Technische Universität München
Institut für Geodäsie, GIS und Landmanagement
Fachgebiet Geoinformationssysteme

Relationale und Objektrelationale Datenbankkonzepte in Geoinformationssystemen

Dipl.-Ing. (Univ.) Stefan Johannes Scheugenpflug

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

Doktor-Ingenieurs (Dr.-Ing.)

genehmigten Dissertation.

Vorsitzender:		Univ.-Prof. Dr.-Ing. Holger Magel
Prüfer der Dissertation:	1.	Univ.-Prof. Dr.-Ing. Matthäus Schilcher
	2.	Univ.-Prof. Dr.-Ing. Liqiu Meng

Die Dissertation wurde am 26.04.2005 bei der Technischen Universität München eingereicht und durch die Fakultät für Bauingenieur- und Vermessungswesen am 13.06.2005 angenommen.

Zusammenfassung

Die integrierte Verwaltung von Geodatenbeständen in relationalen Datenbanken bietet gegenüber der klassischen, getrennten Datenhaltung von Geometrie-, Sach- und Metadaten in Dateisystemen eine Reihe von Vorteilen. Dies gilt insbesondere für ein effizienteres Geodatenmanagement, das den gesicherten Mehrbenutzerbetrieb und die Sicherung von Qualitätsaspekten für Geodaten einschließt. Darüber hinaus ergibt sich ein Quantensprung bei der breiten Nutzbarmachung und Fortführung von Geodatenbeständen.

Mit der Erweiterung relationaler Datenbankmanagementsysteme (RDBMS) um objektorientierte Konzepte und der Etablierung der daraus resultierenden objektrelationalen Datenbankmanagementsysteme (ORDBMS) stellt sich die Frage nach dem zusätzlichen Mehrwert aber auch nach den Grenzen ihrer Nutzung für die Geoinformatik einerseits und die Anwenderpraxis andererseits.
Speziell die von den führenden ORDBMS-Herstellern angebotenen Erweiterungen zur Speicherung und Analyse von raumbezogenen Daten bieten in Kombination mit den Produkten der GIS-Anbieter mittlerweile eine stabile Plattform für ein effizientes Geodatenmanagement. Die vollständige Integration von objektrelationalen Datenbanken in Geoinformationssystemen (GIS), d.h. die unmittelbare Nutzung ihrer zusätzlich verfügbaren Modellierungsmittel für die GIS-Anwendungsentwicklung ist bislang jedoch noch nicht ausreichend untersucht.

Die Arbeit diskutiert ausgehend von den konzeptionellen Anforderungen der GIS-Anwender die erzielbaren Mehrwertaspekte sowie die noch bestehenden Defizite beim Einsatz integrierter Datenhaltungskonzepte auf Basis von relationalen Datenbanksystemen (RDBS) und vergleicht sie mit denen objektrelationaler Datenbanksysteme (ORDBS). Der Vergleich der Leistungsfähigkeit der verschiedenen Modellierungs- und Datenhaltungskonzepte für Geometrie-, Sach- und Metadaten erfolgt auf Grundlage empirischer Untersuchungen und ausgewählter GIS-Anwendungen / -Analysen. Die Arbeit beschreibt die bei der Migration eines realen Geodatenbestandes in den jeweiligen Datenhaltungstyp entstehenden Datenbank-Strukturen und entwickelt auf deren Basis Konzepte zur Optimierung der Nutzung und Fortschreibung zentraler Geodatenbasen sowie zur erweiterten Modellierung von Geoobjekten.

In der Arbeit wird herausgearbeitet, für welche GIS-Anwendungen sich der mit der Nutzung höherer Datenhaltungskonzepte einhergehende Migrationsaufwand lohnt und welche Einschränkungen aktuell damit verbunden sind. Basierend auf der Diskrepanz bei der Abbildung konzeptioneller GIS-Anforderungen im logischen bzw. physikalischen Modell der GIS- bzw. Datenbank-Hersteller werden der noch bestehende Forschungsbedarf formuliert und Empfehlungen für die künftige GIS- und Datenbankentwicklung abgeleitet.

Die im Rahmen dieser Arbeit gewonnenen Erkenntnisse basieren auf einer Systemplattform, die auf Oracle 9i inkl. seiner Erweiterung Oracle Spatial sowie der Middleware-Komponente ArcSDE und ArcInfo 8.3 von ESRI aufsetzt. Als Testgebiet dient der „Nationalpark Bayerischer Wald“, der sich aufgrund seiner komplexen raumzeitlichen Prozesse wie etwa der Totholzausbreitung infolge mehrjähriger Borkenkäfermassenvermehrung hervorragend für die Untersuchungsziele eignet. Dies gilt vor allem für den Vergleich des relationalen und objektrelationalen Modellansatzes hinsichtlich des Potentials zur Adaption der aus Anwendersicht bestehenden konzeptionellen Anforderungen im Allgemeinen und der vollständigen Abbildung von Geoobjekten im Speziellen.

Abstract

The management of spatial data in relational databases provides a number of advantages compared to traditional (separated) approaches where geometry and metadata are kept in file systems as opposed to alphanumeric data that reside in relational databases. Above all relational databases ensure efficient spatial data management by providing multi-user support and better control of spatial data quality. In addition relational database approaches lead to significant improvements for wide usage and availability of geographical information and are also better suited for maintenance and update processes.

By extending Relational Database Management Systems (RDBMS) to support object-oriented concepts, reliable object-relational database management systems (ORDBMS) have become available. This development poses two interesting questions. What is the added value and what are the limitations for practical applications of Geographic Information Systems (GIS) and for geographic information science in general? Today special built-in extensions by leading ORDBMS vendors for storing and analysing spatial data can be combined with GIS products and these combinations already provide a reliable platform for efficient spatial data management. However a full integration of ORDBS and GIS which would allow for immediate utilization of additional modelling potential for GIS application development has not yet been thoroughly researched.

Starting from an analysis of the conceptual requirements of GIS users this research investigates the potential for achieving added value through ORDBMS and it examines the remaining deficits of integrated geospatial data management concepts by comparing RDBS based approaches with ORDBS based strategies. Empirical research and selected GIS applications / analysis are exploited to evaluate and compare the capabilities of different data modelling and data management concepts to handle geometry data, alphanumerical data and metadata. This research describes the database structures resulting from the migration of real data into particular data (management) types. Concepts to optimize usage, maintenance and update processes of central spatial databases are developed and it is pointed out how geospatial objects can be entirely modelled to achieve complete integration.

This research classifies GIS applications where the migration to more sophisticated data management concepts prevails over the costs being involved. Limitations which are presently remaining are identified. Based on the discrepancy between conceptual GIS requirements and their mappings into logical and physical database models of GIS and database vendors, the need for further research is outlined. In addition recommendations for future GIS and database development are given.

The system platform used for this research is based on Oracle 9i, including the Oracle Spatial extension, ESRI's middleware component ArcSDE and ArcInfo 8.3. The Bavarian Forest National Park was selected as the test area mainly because complex spatiotemporal processes such as the spread and distribution of deadwood after bark beetle attacks can be observed here. This condition makes the test area perfectly suited to meet the objectives of this research and it allows to put an emphasis on the comparison of the relational and object-relational database modelling approaches with respect to adaptation of user-driven conceptual requirements in general and the entire modelling process of geospatial objects in particular.

Vorwort

Die vorliegende Arbeit ist während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Institut für Geodäsie, GIS und Landmanagement, Fachgebiet Geoinformationssysteme in den Jahren 1999 - 2005, entstanden.

Meinem Doktorvater, Herrn Univ.-Prof. Dr.-Ing. Matthäus Schilcher vom Fachgebiet Geoinformationssysteme der Technischen Universität München möchte ich ganz besonders für die zahlreichen Anregungen, die ständige Förderung und nicht zuletzt für das uneingeschränkte Vertrauen während der gesamten Arbeit danken.
Frau Prof. Dr.-Ing. Liqiu Meng, Lehrstuhl für Kartographie der Technischen Universität München, danke ich für die freundliche Übernahme des Korreferates. Herrn Univ.-Prof. Dr.-Ing. Holger Magel, Lehrstuhl für Bodenordnung und Landentwicklung der Technischen Universität München gilt mein Dank für den Vorsitz der Prüfung.

