version 2

if (%i6.ne.1) goto v2

***************
```

```
****************** Version 1 ******************************
************
* Query 1 V1 is running!

%t10='q1_1'

KOM ' '
KOM 'Query 1_1 - Load data'

* If data already existing goto end without loading data a second time
* Usage has no sense on query 1_1 because we need at least one query result

if (%i7.eq.0) goto ende

goto load

************************************************************
.v2 *

***************
****************** Make all definitions ******************
************
*
* For versions 2 and 3 the same procedure
*
* Set flag to recognize data on later use of the values

%t10='q1_defs'

KOM '-- Make definitions'
KOM '---- Table def. point'; dataout
gbaudit dpl=off rdb=on
gbcretd point 'name' l
gbaudit dpl=off rdb=off

sql select max(gbcmdnumber) into %R99 from gbaudit
sql insert into perf(qname,cmdnumber) values (%t10,%R99)

KOM '---- Linkage def. point'; dataout
gbaudit dpl=off rdb=on
gbcreld point name txt desc write
gbaudit dpl=off rdb=off

sql select max(gbcmdnumber) into %R99 from gbaudit
sql insert into perf(qname,cmdnumber) values (%t10,%R99)

KOM '---- Table def. poly'; dataout
gbaudit dpl=off rdb=on
gbcretd poly 'name' l
gbaudit dpl=off rdb=off

sql select max(gbcmdnumber) into %R99 from gbaudit
sql insert into perf(qname,cmdnumber) values (%t10,%R99)

KOM '---- Linkage def. poly'; dataout
gbaudit dpl=off rdb=on
gbcreld poly name nam par write
gbaudit dpl=off rdb=off

sql select max(gbcmdnumber) into %R99 from gbaudit
sql insert into perf(qname,cmdnumber) values (%t10,%R99)
```

```
KOM ’---- Reconnect to DB’; dataout
gbnam bench holger 2
KOM ’---- connected!’; dataout

************************************************************
* If not version 2 then try version 3

if (%i6.ne.2) goto v3

***************
******************** Version 2 ****************************
************
* Query 1 V2 is running!

%t10=’q1_2’

KOM ’ ’
KOM ’Query 1_2 - Load data’

* If data exist then need not load data a second time
if (%i7.eq.0) goto ende

goto load

************************************************************
.v3 *
***************
******************** Set the ATW filter *********************
************
*
* Only needed for version 3
*
* Flag to recognize q1_3

%t10=’q1_atw’

KOM ’---- set ATW filter’; dataout
gbaudit dpl=off rdb=on
msgsup; gbatw pg point txt
gbaudit dpl=off rdb=off

sql select max(gbcmdnumber) into %R99 from gbaudit
sql insert into perf(qname,cmdnumber) values (%t10,%R99)


***************
******************** Version 3 ****************************
************
* If data exist then need not load data a second time
* query 1 V3 is running!

%t10=’q1_3’

KOM ’ ’
KOM ’Query 1_3 - Load data’

if (%i7.eq.0) goto ende

************************************************************
.load *
KOM ’-- Load landuse data from sqd file’; dataout
```

```
gbaudit dpl=off rdb=on
gbqgsa 0  /usr1/people/ziegler/s2k/other/landuse.sqd
gbaudit dpl=off rdb=off

sql select max(gbcmdnumber) into %R99 from gbaudit
sql insert into perf(qname,cmdnumber) values (%t10,%R99)

************************************************************
KOM '-- Load point data into layer 1 from sqd file'; dataout

KOM '---- ca.sqd'; dataout
gbaudit dpl=off rdb=on
gbqgel a  /usr1/people/ziegler/s2k/point/ca.sqd
gbaudit dpl=off rdb=off
sql select max(gbcmdnumber) into %R99 from gbaudit
sql insert into perf(qname,cmdnumber) values (%t10,%R99)


KOM '---- 2'; dataout
gbaudit dpl=off rdb=on
gbqgel a  /usr1/people/ziegler/s2k/point/ca.2.sqd
gbaudit dpl=off rdb=off
sql select max(gbcmdnumber) into %R99 from gbaudit
sql insert into perf(qname,cmdnumber) values (%t10,%R99)

...

KOM '---- 13'; dataout
gbaudit dpl=off rdb=on
gbqgel a  /usr1/people/ziegler/s2k/point/ca.13.sqd
gbaudit dpl=off rdb=off
sql select max(gbcmdnumber) into %R99 from gbaudit
sql insert into perf(qname,cmdnumber) values (%t10,%R99)


************************************************************
KOM '-- Load polygondata into layer 2 from sqd file'; dataout

KOM '---- 0.sqd'; dataout
gbaudit dpl=off rdb=on
gbqgel a  /usr1/people/ziegler/s2k/poly/0.sqd
gbaudit dpl=off rdb=off
sql select max(gbcmdnumber) into %R99 from gbaudit
sql insert into perf(qname,cmdnumber) values (%t10,%R99)

**********

KOM '---- 11.sqd'; dataout
gbaudit dpl=off rdb=on
gbqgel a  /usr1/people/ziegler/s2k/poly/11.sqd
gbaudit dpl=off rdb=off
sql select max(gbcmdnumber) into %R99 from gbaudit
sql insert into perf(qname,cmdnumber) values (%t10,%R99)

KOM '---- 12.sqd'; dataout
gbaudit dpl=off rdb=on
gbqgel a  /usr1/people/ziegler/s2k/poly/12.sqd
gbaudit dpl=off rdb=off
sql select max(gbcmdnumber) into %R99 from gbaudit
sql insert into perf(qname,cmdnumber) values (%t10,%R99)

...
```

```
KOM '---- 92.sqd'; dataout
gbaudit dpl=off rdb=on
gbqgel a  /usr1/people/ziegler/s2k/poly/92.sqd
gbaudit dpl=off rdb=off
sql select max(gbcmdnumber) into %R99 from gbaudit
sql insert into perf(qname,cmdnumber) values (%t10,%R99)

***************

KOM '---- islands.sqd'; dataout

gbaudit dpl=off rdb=on
gbqgel a  /usr1/people/ziegler/s2k/poly/islands.sqd
gbaudit dpl=off rdb=off
sql select max(gbcmdnumber) into %R99 from gbaudit
sql insert into perf(qname,cmdnumber) values (%t10,%R99)

gbaudit dpl=off rdb=on
gbqgel a  /usr1/people/ziegler/s2k/poly/islands.1.sqd
gbaudit dpl=off rdb=off
sql select max(gbcmdnumber) into %R99 from gbaudit
sql insert into perf(qname,cmdnumber) values (%t10,%R99)

...

gbaudit dpl=off rdb=on
gbqgel a  /usr1/people/ziegler/s2k/poly/islands.16.sqd
gbaudit dpl=off rdb=off
sql select max(gbcmdnumber) into %R99 from gbaudit
sql insert into perf(qname,cmdnumber) values (%t10,%R99)

************************************************************
***************
******************** Save performance data ******************
*************

.ende *


KOM 'Transfering performance-data into bwerte-table'; dataout

sql insert into bwerte (qname,\
                    cmdnumber,cmdname,\
                    elapsed,cputime,\
                    elembr,elembw,\
                    elemlr,elemlw,\
                    cellbr,cellbw,\
                    celllr,celllw,\
                    bytebr,bytebw) \
               select p.qname,\
                    gbcmdnumber,gbcmdname,\
                    gbelapsed,gbcputime,\
                    gbelembr,gbelembw,\
                    gbelemlr,gbelemlw,\
                    gbcellbr,gbcellbw,\
                    gbcelllr,gbcelllw,\
                    gbbytebr,gbbytebw \
   from perf p, gbaudit \
     where gbcmdnumber=p.cmdnumber
```

```
*
*

ENDP
```

ad 5. nicht-räumliche Selektion: Suche einen Punkt mit einem Namen n

```
PROCN

DOCBEG
KOM '************************************************************'
KOM 'Query 5 - Find the Point record that has a specific name'
KOM '************************************************************'
DOCEND

PARBEG

IN NOQR %i50  PTX='No. of query repititions'
IN VER %i56 PTX='Version of query (gdbx:1=plain data,2=ld+td,3=td+ld+atw; gql:4,5,6)'

PAREND

********************
* make clean
********************

*sql whenever sqlerror continue
sql delete from gbaudit
sql delete from perf

*****************************
* find out the version
*****************************

if (%i56.ne.1) goto v2
%t51='q5gdbx_1'
goto cont

.v2 if (%i56.ne.2) goto v3
%t51='q5gdbx_2'
goto cont

.v3 if (%i56.ne.3) goto v4
%t51='q5gdbx_3'
goto cont

.v4 if (%i56.ne.4) goto v5
%t51='q5gql_1'
goto cont

.v5 if (%i56.ne.5) goto v6
%t51='q5gql_2'
goto cont

.v6 %t51='q5gql_3'

.cont *



*******************************************
```

```
* Start Query 5
**********************************************

KOM ' '
KOM 'Query 5 ('//%t51//') - Find the Point record that has a specific name'; dataout

*
* clear old elementset
*

KOM '-- delete old data'; dataout
gbresem elementset

sql delete from elementset2

*
* create cursor for query 5
*

KOM '-- declare a cursor'; dataout
sql begin work
sql declare c_q5 cursor for select name from random where type='p'
sql open c_q5

KOM '-- go into loop for benchmarking...'; dataout

if(%i50.le.1) goto noloop


DO %i59 1 %i50

  sql fetch c_q5 into %t52
  KOM '---- name= '//%t52; dataout

  if (%i56.ge.4) goto gql


**********
* GDB-X
**********
  gbaudit dpl=off rdb=on

  sql insert into elementset2 (location)\
     select location from point where name=%t52

  gbaudit dpl=off rdb=off
  goto nogql

**********


.gql *

*********
* GQL
*********
  gbaudit dpl=off rdb=on

  sql insert into elementset2\
     select location from point\
        where value_of(location,txt)=%t52
```

```
  gbaudit dpl=off rdb=off

*********

.nogql *


*
* save actual performance-values-identification into table perf
*

  sql select max(gbcmdnumber) into %R99 from gbaudit
  sql insert into perf(qname,cmdnumber) values (%t51,%R99)

ENDDO

goto weiter

**************************
* On error write this
**************************

.noloop KOM '---- NO LOOP!'; dataout
.weiter *

***********************
* save log data
***********************

KOM '-- write found elements into file:  '//%t51//'.001!'
gbgqsa 1 elementset2 0 fil=(%t51) 0 0 0 0

KOM '-- closing cursor and commit work'; dataout
sql close c_q5
sql commit work

KOM ' '
KOM 'Transfering performance-data into bwerte-table'; dataout

sql insert into bwerte (qname,\
                        cmdnumber, cmdname,\
                        elapsed,cputime,\
                        elembr,elembw,\
                        elemlr,elemlw,\
                        cellbr,cellbw,\
                        celllr,celllw,\
                        bytebr,bytebw\
                       )\
                select p.qname,\
                        gbcmdnumber,gbcmdname,\
                        gbelapsed,gbcputime,\
                        gbelembr,gbelembw,\
                        gbelemlr,gbelemlw,\
                        gbcellbr,gbcellbw,\
                        gbcelllr,gbcelllw,\
                        gbbytebr,gbbytebw\
    from perf p, gbaudit where gbcmdnumber=p.cmdnumber



*sql whenever sqlerror stop
```

```
*
*

ENDP
```

ad 6. räumliche Selektion (mit Erzeugung einer neuen Tabelle): Suche alle Polygone, die sich mit einem Rechteck r überschneiden.

```
PROCN

DOCBEG
KOM '************************************************************'
KOM 'Query 6 - Find all polygons that intersect a rectangle'
KOM '************************************************************'
DOCEND

PARBEG

IN NOQR %i60  PTX='No. of query repititions'
IN VER %i66 PTX='Version of query (gdbx:1=plain data,2=ld+td,3=td+ld+atw; gql:4,5,6)'

PAREND

********************
* make clean
********************

*sql whenever sqlerror continue
sql delete from gbaudit
sql delete from perf

sql delete from elementset2

*****************************
* find out the version
*****************************

if (%i66.ne.1) goto v2
%t61='q6gdbx_1'
goto cont

.v2 if (%i66.ne.2) goto v3
%t61='q6gdbx_2'
goto cont

.v3 if (%i66.ne.3) goto v4
%t61='q6gdbx_3'
goto cont

.v4 if (%i66.ne.4) goto v5
%t61='q6gql_1'
goto cont

.v5 if (%i66.ne.5) goto v6
%t61='q6gql_2'
goto cont

.v6 %t61='q6gql_3'

.cont *
```

```
**********************************************
* Start Query 6
**********************************************

KOM ' '
KOM 'Query 6 ('//%t61//') - Find all polygons that intersects a rectangle'
dataout

*
* clear old elementset
*

KOM '-- delete old data'; dataout
gbresem elementset

*
* create cursor for query 5
*

KOM '-- declare a cursor'; dataout
sql begin work
sql declare c_q6 cursor for select x1, y1, x2, y2 from random where type='r'
sql open c_q6

KOM '-- go into loop for benchmarking...'; dataout

if(%i60.le.1) goto noloop

DO %i69 1 %i60

  sql fetch c_q6 into %i61, %i62, %i63, %i64

*
* Convert all integers into strings
*

  set %t1 cvt %i61 *
  set %t2 cvt %i62 *
  set %t3 cvt %i63 *
  set %t4 cvt %i64 *

  KOM '---- rectangle: '//%t1//' '//%t2//' '//%t3//' '//%t4; dataout

  if (%i66.ge.4) goto gql

******************
* GDB-X
******************

  gbaudit dpl=off rdb=on

  gbsrt fln 'X>'//%t1//',X<'//%t3//',Y>'//%t2//',Y<'//%t4 0 0 1100000 1400000

  gbaudit dpl=off rdb=off
  goto nogql

******************


.gql *
```

```
*****************
* GQL
*****************

  gbaudit dpl=off rdb=on

  sql insert into elementset2\
    select location from poly\
      where intersect(rectangle(%i61 %i62 %i63 %i64), poly.location)

gbaudit dpl=off rdb=off

*****************
.nogql *

* save actual performance-values-identification into table perf

  sql select max(gbcmdnumber) into %R99 from gbaudit
  sql insert into perf(qname,cmdnumber) values (%t61,%R99)

ENDDO

goto weiter


**************************
* On error write this
**************************

.noloop KOM '---- NO LOOP!'; dataout
.weiter *


***********************
* save log data
***********************

KOM '-- write result in '//%t61//'.001!'
gbgqsa 1 elementset2 0 fil=(%t61) 0 0 0 0

KOM '-- closing cursor and commit work'; dataout
sql close c_q6
sql commit work

KOM ' '
KOM 'Transfering performance-data into bwerte-table'; dataout

sql insert into bwerte (qname,\
                    cmdnumber, cmdname,\
                    elapsed,cputime,\
                    elembr,elembw,\
                    elemlr,elemlw,\
                    cellbr,cellbw,\
                    celllr,celllw,\
                    bytebr,bytebw\
                   )\
              select p.qname,\
                    gbcmdnumber,gbcmdname,\
                    gbelapsed,gbcputime,\
                    gbelembr,gbelembw,\
                    gbelemlr,gbelemlw,\
                    gbcellbr,gbcellbw,\
```