Weiterhin danke ich Herrn Prof. Dr.-Ing. Georg Lother, Fachbereich Geoinformationswesen der Fachhochschule München, der durch seine zahlreichen fachlichen Anregungen zum Gelingen dieser Arbeit beigetragen hat. Den Herren Maik Sandmann und Hans Viehmann von Oracle Deutschland danke ich für Ihre Unterstützung bei technischen Fragen und ihre Anregungen.

Mein Dank gilt ferner Herrn Dr. Peter Biber vom Lehrstuhl für Waldwachstumskunde für seine wertvollen Hinweise bei forstwissenschaftlichen Fragen sowie Herrn Prof. Dr. Reinhard Mosandl vom Lehrstuhl für Waldbau und Forsteinrichtung für die Zusage von Verjüngungsdaten im Nationalpark Bayerischer Wald. Ganz besonders herzlich danken möchte ich Frau Dr. M. Bauer für Ihre Bereitschaft zur fachlichen Erläuterung der Datenbestände. Ausserdem danke ich Herrn Arthur Reinelt von der Nationalparkverwaltung Bayerischer Wald für die von Beginn an beständige und kreative Zusammenarbeit.

Besonders unterstreichen möchte ich ferner die wertvolle Unterstützung innerhalb des GIS-Teams am Fachgebiet Geoinformationssysteme der Technischen Universität München. In diese Arbeit sind außerdem die Erkenntnisse der Diplomarbeiten von Tobias Hochhäusler und Alexander Eichinger eingegangen.

Die vorliegende Arbeit widme ich meinen Eltern, die meine wissenschaftlichen Interessen stets unterstützt haben.

München im April 2005

Inhaltsverzeichnis

Kapitel 1

Einführung

Den klassischen Ansatz zur kombinierten Nutzung von Geodaten aus verschiedenen Datenquellen stellt nach wie vor die *Datenintegration* dar. Der physische Datenaustausch im Vorfeld des Integrationsprozesses führt zur Duplizierung von Geodatenbeständen was mit enormem Aufwand verbunden ist, um die eigene Datenbasis laufend aktuell zu halten. Sind die Geodaten veraltet, muss der Prozess der Datenbeschaffung, ggf. Aufbereitung und -Integration zyklisch wiederholt werden. Dies trifft in besonderem Maße für den Aufbau von Geoinformationssystemen (GIS) zu, die im Gegensatz zu GIS-Projekten die langfristige Bereitstellung und Pflege von Geobasis- und / oder Geofachdaten verfolgen.

Vor dem Hintergrund der Aktualität als wichtigstes Qualitätskriterium von Geodaten, der gewonnenen Erkenntnis über die zu erzielende Wertschöpfung durch das *Wirtschaftsgut Geodaten* und des Siegeszugs des Internet während der letzten Dekade hat sich der Aufbau moderner Geodateninfrastrukturen (GDI) zum Forschungsschwerpunkt Nummer Eins der Geoinformatik entwickelt. Durch die Ad-Hoc-Nutzung verteilter Geodaten über Geo-Web-Services werden Sekundärdatenbestände und damit Mehraufwändungen für die laufende Aktualisierung der Geodatenbasis von vornherein ausgeschlossen.
Während die internetbasierte Erschließung und Kombination heterogener Datenquellen technisch weitgehend gelöst ist und sich somit Auskunftslösungen bereits geraume Zeit realisieren lassen, stößt dieser moderne Ansatz bei der flexiblen Verarbeitung von Datenbeständen zur Analyse komplexer Zusammenhänge noch an seine Grenzen. Die umfangreiche Verkettung von Operationen des Geoprocessing, etwa zur Erforschung des hochkomplexen Beziehungsgeflechts von Prozessen der Natur, ist demnach bislang lediglich auf Basis des klassischen Ansatzes der zentralen Datenintegration realisierbar.

Beim Ansatz der Datenintegration stellt sich nun die Frage, welches zentrale Datenhaltungskonzept den vielfältigen, konzeptionellen Anforderungen der GIS-Anwender am besten gerecht wird bzw. das größte Potential für weitere noch notwendige Entwicklungen bietet.

Aufgrund der besonderen Eigenschaften von Geodaten, die primär in der Mehrdimensionalität, in den topologischen Strukturen und dem im Vergleich zu alphanumerischen Daten enormen Datenvolumen begründet sind, galten Geodaten aus Sicht der Informatik lange Zeit als exotisch und zu komplex, um sie performant in Standarddatenbanken zu verwalten. Darüber hinaus wirkten sich die hohen Entwicklungskosten nachteilig auf die zügige Implementierung und Nutzung von geeigneten Technologien aus. Entsprechend stiefmütterlich wurden raumbezogene Daten lange von den Datenbankherstellern behandelt.
Die Folge war die Trennung der Datenhaltung für Geometrie- und Sachdaten in GIS.
Dieser klassische Modellansatz, der gerade in GIS-Projekten bis heute fast ausschließlich bevorzugt wird, sieht die Verwaltung der Geometrie von Geoobjekten in den speziellen GIS-Dateiformaten des Herstellers über ein Dateisystem vor, während die komplexen Sachdaten in einem Relationalen Datenbanksystem (RDBS) modelliert und gespeichert sind.

Das vollständig integrierte, zuverlässige Management von Geometrie und Sachdaten in Datenbanken gewann in der Geoinformatik erst an Bedeutung, als Vektor- und Rastergeometrien in komprimierter Form auf binäre Basisdatentypen der Relationalen Standarddatenbanken abgebildet werden konnten und raumbezogene Abfragen dank

effizienter Indizierungsmethoden performanter wurden. Dies wurde von den GIS-Herstellern jedoch individuell verschieden gelöst, so dass bis heute stark proprietäre Implementierungen bzw. Datenbankmodelle vorherrschen.

Der Bedarf neben textuellen auch komplexere Daten wie z.B. Multimedia-Objekte in Datenbanken abzulegen ohne die bestehenden Vorteile bei der Verwaltung und Abfrage großer relationaler Datenbestände aufgeben zu müssen, führte zur Erweiterung Relationaler Datenbankmanagementsysteme um objektorientierte Konzepte und schließlich zur Entwicklung der Objektrelationalen Datenbankmanagementsysteme (ORDBMS). Die zunehmende Bedeutung von raumbezogenen Daten für Wirtschaftsprozesse veranlasste die Datenbank-Hersteller, neue auf den nun verfügbaren objektorientierten Modellstrukturen aufsetzende Basisdatentypen zur direkten Speicherung von vektoriellen Geodaten in Tabellenspalten anzubieten. Spezielle, auf den Objekttypen definierte Methoden zur Analyse der gespeicherten Vektoren steigern die Leistungsfähigkeit der ORDBS um ein Vielfaches.

1.1 Motivation und Zielsetzung

Die führenden GIS-Hersteller unterstützen bislang ausschließlich die oben skizzierten räumlichen Basisdatentypen von ORDBMS direkt, so dass neben dem binären Speicherformat des jeweiligen Herstellers ein zweites, objektrelationales Schema für die Speicherung von Vektordaten verfügbar ist. Das enorme Potential zur Modellierung komplexer Objekte, das in Objektrelationalen Datenbanksystemen steckt und für Anwendungen der Geoinformatik speziell für die vollständige Abbildung von Geoobjekten genutzt werden könnte, scheint vor diesem Hintergrund regelrecht verschenkt zu werden. Der Hype um die ehemals als „Wunderwaffe“ gepriesenen ORDBMS ist nahezu vorbei und die Integration von ORDBMS in GIS scheint beendet, ehe sie richtig begonnen hat.
Diese auf den ersten Blick ernüchternde Erkenntnis ist jedoch zugleich Motivation, das Potenzial und die Schwächen einer auf RDBMS und ORDBMS realisierten integrierten Datenhaltung aufzuzeigen und aus Sicht der konzeptionellen Anforderungen des GIS-Anwenders zu bewerten.

Die beiden Kernfragen dieser Arbeit lauten somit:

Wie gut können die konzeptionellen Vorgaben durch die logischen Datenbankmodelle (RDBS bzw. ORDBS) adaptiert werden?
Wie konsequent wurde dieses Potenzial von den GIS- bzw. DBMS-Herstellern bislang genutzt?

Nicht alles was aus Anwendersicht erwünscht und technisch realisierbar ist, muss sich automatisch in der Implementierung der GIS bzw. DBMS wiederfinden. Die Arbeit beleuchtet die möglichen Hindernisse auf dem Weg zur stärkeren Nutzbarmachung von objektrelationaler Technologie in GIS.
Ein Vergleich des relationalen und objektrelationalen Datenhaltungsansatzes hinsichtlich der bei der Migration von Geodatenbeständen entstehenden Datenbankstrukturen, des realisierbaren Mehrwerts für den GIS-Anwender sowie der bestehenden Schwächen sind ebenso Gegenstand der Arbeit.
Die gewonnenen Erkenntnisse bilden die Grundlage zur Definition von Rahmenbedingungen bzw. Klassifizierung von GIS-Anwendungen, für die sich der Migrationsaufwand lohnt bzw. die einzig auf Basis eines bestimmten Datenhaltungskonzepts realisierbar sind. Die Aussagen werden durch die Demonstration ausgewählter GIS-Anwendungen untermauert.
Die Arbeit schließt mit Empfehlungen zur künftigen GIS- und Datenbankentwicklung im Kontext einer scheinbar stagnierenden Nutzbarmachung objektrelationaler Technologie für die Anforderungen der Geoinformatik.

1.2 Gliederung der Arbeit

Kapitel 2 beschreibt den aktuellen technischen Stand der zentralen Modellierungs- bzw. Datenhaltungskonzepte in GIS und die Defizite der unterschiedlichen Ansätze. Es werden die konzeptionellen Anforderungen an das Datenmanagement in der Geoinformatik aus Anwendersicht formuliert und die Strategie dieser Arbeit zur Evaluierung der Leistungsfähigkeit der vorgestellten Ansätze skizziert. Es folgt die Auswahl eines geeigneten Testgebietes und die Vorstellung der zur Untersuchung der Forschungsfragen dieser Arbeit herangezogenen Datenbasis.

Kapitel 3 behandelt theoretische Aspekte objektrelationaler Datenbankmodelle und stellt hier besonders die gegenüber dem relationalen Paradigma vorhandene Effizienzsteigerung für die Geoinformatik heraus. Das aktuelle Potential der objektrelationalen Modellierung wird auf Grundlage der in den ANSI/ISO/IEC-Spezifikationen von SQL:1999 und SQL:2003 eingeführten objektrelationalen Erweiterungen von SQL vorgestellt. Die für raumbezogene Anwendungen bedeutsamen Erweiterungen des Standards zur strukturierten Verwaltung von Geodaten und XML-Daten werden erläutert.

Kapitel 4 beschreibt ausführlich die bei der Migration eines realen Geo-, Sach- und Metadatenbestandes in ein integriertes relationales Datenhaltungskonzept entstehenden Datenbankstrukturen. Auf Basis dieser Strukturen werden Konzepte zur Optimierung der Nutzung und Fortschreibung zentraler Geodatenbasen entwickelt. Die empirischen Untersuchungen zur Leistungsfähigkeit des Ansatzes werden durch die Realisierung konkreter für das Testgebiet relevanter Anwendungen ergänzt.

Kapitel 5 beschreibt die Migration des vorliegenden Ausgangsdatenbestandes in ein integriertes objektrelationales Datenhaltungskonzept unter besonderer Betrachtung der dabei entstehenden Datenbankstrukturen. Es werden mögliche Schwerpunkte für die erweiterte Nutzung von ORDBMS in der Geoinformatik und die noch bestehenden Defizite bei der Integration in GIS thematisiert. Das Analysepotential, das der ORDBS-basierte Ansatz dank der implementierten räumlichen Operatoren und Funktionen bietet, wird am Beispiel konkreter GIS-Anwendungen im Testgebiet Nationalpark Bayerischer Wald demonstriert.

Kapitel 6 vergleicht ausgehend vom bisherigen Konzept der getrennten Datenhaltung die beiden Modellierungs-Paradigmen bzw. die auf ihnen aufsetzenden Datenhaltungsansätze nach ihrer Leistungsfähigkeit. Die in Kapitel 2 aus Anwendersicht formulierten, konzeptionellen Anforderungen werden nochmals explizit aus der Sicht des Ansatzes beleuchtet, der zweckmäßig erscheint, um den jeweiligen Bedürfnissen gerecht zu werden. Der Vergleich schließt die in der Arbeit realisierten Anwendungen mit ein.

Kapitel 7 schließt die Arbeit mit einem persönlichen Fazit ab. Es werden noch einmal die wesentlichen Hemmnisse des Integrationsprozesses von objektrelationalen Konzepten in GIS beleuchtet und Empfehlungen für die künftige Entwicklung im Sinne der Geoinformatik gegeben.

Kapitel 2

Ausgangssituation und Problembeschreibung

Gemäß der heterogenen Entwicklung der Datenbanken und ihrer unterschiedlichen Modellierungs-Paradigmen existieren unterschiedliche Ansätze zur zentralen Verwaltung und Abbildung von Geodaten in GIS. Die konzeptionellen Anforderungen der GIS-Anwender an ein optimales Geodatenmanagement werden durch die derzeit verfügbaren logischen und physikalischen Datenbank-Modelle zur qualitativ vollständigen Abbildung von Geoobjekten bzw. ihrer unmittelbaren Nutzbarkeit in GIS nur unzureichend erfüllt.

Abschnitt 2.1 beschreibt den aktuellen technischen Stand der zentralen Datenhaltung in GIS. Abschnitt 2.2 formuliert die konzeptionellen Anforderungen an die Datenhaltung in der Geoinformatik aus Anwendersicht und skizziert die Strategie dieser Arbeit zur Evaluierung der Leistungsfähigkeit der bestehenden Ansätze durch empirische Untersuchungen. Abschnitt 2.3 stellt das Testgebiet der Arbeit vor und gibt einen Überblick über die zur Evaluierung verfügbare Datenbasis.

2.1 Datenbankkonzepte zur Verwaltung von Geodaten

Die zentrale Verwaltung von großen Geodatenbeständen basiert aktuell auf GIS-Architekturen bzw. Datenhaltungskonzepten, die sich in vier Klassen gruppieren lassen (vgl. Abb. 2.1):

1. **Getrennte Datenhaltung von Geometrie und Sachdaten**

   Der *getrennte Datenhaltungsansatz* stellt den klassischen Modellierungs-Ansatz für Desktop-GIS dar, der durch eine relativ einfache Architektur gekennzeichnet ist. Lediglich komplexe Sachdaten werden in einem RDBS modelliert und verwaltet, sämtliche Geometriedaten sowie deren direkte und ggf. indirekte Attribute werden dagegen ausserhalb des RDBS in GIS-eigenen, proprietären Dateien mit Hilfe des Betriebssystems gespeichert. Die Verküpfung zwischen Geometrie und Sachdaten erfolgt durch Identifikatoren. Metadaten werden Datensatz-begleitend in geeigneten Formaten - vorzugsweise in XML-Dateien - verwaltet.
   Die physikalische Trennung zwischen Geometriedaten und direkten Attributen kann - insbesondere bei der Fortführung der Attribute ausserhalb des GIS - zu Inkonsistenzen führen. Entsprechendes gilt bei der Änderung von Schlüsselwerten auch für die alphanumerischen Daten des komplexen Sachdatenmodells, da hier keine Mittel zur Konsistenzsicherung zwischen Geometrie- und Sachdaten existieren bzw. sich diese nur mit viel Aufwand implementieren lassen.
   Trotz der seit geraumer Zeit verfügbaren höherwertigen Konzepte basieren bis heute die meisten GIS auf diesem einfachen Ansatz.