```
                gbcelllr,gbcelllw,\
                gbbytebr,gbbytebw\
  from perf p, gbaudit where gbcmdnumber=p.cmdnumber

*sql whenever sqlerror stop

*
*

ENDP
```

ad 7. Kombination räumlicher Kriterien: Suche alle Polygone, innerhalb des Kreises k und einer Fläche größer x

```
PROCN

DOCBEG
KOM '*****************************************************************'
KOM 'Query 7 - Find all polygons that intersect a spec. rectangle'
KOM '*****************************************************************'
DOCEND

PARBEG

IN NOQR %i70  PTX='Number of query repititons'
IN VER %i76 PTX='Version of query (gql:4=plain data,5=ld+td,6=td+ld+atw)'

PAREND

********************
* make clean
********************

*sql whenever sqlerror continue
sql delete from gbaudit
sql delete from perf

sql delete from elementset2

*****************************
* find out the version
*****************************

if (%i76.ne.4) goto v5
%t71='q7gql_4'
goto cont

.v5 if (%i76.ne.5) goto v6
%t71='q7gql_5'
goto cont

.v6 %t71='q7gql_6'
.cont *



*********************************************
* Start Query 7
*********************************************
```

```
KOM ' '
KOM 'Query 7 ('//%t71//') - Find all polygons that are more than a specific size and within a specific circle'
dataout

*
* clear old elementset
*

KOM '-- delete old data'; dataout
gbresem elementset

*
* create cursor query 7
*

KOM '-- declare a cursor'; dataout
sql begin work
sql declare c_q7 cursor for select x1, y1 from random where type='k'
sql open c_q7

KOM '-- go into loop for benchmarking...'; dataout

if(%i70.le.1) goto noloop

DO %i79 1 %i70

*
* create a circle
*

* %i71/%i72 = x/y-coordinate of circles center (get from cursor)
* %i73/%i74 = x/y-coordinate of circles margin (computed)
* size of polygons to find are bigger than 1 square km = 1000000 square m
* radius is set to 10000 m

  sql fetch c_q7 into %i71, %i72

  %i73 = %i71+10000
  %i74 = %i72

  set %t1 cvt %i71 *
  set %t2 cvt %i72 *
  set %t3 cvt %i73 *
  set %t4 cvt %i74 *

  KOM 'rectangle: x='//%t1//' y= '//%t2//' rx='//%t3//' ry='//%t4; dataout


******************
* retrieve data
******************

  gbaudit dpl=off rdb=on

  sql insert into elementset2\
     select location from poly\
        where contains(circle(%i71 %i72 %i73 %i74), poly.location)\
          and area(poly.location)>1000000

  gbaudit dpl=off rdb=off

******************
```

```
*
* save actual performance-values-identification into table perf
*

  sql select max(gbcmdnumber) into %R99 from gbaudit
  sql insert into perf(qname,cmdnumber) values (%t71,%R99)

ENDDO

goto weiter


**************************
* On error write this
**************************

.noloop KOM '---- NO LOOP!'; dataout
.weiter *


***********************
* save log data
***********************

KOM '-- write result in '//%t71//'.001!'
gbgqsa 1 elementset2 0 fil=(%t71) 0 0 0 0

KOM '-- closing cursor and commit work'; dataout
sql close c_q7
sql commit work

KOM ' '
KOM 'Transfering performance-data into bwerte-table'; dataout

sql insert into bwerte (qname,\
                        cmdnumber, cmdname,\
                        elapsed,cputime,\
                        elembr,elembw,\
                        elemlr,elemlw,\
                        cellbr,cellbw,\
                        celllr,celllw,\
                        bytebr,bytebw\
                       )\
              select p.qname,\
                        gbcmdnumber,gbcmdname,\
                        gbelapsed,gbcputime,\
                        gbelembr,gbelembw,\
                        gbelemlr,gbelemlw,\
                        gbcellbr,gbcellbw,\
                        gbcelllr,gbcelllw,\
                        gbbytebr,gbbytebw\
   from perf p, gbaudit where gbcmdnumber=p.cmdnumber

*sql whenever sqlerror stop

*
*

ENDP
```

ad 8. Verbund zwischen Punkt- und Polygondaten: Gib die Landnutzung (landuse) aller Polygone innerhalb eines Rechtecks von 50 km um einen Punkt p aus

```
PROCN

DOCBEG
KOM '*************************************************************'
KOM 'Query 8 - Show landuse in a 50 km quadrangle surrounding a given point'
KOM '*************************************************************'
DOCEND

PARBEG

IN NOQR %i80  PTX='No. of query repititions'
IN VER %i86 PTX='Version of query (gdbx:1=plain data,2=ld+td,3=td+ld+atw; gql:4,5,6)'

PAREND


*******************
* make clean
*******************

*sql whenever sqlerror continue
sql delete from gbaudit
sql delete from perf


*****************************
* find out the version
*****************************

if (%i86.ne.1) goto v2
%t81='q8gdbx_1'
goto cont

.v2 if (%i86.ne.2) goto v3
%t81='q8gdbx_2'
goto cont

.v3 if (%i86.ne.3) goto v4
%t81='q8gdbx_3'
goto cont

.v4 if (%i86.ne.4) goto v5
%t81='q8gql_1'
goto cont

.v5 if (%i86.ne.5) goto v6
%t81='q8gql_2'
goto cont

.v6 %t81='q8gql_3'

.cont *



*********************************************
* Start Query 8
*********************************************
```

```
KOM ' '
KOM 'Query 8 ('//%t81//') - Show landuse in a 50 km quadrangle surrounding a given point'
dataout

*
* clear old elementset
*

KOM '-- delete old data'; dataout
gbresem elementset


*
* create cursor for query 8
*

KOM '-- declare a cursor'; dataout
sql begin work
sql declare c_q8 cursor for select name, x1, y1 from random where type='p'
sql open c_q8

KOM '-- create temporary table q8temp'; dataout
sql create table q8temp (landuse char(5), location char(16))

KOM '-- go into loop for benchmarking...'; dataout

if(%i80.le.1) goto noloop


DO %i89 1 %i80


**********************************
* read pointname and coordinates
**********************************
*
* %t82      = pointname
* %i81/%i82 = x/y-coordinate of point
*
**********************************

  gbaudit dpl=off rdb=on

  sql fetch c_q8 into %t82, %i81, %i82

  gbaudit dpl=off rdb=off


*
* save actual performance-values-identification into table perf
*

  sql select max(gbcmdnumber) into %R99 from gbaudit
  sql insert into perf(qname,cmdnumber) values (%t81,%R99)


**********************************
* create 50km x 50km rectangle
**********************************
*
* %i87/%i88  = X/Y-coordinate left/bottom
```

```
* %i83/%i84  = X/Y-coordinate right/top
*
**********************************

 %i87=%i81-25000
 %i88=%i82-25000

 set %t1 cvt %i87 *
 set %t2 cvt %i88 *

 %i83=%i81+25000
 %i84=%i82+25000

 set %t3 cvt %i83 *
 set %t4 cvt %i84 *

 KOM 'rectangle: '//%t1//' '//%t2//' '//%t3//' '//%t4; dataout
 KOM 'pointname: '//%t82; dataout

 if (%i86.ge.4) goto gql


*********************************************
* GDB-X
*********************************************
*
* search for the polygons in the rectangle
*
*********************************************

 gbaudit dpl=off rdb=on

 gbsrt fl 'X>'//%t1//',X<'//%t3//',Y>'//%t2//',Y<'//%t4 0 0 1100000 1400000

 gbaudit dpl=off rdb=off


*
* save actual performance-values-identification into table perf
*

 sql select max(gbcmdnumber) into %R99 from gbaudit
 sql insert into perf(qname,cmdnumber) values (%t81,%R99)