2. **Integrierte relationale Datenhaltung ohne transparente Konsistenzsicherung**

   Ansatz 2 geht von einer *integrierten Datenhaltung* aus, bei der sämtliche Geometrie-, Sach- und Metadaten über ein RDBMS in ein herstellerspezifisches relationales Datenbankmodell abgebildet werden, also in Relationen gespeichert sind. Dieser Ansatz basiert somit auf der Konvertierung von proprietären Geometrieformaten in Werte binärer Datentypen (z.B. BLOB, Binary Large Object), die in bestimmten Tabellenspalten

des Relationenmodells eines GIS-Herstellers aufgenommen werden. Direkte Attribute und räumliche Indizes werden in separaten Relationen verwaltet und sind über Fremdschlüsselbeziehungen mit den Geometrien verknüpft.
In der Regel bleibt das logische Modell eines GIS-Herstellers in seiner Gesamtheit für den Anwender aber auch für den Datenbankentwickler verborgen (sogenannte Black Box). Die Implementierung ausreichender Integritätsbedingungen für die Konsistenzsicherung zwischen Geometrie und Sachdaten ist entweder unzureichend oder nach aussen nicht transparent. Zur Wahrung der Konsistenz propagieren die GIS-Hersteller, schreibende Transaktionen lediglich mit ihren eigenen GIS-Produkten anzustoßen, da nur diese in Kombination mit der Middleware-Komponente die zugrunde liegenden Datenbankstrukturen ausreichend interpretieren können. Dies gilt in besonderem Maße für die Fortführung direkter Attribute. Die Abhängigkeit vom GIS-Hersteller bleibt also bestehen und wird bei einer Migration von Geodaten aus Ansatz 1 nach Ansatz 2 lediglich von den eigenen Grafikformaten auf das eigene Datenbankmodell übertragen.
Dieser zweite Ansatz ist „State of the Art“ beim Aufbau von Geodatenbanken.

3. **Integrierte relationale Datenhaltung mit transparenter Konsistenzsicherung**

   Ansatz 3 sieht gegenüber Ansatz 2 die zur *Konsistenzsicherung* nötige Implementierung von Integritätsbedingungen über Konzepte der aktiven Datenbank (Constraints und Trigger) und deren Transparenz nach aussen vor. Durch die vollständige Offenlegung des spezifischen Datenmodells eines GIS-Herstellers ergeben sich für den Anwender größere Freiheiten bei der Auswahl der Benutzerschnittstellen. Zudem lässt sich das Datenmodell individuell erweitern, um beispielsweise leichter eigene Automatismen zur automatisierten Fortschreibung von speziellen, eigenständig entwickelten Metadatenkatalogen zu realisieren. Die Deduktion von impliziten Metadaten, etwa das Berechnen der Bounding Box oder das Auslesen des Objektgeometrietyps aus Geodatensätzen, ist zwar bereits in Ansatz 2 von den meisten GIS-Anbietern in deren Produkten umgesetzt. Ein Höchstmaß an Flexibilität bzgl. der Auswahl der Zugriffswerkzeuge und Möglichkeiten zur Modellerweiterung erfordert jedoch zwingend die Kenntnis aller zugrundeliegenden DDL-Strukturen.
   Ansatz 3 ist aus Anwendersicht wünschenswert, wird in der Praxis jedoch bislang kaum berücksichtigt.

4. **Objektrelationale Datenhaltung**

   Nach der Einführung von Objekttypen und der damit möglichen Definition und Komposition abstrakter Datentypen in ORDBS, bieten führende Datenbankhersteller seit geraumer Zeit native Datentypen bzw. Speicherformate für die direkte Verwaltung von Geometrie- und XML-Daten in der Datenbank an (vgl. Abschnitt 3.6). Damit können Geometrie- und Metadaten in Spalten objektrelationaler Tabellen gemeinsam mit ihren direkten Attributen gespeichert werden. Dies ermöglicht zum einen den direkten Zugriff auf die Geometrieobjekte und ihre Metainformationen über die SQL-Schnittstelle der Datenbank, zum anderen wird dadurch unmittelbar die Konsistenzerhaltung zwischen Geometrie und Sach- bzw. Metadaten begünstigt.
   Die GIS-Hersteller demonstrieren hier ihre Kooperation mit den führenden ORDBMS-Anbietern durch die direkte Unterstützung der von der jeweiligen Datenbank angebotenen räumlichen Datentypen inkl. der darauf aufsetzenden Indizierungsmethoden. Das heisst konkret, dass Geometriedaten über das GIS-Frontend als Spaltenobjekte des objektrelationalen Basisdatentyps in der Datenbank abgelegt und dann wie gewohnt angefragt und editiert werden können. Entgegen Ansatz 2 und Ansatz 3 müssen nicht mehr verschiedene relationale Tabellen über Joins verknüpft werden, um die Gesamtsicht auf die geometrische und attributive Information eines Geoobjekts zu erhalten.
   Mittlerweile bieten die Datenbank-Hersteller im Allgemeinen eigene, einfache Geodaten-Viewer zur Visualisierung der Geometriedaten an, so dass neben der Interpretation der Geodaten auch für die einfache grafische Darstellung der Daten zumindest theoretisch auf die GIS-Software-Komponenten verzichtet werden kann.

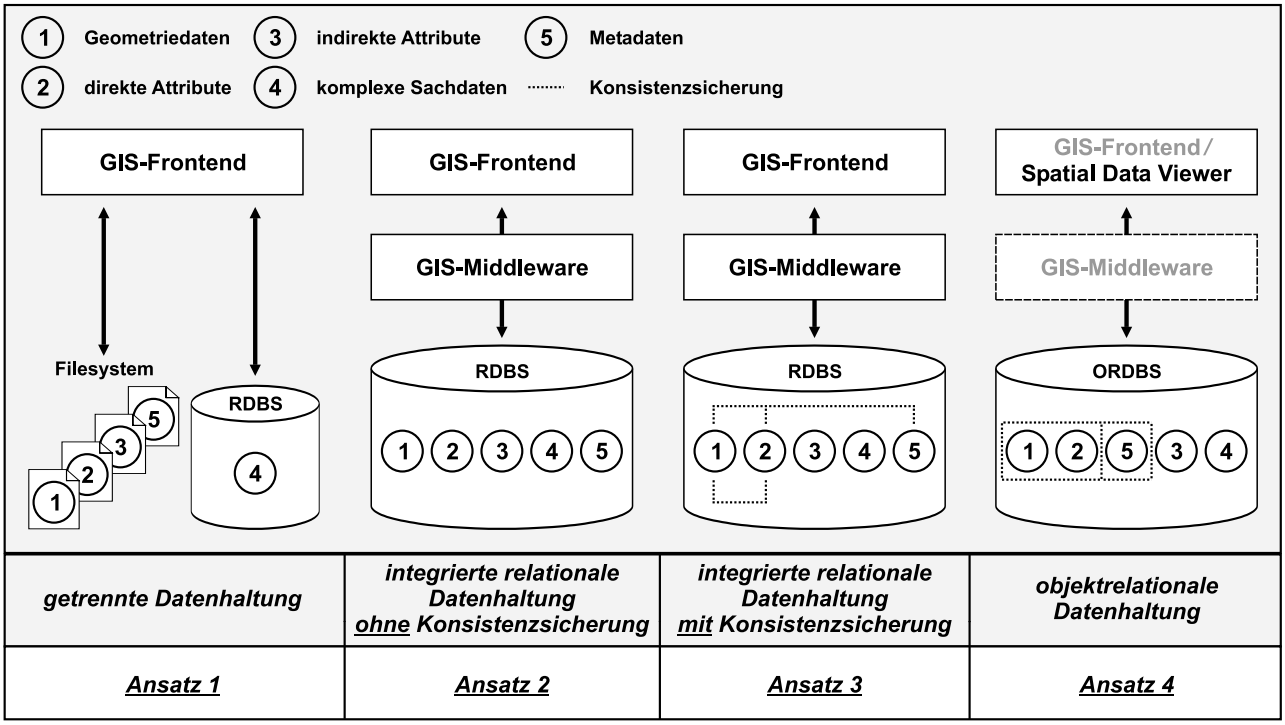

Abb. 2.1: Klassifizierung von GIS-Herstellersystemen und Datenhaltungsansätzen für GIS