*********************************************
* search for landuse and location of the found polygons
*********************************************

 gbaudit dpl=off rdb=on

 sql insert into q8temp\
   select landuse, location from poly\
     where location in\
       (select location from elementset)

 gbaudit dpl=off rdb=off

 goto nogql

*********************************************
```

```
.gql *


*********************************************
* GQL
*********************************************
*
* search for the polygons in the rectangle
*
*********************************************

  gbaudit dpl=off rdb=on

  sql insert into q8temp\
    select landuse, location from poly\
      where contains(rectangle(%i87 %i88 %i83 %i84), poly.location)

  gbaudit dpl=off rdb=off

*********************************************
.nogql *


*
* save actual performance-values-identification into table perf
*

  sql select max(gbcmdnumber) into %R99 from gbaudit
  sql insert into perf(qname,cmdnumber) values (%t81,%R99)

ENDDO

goto weiter


**************************
* On error write this
**************************

.noloop KOM '---- NO LOOP!'; dataout
.weiter *


*******************************
* save log data and make clean
*******************************

KOM '-- write found elements into '//%t81//'.001!'
gbgqsa 1 q8temp 0 fil=(%t81) 0 0 0 0

KOM '-- drop temporary table q8temp'; dataout
sql drop table q8temp

KOM '-- closing cursor and commit work'; dataout
sql close c_q8
sql commit work

KOM 'Transfering performance-data into bwerte-table'; dataout

sql insert into bwerte (qname,\
                        cmdnumber, cmdname,\
```

```
                elapsed,cputime,\
                elembr,elembw,\
                elemlr,elemlw,\
                cellbr,cellbw,\
                celllr,celllw,\
                bytebr,bytebw\
               )\
          select p.qname,\
                gbcmdnumber,gbcmdname,\
                gbelapsed,gbcputime,\
                gbelembr,gbelembw,\
                gbelemlr,gbelemlw,\
                gbcellbr,gbcellbw,\
                gbcelllr,gbcelllw,\
                gbbytebr,gbbytebw\
  from perf p, gbaudit where gbcmdnumber=p.cmdnumber

*
*

ENDP
```

ad 10. Verbund zwischen Punkt- und Polygondaten (mit Erzeugen einer Tabelle): Ermittle die Namen aller Punkte, die innerhalb von Polygonen mit Landnutzung (landuse) liegen und lege sie in einer Tabelle ab

```
PROCN

* Works only with GQL

DOCBEG
KOM '*************************************************************************'
KOM 'Query 10 - Find the names of all points within polygons of spec. landuse'
KOM '*************************************************************************'
DOCEND

PARBEG

IN NOQR %i20  PTX='No. of query repititions'
IN VER %i26 PTX='Version of query (gql:4=plain data,5=ld+td,6=td+ld+atw)'

PAREND

********************
* make clean
********************

*sql whenever sqlerror continue
sql delete from gbaudit
sql delete from perf


*****************************
* find out the version
*****************************

if (%i26.ne.4) goto v5
%t21='q10gql_4'
goto cont
```

```
.v5 if (%i26.ne.5) goto v6
%t21='q10gql_5'
goto cont

.v6 %t21='q10gql_6'
.cont *


*********************************************
* Start Query 10
*********************************************

KOM ' '
KOM 'Query 10 ('//%t21//') - Find the names of all points within polygons of spec. landuse'
dataout


*
* clear old elementset
*

KOM '-- delete old data'; dataout
gbresem elementset


*
* create cursor for query 5 and make temporary tables
*

KOM '-- create temporary table q10temp'; dataout
sql create table q10temp (name char(70))

KOM '-- declare a cursor'; dataout
sql begin work
sql declare c_q10 cursor for select name from random where type='l'
sql open c_q10

KOM '-- go into loop for benchmarking...'; dataout

if(%i20.le.1) goto noloop



DO %i29 1 %i20

*
* get a landuse to search for into the cursor
*

  sql fetch c_q10 into %t22
  KOM '---- landuse= '//%t22


*******************
* retrieve data
*******************

  gbaudit dpl=off rdb=on

  sql insert into q10temp\
     select name from point, poly\
        where contains(poly.location,point.location)\
```

```
        and poly.landuse=%t22

  gbaudit dpl=off rdb=off

*******************


*
* save actual performance-values-identification into table perf
*

  sql select max(gbcmdnumber) into %R99 from gbaudit
  sql insert into perf(qname,cmdnumber) values (%t21,%R99)

ENDDO

goto weiter


**************************
* On error write this
**************************

.noloop KOM '---- NO LOOP!'; dataout
.weiter *


*********************************
* save log data and make clean
*********************************

KOM '-- write found elements into '//%t21//'.001!'
msgsup; gbgqsa 1 q10temp 0 fil=(%t21) 0 0 0 0

KOM '-- drop temporary table q10temp'; dataout
sql drop table q10temp

KOM '-- closing cursor and commit work'; dataout
sql close c_q10
sql commit work

KOM 'Transfering performance-data into bwerte-table'; dataout

sql insert into bwerte (qname,\
                    cmdnumber, cmdname,\
                    elapsed,cputime,\
                    elembr,elembw,\
                    elemlr,elemlw,\
                    cellbr,cellbw,\
                    celllr,celllw,\
                    bytebr,bytebw\
                   )\
              select p.qname,\
                    gbcmdnumber,gbcmdname,\
                    gbelapsed,gbcputime,\
                    gbelembr,gbelembw,\
                    gbelemlr,gbelemlw,\
                    gbcellbr,gbcellbw,\
                    gbcelllr,gbcelllw,\
                    gbbytebr,gbbytebw\
   from perf p, gbaudit where gbcmdnumber=p.cmdnumber
```

*
*

ENDP

A.3SQL-Skripts des S2K-Benchmarks mit Oracle Spatial

ad 1. Erzeugen der Datenbank, Laden der Daten, Erzeugen der Indizes

Datenbankschema

```
CREATE TABLE S2K.LOC
(
  G_ID NUMBER NOT NULL UNIQUE,
  NAM VARCHAR(255),
  GEOMETRY MDSYS.SDO_GEOMETRY
);

CREATE TABLE S2K.USE (
  G_ID NUMBER NOT NULL UNIQUE,
  FKT INT,
  GEOMETRY MDSYS.SDO_GEOMETRY
);
```

Tabellen für Zwischenergebnisse

```
CREATE TABLE S2K.LOC_RESULT (
  G_ID NUMBER,
);

CREATE TABLE S2K.USE_RESULT (
  G_ID NUMBER,
);
```

AWK Skripts zum Umsetzen der Daten

```
BEGIN                   {
                        FS = „:“;
                        id=0;
                        gtype=1;
                        srid=0;
                        printf(„LOAD DATA\n“ \
                        „INFILE *\n“ \
                        „APPEND\n“ \
                        „INTO TABLE LOC\n“ \
                        „FIELDS TERMINATED BY ,,‘ OPTIONALLY ENCLOSED BY ,\“, \n“ \
                        „TRAILING NULLCOLS\n“ \
                        „(\n“ \
                        „ G_ID SEQUENCE (COUNT, 1),\n“ \
                        „ NAM CHAR(255),\n“ \
                        „ GEOMETRY COLUMN OBJECT\n“ \
                        „ (\n“ \
                        „  SDO_GTYPE INTEGER EXTERNAL,\n“ \
                        „  SDO_ELEM_INFO VARRAY TERMINATED BY ,,/‘ „ \
                        „(X FLOAT EXTERNAL),\n“ \
                        „  SDO_ORDINATES VARRAY TERMINATED BY ,,/‘ „ \
                        „(X FLOAT EXTERNAL)\n“ \
                        „ )\n“ \
                        „)\n“ \
                        „BEGINDATA\n“ \
                        );
                        }
                        {
                        id++;
```