2.2 Konzeptionelle Anforderungen an die Modellierung und das Daten-Management in der Geoinformatik

Die konzeptionellen Anforderungen des GIS-Anwenders an ein leistungsfähiges Geodatenmanagement sind vielfältig, lassen sich jedoch unter den Kernanforderungen

- **vollständige Modellierung** (Speicherung) von Geoobjekten

  und

- **flexibler Zugriff** (Austausch) von Geodaten

subsummieren. Für die *vollständige Modellierung von Geoobjekten* ist die Abbildung folgender elementarer Geoobjekt-Komponenten von Bedeutung:

- Geometrie:	Absolute Position und geometrische Ausprägung im 2D- bzw. 3D-Raum
- Attribute:	Kerneigenschaften von Geoobjekten
- Thematik:	Fachliche Bedeutung, Definition bzw. Funktion von Geoobjekten
- Topologie:	Räumliche Beziehungen zwischen Geoobjekten
- Dynamik:	Zeitliche Veränderung von Geoobjekten, Versionenverwaltung

Darüber hinaus sind von Bedeutung:

- Annotation:	Beschriftung von Geoobjekten bei Visualisierungen
- Metadaten:	Beschreibung von Geodaten, insbesondere unter qualitativen Gesichtspunkten
- Multimedia:	Audiovisuelle Informationen zu Geoobjekten

Die konzeptionellen Anforderungen des GIS-Anwenders lassen sich zu funktionalen und qualitativen Aspekten zusammenfassen.

Funktionale Anforderungen sind etwa:

1. **Zentrale Benutzerverwaltung und Definition feingranularer Zugriffsrechte**

   Zum Schutz vor unberechtigtem Zugriff muss ein GIS über ein zentrales, datenbankbasiertes Nutzermanagement mit Authentifizierung etwa über Benutzername / Passworte verfügen. Die Funktionalität zur Erteilung von Lese- und Schreibrechten an Benutzer sollte projektiv bis zur Attributebene von Objekten reichen und selektiv objektbasierte Grants innerhalb von Objektklassen zulassen. Erweiterte Sicherheitsmethoden wie etwa Verschlüsselungstechniken sind darüber hinaus wünschenswert.

2. **Flexibler Austausch von Geodaten und Metadaten über das Internet**

   Die zuverlässige Distribution von Geoinformationen zwischen Server und Clients sowie die dezentrale Einpflegung neuer bzw. Fortführung vorhandener Daten muss gewährleistet sein.
   Für die Platzierung einer Geodatenbasis als Knoten einer auf OpenGIS-Standards basierenden GDI sowie für die Definition und Nutzung von Geodiensten muss gewährleistet sein, dass proprietäre Speicherstrukturen für OGC Web Services (OWS), etwa WMS und WFS, interpretiert werden können.

3. **Offener Zugriff auf die Geodatenbasis**

   Der Benutzer sollte flexibel bzgl. der Wahl der Benutzerschnittstelle sein. Die Abhängigkeit von den GIS-Systemen spezifischer Hersteller sollte vermieden werden, stattdessen sollte der direkte Zugriff auf Geodaten mittels SQL erfolgen können (Interoperabilität).

4. **Zusammenführung und Analyse von Geodaten**

   Geodaten müssen über deren Raumbezug und temporalen Attribute in Beziehung gesetzt werden können. Leistungsstarke Funktionen für kombiniertes Geoprocessing zur geschlossenen Auswertung sind ebenso unabdingbar wie die Möglichkeit, dynamische Information algorithmisch aus gespeicherter statischer Information ad hoc abzuleiten.

5. **Flexible Modellierungsmittel zur vollständigen Abbildung von raumbezogenen Daten in 2D und 3D inklusive Zeitbezug**

   Die Komponenten der Systemplattform müssen Datentypen bzw. Werkzeuge für das Design eines flexiblen raumzeitlichen Datenmodells zur Verwaltung und Auswertung räumlicher Daten und ihrer zeitlichen Veränderung anbieten.

6. **Modellierbarkeit bekannter komplexer Objekte und deren Beziehungen**

   Die Möglichkeit zur direkten Modellierung komplexer (aus Komponenten zusammengesetzter) Objekte und ihrer Beziehungen muss bestehen. Trotzdem müssen komplexe Objekte unmittelbar visualisier- und analysierbar bleiben.
   Dabei kommt insbesondere der vollständigen Abbildung von Geoobjekten durch die Modellierung aller ihrer oben aufgeführten Komponenten entscheidende Bedeutung zu.

7. **Skalierbarkeit der Systemplattform**

   Beim Aufbau zentraler Datenbestände ist es riskant, nur den aktuellen Bedarf zu betrachten. Um Investitionen langfristig zu sichern und auch heute noch nicht absehbaren, künftig aber gewünschten Anforderungen zu genügen, muss die zugrunde liegende Systemplattform hochskalierbar sein. Die Skalierbarkeit ist essentiell insbesondere im Hinblick auf das Rasterdatenmanagement infolge der rasant steigenden Anzahl hochauflösender Rasterdaten / Sensoren.

8. **Verfügbarkeit von zentralen, Backup- und Recovery-Mechanismen**

   RAID-Systeme bzw. ähnliche auf Redundanz beruhenden Sicherungen gegen Hardwareausfall sind durch flexible Backup- und Recovery-Mechanismen zu ergänzen. Dabei sollte der Sicherungsprozess bzw. die Wiederherstellung im Störfall möglichst wenig Zeit in Anspruch nehmen.

Als qualitative Anforderungen lassen sich aufführen:

1. **Wahrung der Konsistenz zwischen Geometrie-, Sach- und Metadaten**

   Die Konsistenz zwischen Geometriedaten und direkten Attributen einerseits sowie den zugehörigen Metadaten andererseits muss permanent gesichert sein. Nach der Fortführung von geometrischer oder attributiver Information sollten die Änderungen vom GIS automatisch in den entsprechenden Metadaten-Elementen nachgeführt werden, sofern dies möglich ist (implizite Metadaten, z.B. räumliche Ausdehnung von Geoobjekt-Klassen). Ein vom Benutzer durchzuführendes Update der betreffenden expliziten Metadaten-Elemente (z.B. Angabe des Verarbeitungssystems) sollte zumindest vom System angestoßen und überwacht werden.

2. **Standardisiertes und konsistentes Handling von Annotation**

   Die Speicherung von Texten, die der Beschriftung von Geoobjekten bei Visualisierungen dienen, sollte standardisiert erfolgen.

   Das Beschriftungs-Objekt sollte so mit der Geometrie verknüpft sein, dass im Fortführungsfall die logische Konsistenz des Gesamtobjekts erhalten bleibt. D.h. zunächst, dass Annotation nur solange wie das zugehörige Geoobjekt existiert. Da Annotation i.d.R. als direktes Attribut gespeichert wird, stellt 2. einen Spezialfall von 1. dar.
   Weiterhin sollte die optimale Platzierung von Beschriftungen relativ zur Geometrie durch Regeln definiert und nach der geometrischen Fortführung vom System neu berechnet werden können (Feature Linked Annotation).

3. **Mehrbenutzerzugriff und Transaktionskonzept für Geometriedaten (lange Transaktionen)**

   Bei schreibenden Zugriffen auf Geodaten durch mehrere Benutzer muss die Konsistenzerhaltung durch zuverlässige und intelligente Sperrmechanismen bzw. Versionierungskonzepte garantiert sein.