```
                printf( „\“%s\“,  %d,  1,1,1,/  „,
                $3, gtype );
                printf(„ %s,%s,“, $1, $2 );
                printf(„/\n“);
                }

BEGIN           {
                FS = „ „;
                ordcount=0;
                id=0;
                gtype=3;
                srid=0;
                printf(„LOAD DATA\n“ \
                „INFILE *\n“ \
                „APPEND\n“ \
                „INTO TABLE USE\n“ \
                „FIELDS TERMINATED BY ,,‘ OPTIONALLY ENCLOSED BY ,\“, \n“ \
                „TRAILING NULLCOLS\n“ \
                „(\n“ \
                „ G_ID SEQUENCE (COUNT, 1),\n“ \
                „ FKT INTEGER EXTERNAL,\n“ \
                „ GEOMETRY COLUMN OBJECT\n“ \
                „ (\n“ \
                „  SDO_GTYPE INTEGER EXTERNAL,\n“ \
                „  SDO_ELEM_INFO VARRAY TERMINATED BY ,,/‘ „ \
                „(X FLOAT EXTERNAL),\n“ \
                „  SDO_ORDINATES VARRAY TERMINATED BY ,,/‘ „ \
                „(X FLOAT EXTERNAL)\n“ \
                „ )\n“ \
                „)\n“ \
                „BEGINDATA\n“ \
                );
                }
/[-0-9 ]+/      {
                # print „Daten:“, FILENAME, NR, $1, $2;
                ordinates[ordcount++] = $1;
                ordinates[ordcount++] = $2;
                }
/^$/            {
                # print „Leer:“, NR, $0;
                # the first two ordinates are a reference point
                id++;
                printf( „%d,  %d,  1,3,1,/  „,
                FILENAME+0, gtype );
                # the 3rd to nth ordinates are pairs of coordinates
                # of the polygon
                for (i=2; i<ordcount; i+=2 )
                {
                if ((i >= 4) &&
                (ordinates[i-2] == ordinates[i]) &&
                (ordinates[i-2+1] == ordinates[i+1]))
                {
                continue;
                }
                printf(„ %s,%s,“, ordinates[i], ordinates[i+1] );
                }
                printf(„/\n“);

                ordcount = 0;
                }
```

Kommandos zum Laden der Daten

```
date

for file in $*
# „point“
do
        echo loading $file.bulk
        sqlldr s2k/s2k@omz1 log=bulk/$file.log control=bulk/$file.bulk
done

date


date

for file in $*
# „0 11 12 13 14 15 16 17 21 22 23 24 31 32 33 41 42 43 51 52 53 54 61 62 71 72 73 74 75 76 77 81 82 83 85 91 92 islands“
do
        echo loading $file.bulk
        sqlldr s2k/s2k@omz1 log=bulk/$file.log control=bulk/$file.bulk
done

date
```

Kommandos zum Erzeugen der Tuningempfehlungen für den Index

```
SET SERVEROUTPUT ON;
DECLARE
        IDX_LEV INTEGER;
        G_TABLE VARCHAR2(64) DEFAULT ‚LOC‘;
        G_COLUMN VARCHAR2(64) DEFAULT ‚GEOMETRY‘;
        IDX_PARAM VARCHAR2(64);
BEGIN
        IDX_LEV := MDSYS.SDO_TUNE.ESTIMATE_TILING_LEVEL(G_TABLE, G_COLUMN,
                10000, ‚ALL_GID_EXTENT‘);
        IDX_PARAM := ‚SDO_LEVEL=‘ || IDX_LEV || ‚, SDO_NUMTILES=‘ || (IDX_LEV+2);
        DBMS_OUTPUT.PUT_LINE(‚DROP INDEX ‚ || G_TABLE || ‚_SPT_IDX;‘);
        DBMS_OUTPUT.PUT_LINE(‚CREATE INDEX ‚ || G_TABLE || ‚_SPT_IDX ‚);
        DBMS_OUTPUT.PUT_LINE(‚ON ‚ || G_TABLE || ‚(‚ || G_COLUMN || ‚) ‚);
        DBMS_OUTPUT.PUT_LINE(‚INDEXTYPE IS MDSYS.SPATIAL_INDEX ‚);
        DBMS_OUTPUT.PUT_LINE(‚PARAMETERS(‘‘ || IDX_PARAM || ‚‘‘);‘);
END;
/
DECLARE
        IDX_LEV INTEGER;
        G_TABLE VARCHAR2(64) DEFAULT ‚USE‘;
        G_COLUMN VARCHAR2(64) DEFAULT ‚GEOMETRY‘;
        IDX_PARAM VARCHAR2(64);
BEGIN
        IDX_LEV := MDSYS.SDO_TUNE.ESTIMATE_TILING_LEVEL(G_TABLE, G_COLUMN,
                10000, ‚ALL_GID_EXTENT‘);
        IDX_PARAM := ‚SDO_LEVEL=‘ || IDX_LEV || ‚, SDO_NUMTILES=‘ || (IDX_LEV+2);
        DBMS_OUTPUT.PUT_LINE(‚DROP INDEX ‚ || G_TABLE || ‚_SPT_IDX;‘);
        DBMS_OUTPUT.PUT_LINE(‚CREATE INDEX ‚ || G_TABLE || ‚_SPT_IDX ‚);
        DBMS_OUTPUT.PUT_LINE(‚ON ‚ || G_TABLE || ‚(‚ || G_COLUMN || ‚) ‚);
        DBMS_OUTPUT.PUT_LINE(‚INDEXTYPE IS MDSYS.SPATIAL_INDEX ‚);
        DBMS_OUTPUT.PUT_LINE(‚PARAMETERS(‘‘ || IDX_PARAM || ‚‘‘);‘);
END;
/
```

Kommandos zum Erzeugen der Indizes

```
CREATE INDEX LOC_SPT_IDX ON LOC(GEOMETRY)
INDEXTYPE IS MDSYS.SPATIAL_INDEX
PARAMETERS(,SDO_LEVEL=6, SDO_NUMTILES=8‘);