4. **Ressourcen-Transparenz bzgl. des vorhandenen Datenangebots für lesende Zugriffe**

   Der gesamte Content einer Geodatenbasis und die genaue fachliche Bedeutung von Geoobjekten muss für lesende Zugriffe im Sinne des Transaktionskonzepts zu jeder Zeit gewährleistet sein. Der lesende Zugriff muss jedoch durch die explizite Zuweisung von Objekt-Berechtigungen auch räumlich und thematisch eingeschränkt werden können.

5. **Zuverlässigkeit des Gesamtsystems**

   Die Stabilität einer Geodatenbank und die Performanz des Zugriffs sind Voraussetzung für die Hochverfügbarkeit der vorgehaltenen Geodaten und damit für die optimale Ausschöpfung ihres wirtschaftlichen Marktwertes.

2.3 Evaluierung der Leistungsfähigkeit von Datenbankkonzepten

Die in Abschnitt 2.2 vorgestellten konzeptionellen Anforderungen des GIS-Anwenders werden von den in Abschnitt 2.1 skizzierten Datenhaltungskonzepten in unterschiedlichem Maße erfüllt. Die ihnen zugrunde liegenden Datenbankmodelle bieten aufgrund ihres verschiedenartigen Angebots an Modellierungskonstrukten unterschiedliche Voraussetzungen zur qualitativ ansprechenden Abbildung aller Komponenten von Geoobjekten.

Objektrelationale Datenbanksysteme (ORDBS) versprechen zwar nicht zuletzt aufgrund der Erweiterbarkeit ihres Typsystems Fortschritte u.a. bei der realitätsnäheren Abbildung von Geoobjekten. Doch auch sie konnten bislang die Lücke, die nach wie vor zwischen Anspruch der Anforderungen und Wirklichkeit der aktuell verfügbaren Systeme und Lösungen klafft, nicht vollständig schließen.

Zielsetzung dieser Arbeit ist es, die Leistungsfähigkeit und Grenzen aktueller Datenbankkonzepte bzgl. eines effizienten Geodaten-Managements im Sinne der logischen und physikalischen Abbildung der konzeptionellen

Maximalanforderungen zu evaluieren. Die Arbeit leistet somit einen Grundlagenbeitrag, um objektrelationale Datenbanksysteme künftig stärker in GIS zu integrieren und das vorhandene Modellierungspotential besser für die vollständige Abbildung von Geoobjekten nutzen zu können.
Die gewonnenen Erkenntnisse dieser Arbeit basieren auf empirischen Untersuchungen, die im Zuge der Migration eines Referenz-Geodatenbestandes in ein relationales (RDBS) bzw. objektrelationales Datenbanksystem (ORDBS) durchgeführt wurden. Die Aussagen stützen sich ferner auf Erfahrungen aus der Entwicklung ausgewählter GIS-Anwendungen, die unmittelbar auf den erzeugten Datenbankstrukturen aufsetzen.

Die Rahmenbedingungen dieser Arbeit werden maßgeblich durch die zugrunde liegende Systemplattform bestimmt, die auf einer Oracle-Datenbank sowie den ESRI-Komponenten *ArcSDE* als GIS-Middleware und *ArcInfo* als High-End GIS-Software basiert. Die Ergebnisse und Empfehlungen dieser Arbeit stützen sich somit auf die entsprechenden proprietären Datenbankstrukturen, die im Zuge der Migrationsprozesse entstanden sind.

Gerade Objektrelationale Datenbanken, die aufgrund Ihrer hohen Flexibilität und der weiteren Unterstützung von SQL als standardisierte Anfragesprache prädestiniert für die Verwaltung und den Zugriff auf komplexe Massendaten sind (vgl. [STONEBRAKER et al., 1999]), sollten in Kombination mit GIS hervorragende Voraussetzungen zur intelligenten Modellierung, Verwaltung, Analyse und Visualisierung von sämtlichen vorgestellten Geoobjektkomponenten bieten. Kapitel 3 widmet sich deshalb zunächst den in den Standards SQL:1999 und SQL:2003 eingeführten objektrelationalen Erweiterungen der Modellstrukturen, die gerade auch als Grundlage zur effizienten und vollständigen Abbildung und Analyse von Geoinformationen genützt werden sollten.

2.4 Auswahl eines Testgebiets

Die Akquisition geeigneter Datenbestände für Lehrzwecke und neue Forschungsvorhaben in GIS ist zumeist mit enormen zeitlichem und finanziellem Aufwand verbunden. Aus diesem Grund wurde bereits von 1996 bis 2002 am Fachgebiet Geoinformationssysteme der Technischen Universität München ein Referenz-Geodatenbestand zum „Nationalpark Bayerischer Wald“ aufgebaut, der bis heute die GIS-Ausbildung an der Hochschule unterstützt und aktuellen und zukünftigen Forschungsprojekten als wertvolle Basis dient. Das Referenz-GIS „Nationalpark Bayerischer Wald“ stellt mit seinem wertvollen Datenbestand somit auch die Grundlage für die Realisierungen und wissenschaftlichen Untersuchungen dieser Arbeit dar.

Die Region des Nationalparks Bayerischer Wald war als Testgebiet wegen der dort verfügbaren Datenvielfalt und -fülle ideal geeignet. Die Symbiose von Natur und Technik sowie die Analyse der Wechselwirkungen in einem sich selbst überlassenen Ökosystem stellte einen zusätzlichen Reiz dar. Zudem war durch die besondere geographische Lage des Nationalparks die Voraussetzung für transnationale GIS-Anwendungen gegeben, insbesondere da mit dem tschechischen Nationalpark Šumava ein ähnlich schützenswerter Lebensraum mit vergleichbaren Szenarien und Aufgaben direkt an den Nationalpark Bayerischer Wald angrenzt.

Abb. 2.2: Testgebiet „Nationalpark Bayerischer Wald“ im Grenzgebiet Deutschland / Tschechien

Getreu dem Motto des Nationalparks, „die Natur sich selbst zu überlassen“, werden Schädlinge wie der Borkenkäfer in der Kernzone des Parks nicht, wie in Wirtschaftswäldern üblich, durch den Menschen bekämpft. Nachdem 1993 eine Borkenkäfermassenvermehrung einsetzte, die bis heute andauert, sind von ca. 24.000 ha des Nationalparks mittlerweile ca. 4000 ha Fichtenwald abgestorben. Die zeitliche Ausbreitung der Schadensflächen im dreidimensionalen Raum und ihre Beeinflussung durch natürliche und anthropogene Faktoren, machte den Einsatz von GIS als leistungsstarkes Analysewerkzeug nahezu unverzichtbar.

Als Testgebiet für die vorliegende Arbeit ist der Nationalpark Bayerische Wald also nicht primär aufgrund der bereits verfügbaren, ausgezeichneten Geodatengrundlage als vielmehr wegen der vorliegenden Kernanwendung ideal geeignet, für die die Berücksichtigung der dritten und vierten Dimension beim Modellierungs- und Analyseprozess unerlässlich ist. Der komplexe natürliche Prozess der zeitlichen Entwicklung der Totholzflächen mit seinem diffizilen Korrelationsgeflecht der dabei einwirkenden Faktoren (vgl. Abschnitt 5.3.1.1) bietet somit eine vorzügliche Ausgangssituation, um die Leistungsfähigkeit eines ORDBS zu evaluieren und dem auf relationalen Strukturen aufsetzenden GIS-basierten Ansatz gegenüber zu stellen. Dies ist insbesondere deshalb der Fall, da sich alle Einflussfaktoren der Totholzausbreitung vollständig durch Geometrie- und Sachdaten beschreiben lassen.

Das Referenz-GIS „Nationalpark Bayerischer Wald“ basiert mit der getrennten Datenhaltung auf dem zur Entwicklungszeit klassischen Ansatz für Desktop-GIS (vgl. Ansatz 1 aus Abschnitt 2.1) und weist daher eine relativ einfache logische GIS-Architektur auf.
Geometrie- bzw. direkte und indirekte Attribute sowie komplexe Sachdaten werden getrennt voneinander gespeichert und verwaltet. Während Geometriedaten in proprietären, für die schnelle grafische Darstellung optimierten Datenformaten gespeichert und mit Hilfe des Betriebssystems in einem Dateisystem verwaltet werden, sind komplexe Sachdaten nach dem relationalen Modell in einem gängigen RDBMS abgebildet. Geometrie- und Sachinformationen sind über gemeinsame eindeutige Schlüsselfelder verknüpfbar (georelationaler Modellierungsansatz). Der Zugriff auf die Geometrie- und Attributinformation erfolgt über die GIS-Software, der Zugriff auf die Sachdateninformationen ist sowohl über das RDBMS oder optional auch via ODBC über das GIS-Frontend möglich. Eine zentrale Metadatenbank zur vollständigen Dokumentation der aus heterogenen Quellen akquirierten ca. 800 Geodatensätze und Sachdaten mit einem Datenvolumen von über 20 GB ist ebenfalls relational modelliert (vgl. [HUBER (2), 2002]).

Die folgende Übersicht beschreibt die wichtigsten Datenquellen des Referenz-GIS „Nationalpark Bayerischer Wald“, dessen Datenbestand für die empirischen Untersuchungen dieser Arbeit fungiert. Die Erkenntnisse dieser Arbeit beruhen somit auf realen und nicht auf synthetischen Geodaten.

Amtliche Geo-Basisdaten der Bayerischen Vermessungsverwaltung	**Behördliche Daten der Nationalparkverwaltung sowie des Bayerischen Staatsministeriums für Landwirtschaft und Forsten**	**Datenbestände des tschechischen Nationalpark Šumava**	**Privatwirtschaftliche Datenbestände**
DFK 1:1 000	Waldkarten 1:10 000	infrastrukturelle Datenbestände	multispektrale Fernerkundungsdaten
Flurkarte 1:5 000	Forstbetriebskarten 1:10 000	umweltrelevante Datenbestände	panchromatische Fernerkundungsdaten
AGLB	Standortskarten 1:10 000		Temperaturverteilungen
ATKIS DLM 25/1	historische Schadensklassifizierungen		Schadensklassifizierungen
ATKIS 500 Bayern	Totholzkartierungen		
DOP 1:5 000	Geologische Karten 1:25 000		
DGM 25	Waldinventuren		
TK 25	IR-Luftbilder		
TK 50	Klimadaten		
TK 100	Walddecker		
ÜK 500	Tourismus-Daten		
	sonstige Erhebungen		

Tab. 2.1: Übersicht der Datenbestände des Referenz-GIS „Nationalpark Bayerischer Wald“

Kapitel 3

Grundlagen Objektrelationaler Datenbankmodelle in GIS

Das folgende Kapitel beschreibt Grundlagen und theoretische Aspekte der objektrelationalen Modellierung in Datenbanksystemen auf Basis der objektorientierten Erweiterungen von SQL:1999 bzw. SQL:2003. Darauf aufbauend wird in Kapitel 5 die Migration von Datensätzen des Testgebiets in ein ORDBS sowie die Entwicklung ausgewählter objektrelationaler GIS-Anwendungen für den Nationalpark Bayerischer Wald beschrieben. In Kapitel 6 werden die durch diesen Ansatz zu erzielenden Mehrwertaspekte erläutert und dem integriert relationalen Ansatz gegenübergestellt.

Im Folgenden werden zunächst allgemeine Anforderungen an Datenbankmanagementsysteme sowie grundlegende Begriffe und Definitionen festgelegt, auf deren Basis die weiteren Ausführungen erfolgen. Anschließend wird ein kurzer Abriss der historischen Entwicklung der Datenbanksysteme und der unterstützten Datenbankmodelle gegeben, ehe detailliert auf das objektrelationale Paradigma eingegangen wird. Die Beschreibung der in den ANSI/ISO/IEC-Spezifikationen von SQL:1999 und SQL:2003 eingeführten objektrelationalen Erweiterungen von SQL sowie der auf ihnen aufsetzenden Strukturen zur Speicherung und Verarbeitung von raumbezogenen und XML-Daten bilden die theoretische Grundlage der technischen Umsetzung in Kapitel 5.

3.1 Allgemeine Grundlagen und Definitionen

Zur eindeutigen Begriffsfestlegung werden zunächst folgende Definitionen getroffen:

Eine **Datenbank (DB)** ist nach [SAAKE et al., 1997] eine *strukturierte Sammlung von Daten, welche Fakten über spezielle Anwendungen des modellierten Ausschnitts der realen Welt repräsentiert, der dauerhaft (persistent) und weitgehend redundanzfrei gespeichert wird.*

Ein **Datenbank-Management-System (DBMS)** umfasst die *Software, die eine Sammlung von Programmen bereitstellt, die das anwendungsunabhängige Erzeugen, Ändern und Löschen einer Datenbank ermöglicht.*

Unter einem **Datenbanksystem (DBS)** wird stets die *Kombination eines DBMS mit einer oder mehreren, unterscheidbaren Datenbanken* verstanden.

Ein **Datenbankmodell** definiert nach [TÜRKER, 2003] *die möglichen Strukturen und Funktionen, die zur Beschreibung und Manipulation einer Datenbank eingesetzt werden können.*
Oft wird anstelle von Datenbankmodell auch verkürzt von Datenmodell gesprochen oder aber der Begriff wird als Synonym für Datenbankschema verwendet.

Datenbankschemata werden mittels einer Data Definition Language (DDL) festgelegt. Instanzen eines solchen Schemas, d.h. die modellierten Objekte der realen Welt, können dann mit einer Data Manipulation Language (DML) erzeugt, geändert und gelöscht werden. Eine *Anfragesprache (Query Language)* dient zur Abfrage der Instanzen. Einschränkungen bzgl. möglicher Ausprägungen von Datenbankschemata lassen sich durch *Integritätsbedingungen* festlegen.

Wenn unterschiedliche Anwendungen auf dieselbe Datenbank zugreifen, so sind die Anforderungen ebenfalls verschieden. Es gibt eine Reihe unterschiedlicher Betrachtungsweisen seitens der Anwender. Das Konzept der **Datenunabhängigkeit** verfolgt das Ziel, eine Datenbank von notwendigen Änderungen der Anwendung abzukoppeln (und umgekehrt). Man unterscheidet:

- *Physische Datenunabhängigkeit:* Die Konzeptionelle Sicht auf einen Datenbestand ist unabhängig von der für die Speicherung der Daten gewählten internen Datenstruktur.
- *Logische Datenunabhängigkeit:* Entkopplung der Datenbank von Änderungen und Erweiterungen der Anwendungsschnittstelle.

Die Datenunabhängigkeit wird durch die **Drei-Ebenen-Schema-Architektur** erreicht, die eine Aufteilung der Datenbankbeschreibung auf drei Schemaebenen vorsieht (vgl. Abb. 3.1):

- Das *interne Schema* legt die konkret zur Implementierung der Datenbank intern genutzten physischen Datenstrukturen fest.
- Das *konzeptionelle Schema* gibt den grundlegenden Aufbau der Datenstruktur wieder. Es beschreibt eine von der konkreten internen Realisierung der Datenspeicherung abstrahierende Gesamtsicht auf den gespeicherten Datenbestand.
- *Externe Schemata* beleuchten im Allgemeinen nur einen Teilbereich des konzeptionellen Schemas, der für die jeweilige Anwendung relevant ist. Sie legen also anwendungsspezifische Sichten auf die Datenbank fest, über die in der Regel die Datenbankzugriffe erfolgen.