CREATE INDEX USE_SPT_IDX ON USE(GEOMETRY)
INDEXTYPE IS MDSYS.SPATIAL_INDEX
PARAMETERS(,SDO_LEVEL=6, SDO_NUMTILES=8‘);
```

ad 5. nicht-räumliche Selektion: Suche einen Punkt mit einem Namen n

ad 6. räumliche Selektion (mit Erzeugung einer neuen Tabelle): Suche alle Polygone, die sich mit einem Rechteck r überschneiden.

ad 7. Kombination räumlicher Kriterien: Suche alle Polygone, innerhalb des Kreises k und einer Fläche größer x

ad 8. Verbund zwischen Punkt- und Polygondaten: Gib die Landnutzung (landuse) aller Polygone innerhalb eines Rechtecks von 50 km um einen Punkt p aus

ad 10. Verbund zwischen Punkt- und Polygondaten (mit Erzeugen einer Tabelle): Ermittle die Namen aller Punkte, die innerhalb von Polygonen mit Landnutzung (landuse) liegen und lege sie in einer Tabelle ab

A.4GIS-Datentypen am Beispiel von SICAD

Gruppe	Basis-GIS Fach-GIS	SICAD Element	Beschreibung
Punkt	Basis	PK	einfacher Punkt
		TX	Text
		SY	Symbol
	Fach	FR	Flurstücksnummer
		PG	Vermessungspunkt
		OB	Objekt
		...	Weitere Typen sind GE, MB, PA, PI, PP, PR, QD, QS, TA, TB, TP, TS
Linie	Basis	LI	gerade Linie
		LY	Linienzug mit geraden Verbindungen
		BO	Bogen
		SN	Spline
	Fach	KB	Fachdatentypen für Kanal
		KH	
		KS	
		L0	Leitung
		LT	
		RO	Rohr
Fläche	Basis	FL	Fläche
		KR	Kreis

Tab. 36: Beschreibung der Elemente von SICAD und ihre Zuordnung zu den drei grundlegenden Typen Punkt, Linie und Fläche

Elementtypentabelle	
Elementtypen	Gruppe
FL	Fläche
KR	Fläche
BO	Linie
KB	Linie
KH	Linie
KS	Linie
LI	Linie
LO	Linie
LT	Linie
LY	Linie
RO	Linie
SN	Linie
FR	Punkt
GE	Punkt
MB	Punkt
OB	Punkt
PA	Punkt
PG	Punkt
PI	Punkt
PK	Punkt
PP	Punkt
PR	Punkt
QD	Punkt
QS	Punkt
SY	Punkt
TA	Punkt
TB	Punkt
TF	Punkt
TP	Punkt
TS	Punkt
TX	Punkt

Tab. 37: Die Tabelle der Elementtypen ordnet alle von GQL unterstützten Elementtypen einer der Gruppen Fläche, Linie oder Punkt zu.

Prioritätstabelle			
Operator		Anzahl konstante Parameter	Priorität
ID	Name		
1	DISTANCE	0	30
1	DISTANCE	1	10
1	DISTANCE	2	1
2	LEN	0	20
2	LEN	1	1
2	LEN	2	NULL
3	AREA	0	20
3	AREA	1	1
3	AREA	2	NULL
4	PERIMETER	0	20
4	PERIMETER	1	1
4	PERIMETER	2	NULL
5	VALUE_OF	0	20
5	VALUE_OF	1	1
5	VALUE_OF	2	NULL
6	OVERLAPS	0	8
6	OVERLAPS	1	3
6	OVERLAPS	2	1
7	TOUCH	0	8
7	TOUCH	1	3
7	TOUCH	2	1
8	INTERSECT	0	8
8	INTERSECT	1	3
8	INTERSECT	2	1
9	CROSS	0	8
9	CROSS	1	3
9	CROSS	2	1
10	CONTAINS	0	7
10	CONTAINS	1	2
10	CONTAINS	2	1
11	WITHIN	0	8
11	WITHIN	1	3
11	WITHIN	2	1
12	EQUALS	0	7
12	EQUALS	1	2
12	EQUALS	2	1
13	DISJOINT	0	8
13	DISJOINT	1	2
13	DISJOINT	2	1

Tab. 38: Die Tabelle der Prioritäten steuert die Ausführungsreihenfolge der Operatoren.

A.5Datenbankschema des Basismodells und des Applikationsmodells

Basismodell:

Metadatentabellen:

```
CREATE TABLE SICAD.SICAD_GEOOBJECT_TYPES (
   GEOOBJECT_TYPE NUMBER NOT NULL UNIQUE,
   DESCRIPTION VARCHAR2(256) NOT NULL,
   CONSTRAINT SIC_GEOO_TYP_IDX PRIMARY KEY (GEOOBJECT_TYPE));

CREATE TABLE SICAD.SICAD_GEOOBJECT_TABLES (
   G_TABLE_SCHEMA VARCHAR2(64),
   G_TABLE_NAME VARCHAR2(64),
   G_GEOMETRY_COLUMN VARCHAR2(64),
   GEOOBJECT_TYPE NUMBER,
   DESCRIPTIVE_NAME VARCHAR2(256),
   CONSTRAINT SIC_GEOO_TAB_IDX PRIMARY KEY
         (G_TABLE_SCHEMA,G_TABLE_NAME,G_GEOMETRY_COLUMN),
   CONSTRAINT SIC_GEOO_TAB_TYP FOREIGN KEY(GEOOBJECT_TYPE)
         REFERENCES SICAD.SICAD_GEOOBJECT_TYPES(GEOOBJECT_TYPE),
   CONSTRAINT SIC_GEOO_TAB_GEO_COL
         FOREIGN KEY(G_TABLE_SCHEMA,G_TABLE_NAME,G_GEOMETRY_COLUMN)
         REFERENCES MDSYS.OGIS_GEOMETRY_COLUMNS
                  (G_TABLE_SCHEMA,G_TABLE_NAME,G_GEOMETRY_COLUMN));

CREATE TABLE SICAD.SICAD_GEOOBJECT_PRESENTATIONS (
   SGP_ID NUMBER NOT NULL UNIQUE,
   G_TABLE_SCHEMA VARCHAR2(64),
   G_TABLE_NAME VARCHAR2(64),
   G_GEOMETRY_COLUMN VARCHAR2(64),
   STYLE_TABLE VARCHAR2(64) NOT NULL,
   GEOOBJECT_STYLE_INTER_TABLE VARCHAR2(64),
   DESCRIPTIVE_NAME VARCHAR2(256),
   DEFAULT_STYLE REF SICAD.SICAD_STYLE_T,
   CONSTRAINT SIC_GEOO_PRE_IDX PRIMARY KEY (SGP_ID),
   CONSTRAINT SIC_PRE_GEOO_TAB
         FOREIGN KEY(G_GEOMETRY_COLUMN,G_TABLE_NAME,G_TABLE_SCHEMA)
         REFERENCES SICAD.SICAD_GEOOBJECT_TABLES
                  (G_GEOMETRY_COLUMN,G_TABLE_NAME,G_TABLE_SCHEMA));
```

Definition abstrakter Datentypen für Styles, Style-Tabellen und Verknüpfungstabellen:

```
CREATE OR REPLACE TYPE SICAD.SICAD_GEO_STYLE_T AS OBJECT (
   POINTSTYLE REF SICAD.SICAD_POINT_STYLE_T,
   LINESTYLE REF SICAD.SICAD_LINE_STYLE_T,
   AREASTYLE REF SICAD.SICAD_AREA_STYLE_T,
   TEXTSTYLE REF SICAD.SICAD_TEXT_STYLE_T,
   SYMBOLSTYLE REF SICAD.SICAD_SYMBOL_STYLE_T);
```

Kommentar: Die Detaildefinitionen der Objekte POINTSTYLE, LINESTYLE, AREASTYLE, TEXTSTYLE und SYMBOLSTYLE sind hier der Übersicht halber nicht aufgeführt. Sie werden zum Verständnis der grundlegenden Elemente des Basismodells nicht benötigt.

```
CREATE OR REPLACE TYPE SICAD.SICAD_STYLE_T AS OBJECT (
   STYLE_ID NUMBER,
   STYLE_NAME VARCHAR2(256),
   DESCRIPTION VARCHAR2(256),
```

```
    GEOSTYLE SICAD.SICAD_GEO_STYLE_T);

CREATE OR REPLACE TYPE SICAD.SICAD_GEOOBJECT_PRESENTATION_T AS OBJECT (
    SGP_ID NUMBER,
    G_ID NUMBER,
    STYLE REF SICAD.SICAD_STYLE_T,
    SEQUENCE_NUMBER NUMBER,
    PRESENTATION_GEOMETRIES MDSYS.SDO_GEOMETRY);
```

Applikationsmodell:

Erzeugen von Geo-Objekt-, Style- und Verknüpfungstabelle für Geo-Objekt Gebäude:

```
CREATE TABLE ALKIS.AL_GEBAEUDE
(
  G_ID NUMBER NOT NULL UNIQUE,
  OBJEKTID CHAR(7),
  OBJEKTARTENKENNUNG CHAR(3),
  MODELLARTENKENNUNG CHAR(8),
  ENTSTEHUNGSDATUM CHAR(6),
  VERSION CHAR(2),
  GEBAEUDEFUNKTION INT,
  NAME VARCHAR(255),
  GESCHOSSE INT,
  GEOMETRY MDSYS.SDO_GEOMETRY
);

CREATE TABLE ALKIS.AL_GEBAEUDE_STYLES OF SICAD.SICAD_STYLE_T;
CREATE TABLE ALKIS.AL_GEBAEUDE_PRESENTATIONS OF SICAD_GEOOBJECT_PRESENTATION_T;
```

Automatische Verknüpfung jedes Geo-Objekts Gebäude mit dem Default-Style:

```
CREATE TRIGGER ALKIS.AL_GEBAEUDE_INS_TRG AFTER INSERT ON ALKIS.AL_GEBAEUDE
        REFERENCING NEW AS NEW
        FOR EACH ROW
        DECLARE
                SGP_ID NUMBER;
                STYLE REF SICAD.SICAD_STYLE_T;
        BEGIN
                SELECT SGP_ID INTO SGP_ID FROM SI-
CAD.SICAD_GEOOBJECT_PRESENTATIONS
                WHERE DESCRIPTIVE_NAME = 'Gebaeude DEFAULT';
                SELECT REF(S) INTO STYLE FROM ALKIS.AL_GEBAEUDE_STYLES S
                WHERE STYLE_NAME = 'DEFAULT';
                INSERT INTO ALKIS.AL_GEBAEUDE_PRESENTATIONS VALUES
                (
                SGP_ID,
                :NEW.G_ID,
                STYLE,
                1,
                NULL
                );
        END;
```

Eintragen von Metadaten für Geo-Objekt Gebäude:

```
DECLARE
        G_TYPE VARCHAR2(64) DEFAULT 'GEBAEUDE_T';
```

```
        G_SCHEMA VARCHAR2(64) DEFAULT USER;
        G_TABLE VARCHAR2(64) DEFAULT 'AL_GEBAEUDE';
        G_COLUMN VARCHAR2(64) DEFAULT 'GEOMETRY';
        G_STYLE_TABLE VARCHAR2(64) DEFAULT 'AL_GEBAEUDE_STYLES';
        G_PRESENTATION_TABLE VARCHAR2(64) DEFAULT 'AL_GEBAEUDE_PRESENTATIONS';
        G_ID NUMBER;
        S_ID NUMBER;
        SPL_ID NUMBER;
        SGP_ID NUMBER;
        STYLE REF SICAD.SICAD_STYLE_T;
BEGIN
 INSERT INTO MDSYS.OGIS_GEOMETRY_COLUMNS VALUES
 (
  G_SCHEMA,
  G_TABLE,
  G_COLUMN,
  NULL,
  NULL,
  NULL,
  NULL,
  NULL,
  NULL,
  NULL
 );
 INSERT INTO SICAD.SICAD_GEOOBJECT_TYPES VALUES
 (
  SICAD.SICAD_GEOOBJECT_TYPE_IDS.NEXTVAL,
  G_TYPE
 ) RETURNING G_ID INTO G_ID;
 INSERT INTO SICAD.SICAD_GEOOBJECT_TABLES VALUES
 (
  G_SCHEMA,
  G_TABLE,
  G_COLUMN,
        G_ID,
  'Gebaeude'
 );
 INSERT INTO MDSYS.SDO_GEOM_METADATA VALUES
 (
  G_TABLE,
  G_COLUMN,
        MDSYS.SDO_DIM_ARRAY(
                MDSYS.SDO_DIM_ELEMENT('X', 10000, 40000, .005),
                MDSYS.SDO_DIM_ELEMENT('Y', 10000, 40000, .005))
 );
 INSERT INTO ALKIS.AL_GEBAEUDE_STYLES VALUES
 (
  SICAD.SICAD_STYLE_IDS.NEXTVAL,
        'DEFAULT',
        'Default Gebaeude Style',
        SICAD.SICAD_GEO_STYLE_T(NULL, NULL, NULL, NULL, NULL)
 ) RETURNING STYLE_ID INTO S_ID;

 SELECT REF(S) INTO STYLE FROM ALKIS.AL_GEBAEUDE_STYLES S
  WHERE STYLE_NAME = 'DEFAULT';

 INSERT INTO SICAD.SICAD_GEOOBJECT_PRESENTATIONS VALUES
 (
  SICAD.SICAD_GEOOBJECT_PRESENTATION_IDS.NEXTVAL,
  G_SCHEMA,
  G_TABLE,
  G_COLUMN,
```

```
   G_STYLE_TABLE,
   G_PRESENTATION_TABLE,
   'Gebaeude DEFAULT',
   STYLE
  ) RETURNING SGP_ID INTO SGP_ID;
  SELECT SPL_ID INTO SPL_ID FROM SICAD.SICAD_PRESENTATION_LAYERS
   WHERE DESCRIPTIVE_NAME = 'DEFAULT';
  INSERT INTO SICAD.SICAD_PRE_LAY_TO_GEOO_PRE VALUES
  (
   SPL_ID,
            SGP_ID
  );
END;
/
COMMIT WORK;
```

Erzeugen von Geo-Objekt-, Style- und Verknüpfungstabelle für Geo-Objekt Bauteile:

```
CREATE TABLE ALKIS.BAUTEILE
(
 G_ID NUMBER NOT NULL UNIQUE,
 OBJEKTID CHAR(7),
 OBJEKTARTENKENNUNG CHAR(3),
 MODELLARTENKENNUNG CHAR(8),
 ENTSTEHUNGSDATUM CHAR(6),
 VERSION CHAR(2),
 ART INT,
 GEHOERT_ZU CHAR(7),
 GEOMETRY MDSYS.SDO_GEOMETRY
);

CREATE TABLE ALKIS.BAUTEILE_STYLES OF SICAD.SICAD_STYLE_T;
CREATE TABLE ALKIS.BAUTEILE_PRESENTATIONS OF SICAD.SICAD_GEOOBJECT_PRESENTATION_T;

CREATE TRIGGER ALKIS.BAUTEILE_INS_TRG AFTER INSERT ON ALKIS.BAUTEILE
        REFERENCING NEW AS NEW
        FOR EACH ROW
        DECLARE
                SGP_ID NUMBER;
                STYLE REF SICAD.SICAD_STYLE_T;
        BEGIN
                SELECT SGP_ID INTO SGP_ID FROM SI-
CAD.SICAD_GEOOBJECT_PRESENTATIONS
                WHERE DESCRIPTIVE_NAME = 'Bauteile DEFAULT';
                SELECT REF(S) INTO STYLE FROM ALKIS.BAUTEILE_STYLES S
                WHERE STYLE_NAME = 'DEFAULT';
                INSERT INTO ALKIS.BAUTEILE_PRESENTATIONS VALUES
                (
                SGP_ID,
                :NEW.G_ID,
                STYLE,
                1,
                NULL
                );
        END;
/

DECLARE
        G_TYPE VARCHAR2(64) DEFAULT 'BAUTEILE_T';
        G_SCHEMA VARCHAR2(64) DEFAULT USER;
```

```
            G_TABLE VARCHAR2(64) DEFAULT 'BAUTEILE';
            G_COLUMN VARCHAR2(64) DEFAULT 'GEOMETRY';
            G_STYLE_TABLE VARCHAR2(64) DEFAULT 'BAUTEILE_STYLES';
            G_PRESENTATION_TABLE VARCHAR2(64) DEFAULT 'BAUTEILE_PRESENTATIONS';
            G_ID NUMBER;
            S_ID NUMBER;
            SPL_ID NUMBER;
            SGP_ID NUMBER;
        STYLE REF SICAD.SICAD_STYLE_T;
BEGIN
  INSERT INTO MDSYS.OGIS_GEOMETRY_COLUMNS VALUES
  (
    G_SCHEMA,
    G_TABLE,
    G_COLUMN,
    NULL,
    NULL,
    NULL,
    NULL,
    NULL,
    NULL,
    NULL
  );
  INSERT INTO SICAD.SICAD_GEOOBJECT_TYPES VALUES
  (
    SICAD.SICAD_GEOOBJECT_TYPE_IDS.NEXTVAL,
    G_TYPE
  ) RETURNING G_ID INTO G_ID;
  INSERT INTO SICAD.SICAD_GEOOBJECT_TABLES VALUES
  (
    G_SCHEMA,
    G_TABLE,
    G_COLUMN,
            G_ID,
    'Bauteile'
  );
  INSERT INTO MDSYS.SDO_GEOM_METADATA VALUES
  (
    G_TABLE,
    G_COLUMN,
            MDSYS.SDO_DIM_ARRAY(
                        MDSYS.SDO_DIM_ELEMENT('X', 10000, 40000, .005),
                        MDSYS.SDO_DIM_ELEMENT('Y', 10000, 40000, .005))
  );
  INSERT INTO ALKIS.BAUTEILE_STYLES VALUES
  (
    SICAD.SICAD_STYLE_IDS.NEXTVAL,
            'DEFAULT',
            'Default Bauteile Style',
            SICAD.SICAD_GEO_STYLE_T(NULL, NULL, NULL, NULL, NULL)
  ) RETURNING STYLE_ID INTO S_ID;

  SELECT REF(S) INTO STYLE FROM ALKIS.BAUTEILE_STYLES S
    WHERE STYLE_NAME = 'DEFAULT';

  INSERT INTO SICAD.SICAD_GEOOBJECT_PRESENTATIONS VALUES
  (
    SICAD.SICAD_GEOOBJECT_PRESENTATION_IDS.NEXTVAL,
    G_SCHEMA,
    G_TABLE,
    G_COLUMN,
    G_STYLE_TABLE,
```

```
   G_PRESENTATION_TABLE,
   'Bauteile DEFAULT',
   STYLE
  ) RETURNING SGP_ID INTO SGP_ID;
  SELECT SPL_ID INTO SPL_ID FROM SICAD.SICAD_PRESENTATION_LAYERS
   WHERE DESCRIPTIVE_NAME = 'DEFAULT';
  INSERT INTO SICAD.SICAD_PRE_LAY_TO_GEOO_PRE VALUES
  (
   SPL_ID,
            SGP_ID
  );
END;
/
COMMIT WORK;
```

Erzeugen von Geo-Objekt-, Style- und Verknüpfungstabelle für Geo-Objekt Flurstücke:

```
CREATE TABLE ALKIS.FLURSTUECKE
(
 G_ID NUMBER NOT NULL UNIQUE,
 OBJEKTID CHAR(7),
 OBJEKTARTENKENNUNG CHAR(3),
 MODELLARTENKENNUNG CHAR(8),
 ENTSTEHUNGSDATUM CHAR(6),
 VERSION CHAR(2),
 LAND INT,
 GEMARKUNG INT,
 FLUR INT,
 ZAEHLER INT,
 NENNER INT,
 FOLGENUMMER CHAR(2),
 AMTLICHE_FLAECHE REAL,
 OBJEKTKOORDINATE_X INT,
 OBJEKTKOORDINATE_Y INT,
 GEOMETRY MDSYS.SDO_GEOMETRY
);
CREATE TABLE ALKIS.FLURSTUECKE_STYLES OF SICAD.SICAD_STYLE_T;
CREATE TABLE ALKIS.FLURSTUECKE_PRESENTATIONS OF SI-
CAD.SICAD_GEOOBJECT_PRESENTATION_T;
CREATE TRIGGER ALKIS.FLURSTUECKE_INS_TRG AFTER INSERT ON ALKIS.FLURSTUECKE
        REFERENCING NEW AS NEW
        FOR EACH ROW
        DECLARE
                SGP_ID NUMBER;
                STYLE REF SICAD.SICAD_STYLE_T;
        BEGIN
                SELECT SGP_ID INTO SGP_ID FROM SI-
CAD.SICAD_GEOOBJECT_PRESENTATIONS
                WHERE DESCRIPTIVE_NAME = 'Flurstuecke DEFAULT';
                SELECT REF(S) INTO STYLE FROM ALKIS.FLURSTUECKE_STYLES S
                WHERE STYLE_NAME = 'DEFAULT';
                INSERT INTO ALKIS.FLURSTUECKE_PRESENTATIONS VALUES
                (
                SGP_ID,
                :NEW.G_ID,
                STYLE,
                1,
                NULL
                );
        END;
/
```

```
DECLARE
		G_TYPE VARCHAR2(64) DEFAULT 'FLURSTUECKE_T';
		G_SCHEMA VARCHAR2(64) DEFAULT USER;
		G_TABLE VARCHAR2(64) DEFAULT 'FLURSTUECKE';
		G_COLUMN VARCHAR2(64) DEFAULT 'GEOMETRY';
		G_STYLE_TABLE VARCHAR2(64) DEFAULT 'FLURSTUECKE_STYLES';
		G_PRESENTATION_TABLE VARCHAR2(64) DEFAULT 'FLURSTUECKE_PRESENTATIONS';
		G_ID NUMBER;
		S_ID NUMBER;
		SPL_ID NUMBER;
		SGP_ID NUMBER;
		STYLE REF SICAD.SICAD_STYLE_T;
BEGIN
 INSERT INTO MDSYS.OGIS_GEOMETRY_COLUMNS VALUES
 (
  G_SCHEMA,
  G_TABLE,
  G_COLUMN,
  NULL,
  NULL,
  NULL,
  NULL,
  NULL,
  NULL,
  NULL
 );
 INSERT INTO SICAD.SICAD_GEOOBJECT_TYPES VALUES
 (
  SICAD.SICAD_GEOOBJECT_TYPE_IDS.NEXTVAL,
  G_TYPE
 ) RETURNING G_ID INTO G_ID;
 INSERT INTO SICAD.SICAD_GEOOBJECT_TABLES VALUES
 (
  G_SCHEMA,
  G_TABLE,
  G_COLUMN,
		G_ID,
  'Flurstuecke'
 );
 INSERT INTO MDSYS.SDO_GEOM_METADATA VALUES
 (
  G_TABLE,
  G_COLUMN,
		MDSYS.SDO_DIM_ARRAY(
				MDSYS.SDO_DIM_ELEMENT('X', 10000, 40000, .005),
				MDSYS.SDO_DIM_ELEMENT('Y', 10000, 40000, .005))
 );
 INSERT INTO ALKIS.FLURSTUECKE_STYLES VALUES
 (
  SICAD.SICAD_STYLE_IDS.NEXTVAL,
		'DEFAULT',
		'Default Flurstueck Style',
		SICAD.SICAD_GEO_STYLE_T(NULL, NULL, NULL, NULL, NULL)
 ) RETURNING STYLE_ID INTO S_ID;

 SELECT REF(S) INTO STYLE FROM ALKIS.FLURSTUECKE_STYLES S
  WHERE STYLE_NAME = 'DEFAULT';

 INSERT INTO SICAD.SICAD_GEOOBJECT_PRESENTATIONS VALUES
 (
  SICAD.SICAD_GEOOBJECT_PRESENTATION_IDS.NEXTVAL,
```

```
    G_SCHEMA,
    G_TABLE,
    G_COLUMN,
    G_STYLE_TABLE,
    G_PRESENTATION_TABLE,
    'Flurstuecke DEFAULT',
    STYLE
  ) RETURNING SGP_ID INTO SGP_ID;
  SELECT SPL_ID INTO SPL_ID FROM SICAD.SICAD_PRESENTATION_LAYERS
    WHERE DESCRIPTIVE_NAME = 'DEFAULT';
  INSERT INTO SICAD.SICAD_PRE_LAY_TO_GEOO_PRE VALUES
  (
    SPL_ID,
                SGP_ID
  );
END;
/
COMMIT WORK;
```