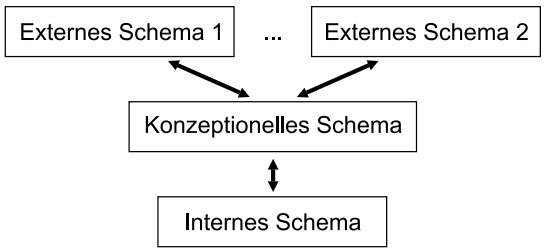

Abb. 3.1: Drei-Ebenen-Schema-Architektur für Datenbanken

Während jede Datenbank genau ein internes und ein konzeptionelles Schema hat, sind somit jeweils mehrere externe Schemata möglich. Das Konzept der externen Schemata erleichtert nicht nur die Konzentration auf das Wesentliche in einer Anwendung, sondern begünstigt die Datensicherheit und den Datenschutz, indem sensible Daten nur in jenen externen Schemata auftreten, wo entsprechende Zugriffsrechte nachgewiesen werden.

Die minimale, von einem DBMS zu erbringende Funktionalität ist in der Literatur übereinstimmend dargelegt. Nach [CODD, 1982] sind mindestens folgende Anforderungen an ein DBMS zu stellen (Codd'sche Regeln):

1. **Integration:** Das DBMS ermöglicht die einheitliche Verwaltung aller von den Anwendungen benötigten Daten.
2. **Operationen:** Ein DBMS muss Operationen zur Datenspeicherung, zur Abfrage und zum Ändern des Datenbestandes zur Verfügung stellen.
3. **Schemakatalog:** Der Schemakatalog beinhaltet die Datenbeschreibungen der Datenbank in einer Form, die Zugriffe von aussen gestattet.
4. **Benutzersichten:** Die Definition externer Schemata wird vom DBMS unterstützt.
5. **Integritätssicherung:** Die Integritätssicherung überwacht die (semantische) Korrektheit des Datenbestandes.

6. **Zugriffskontrolle:** Unautorisierte Zugriffe auf die gespeicherten Daten werden durch Authentifizierung der Benutzer unterbunden.

7. **Transaktionen:** Eine Transaktion ist eine Folge von Datenbankänderungen, die als eine Einheit ausgeführt und deren Resultat bei Wahrung der Konsistenz persistent in der Datenbank gespeichert wird.

8. **Synchronisation:** Transaktionen im Mehrbenutzerbetrieb werden synchronisiert, um Schreibkonflikte auf gemeinsam benötigten Datenbeständen zu vermeiden.

9. **Datensicherung:** Die Datensicherung ermöglicht die (vollständige) Wiederherstellung von Daten auch nach Systemfehlern.

Als weitere oben nicht aufgeführte Forderung gilt, dass ein DBMS die Datenspeicherung über die Lebensdauer einer Anwendungsausführung hinaus garantiert (Persistenz).

Ein DBMS bietet im Sinne der Datenunabhängigkeit eine Schnittstelle, welche die Anwendungsprogrammierer davon befreit, die oben aufgeführten Aufgaben eines DBMS in jedem Anwendungsprogramm neu implementieren zu müssen. Je „höher“ die Datenbankschnittstelle ist (high-level interface), desto kompakter werden die Anwendungsprogramme (AP). Die folgende Abbildung nach [TÜRKER, 2003] veranschaulicht dies:

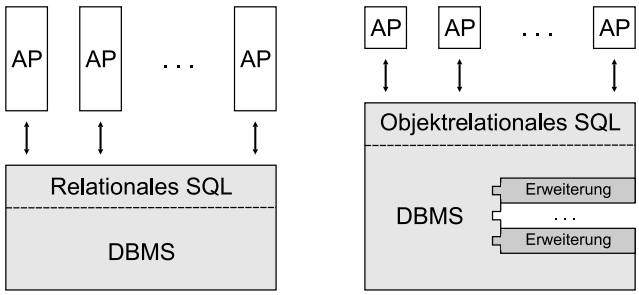

Abb. 3.2: Architektur von Datenbanksystemen

Die Datenbankschnittstelle hängt unmittelbar vom zugrunde liegenden Datenbankmodell ab.
Eine relationale Datenbankschnittstelle bietet generische Operationen zum Erzeugen, Manipulieren und Entfernen von Relationen an. Objektrelationale Datenbanken (ORDBMS) sehen eine höhere Schnittstelle vor, die neben Operationen für Relationen auch benutzerdefinierte Prozeduren und Funktionen für objektorientierte Strukturen unterstützt. ORDBMS sind erweiterbar im Sinne, dass benutzerdefinierte Funktionalität transparent für alle Anwendungsprogramme vom DBMS unterstützt wird.
Auf die Anforderungen von GIS-Anwendungen übertragen bedeutet dies, dass sich dadurch komplexe Datentypen zur Verwaltung von Vektor-, Raster- und Metadaten sowie Methoden für raumbezogene Analysen definieren lassen.

3.2 Entwicklung von Datenbanken und Modellierungskonzepten

Flache Dateien, die wenig Struktur aufweisen und üblicherweise in **Dateisystemen** gespeichert werden, eignen sich nach [BARTELME, 2000] nur für einfache konkrete und statische Anwendungen, sind jedoch für Analysen eines objektstrukturierten Geodatenbestandes ungeeignet, da sie nur sehr beschränkt abfragbar, verknüpfbar und veränderbar sind. Die Forderung nach der Erweiterbarkeit und Anpassung bestehender Strukturen für den flexiblen, parallelen Datenzugriff beliebiger Anwendungen sowie die Wahrung der Datenintegrität erklären die gewaltigen Anstrengungen im Bereich der Datenbankentwicklung. Detaillierte Darstellungen der Vorzüge und wesentlichen Defizite von Dateisystemen gegenüber Datenbanksystemen finden sich in [NEUMANN, 1996] und [STONEBRAKER et al., 1999].

Seit drei Jahrzehnten bestimmen Datenbankkonzepte die Organisationsformen großer Datenmengen. Durch die Ablösung sequentieller Speichermedien wie Magnetbänder durch Direktzugriffsspeichereinheiten war es möglich geworden, Beziehungsnetze zwischen Objekten der Realität in computergestützten Datenbanken darzustellen. Die einzelnen Paradigmen der Datenbankmodellierung haben sich seit jeher gegenseitig beeinflusst und führten zur Erweiterung bestehender Datenbanktypen. Obwohl in der technischen Datenbankliteratur viele Modellierungstechniken vorgeschlagen wurden, haben sich doch nur wenige im kommerziellen Betrieb durchsetzen können. Die DBMS der ersten Generation der frühen 70iger Jahre umfassen hierarchische und netzorientierte Datenbanken.

3.2.1 Hierarchisches Datenbankmodell

Hierarchische Datenbankmodelle traten speziell ab 1969 durch die Einführung des Information Managegment Systems (IMS) von IBM in Erscheinung. Sie bieten sich für Daten an, die immer nur durch 1:n Assoziationen über mehrere Ebenen verknüpft sind und keine Querverbindungen innerhalb der Ebenen zulassen. Diese hierarchische Assoziation zwischen den Entitätstypen entspricht einer Baumstruktur der jeweiligen Instanzen. Zu jeder Instanz gibt es genau eine unmittelbare Vorgängerinstanz. Dies impliziert, dass zwischen zwei beliebigen Instanzen in der Baumstruktur nur eine einzige Verbindung hergestellt werden kann. Hierarchische Datenbanken ermöglichen zwar einen relativ schnellen Zugriff, setzen jedoch eine einheitliche Beziehung vom Typ 1:n voraus, die explizit in der Struktur genutzt wird, ohne sie speziell abzuspeichern. Daraus ergeben sich eklatante Einschränkungen bzgl. der Flexibilität der Modellierung.

3.2.2 Netzwerkmodell

Das Netzwerkmodell geht auf die Normierungsbemühungen von CODASYL (Conference on Data Systems Languages) zurück und ist zusammen mit seiner historischen Entwicklung in [OLLE, 1981] umfassend beschrieben. Im Gegensatz zu der vertikalen Baumstruktur des hierarchischen Modells, bei dem jeder Knoten nur einen Herkunftsknoten besitzt, kann ein Knoten im CODASYL-Datenbankmodell mehrere Eltern-Knoten besitzen, wodurch sich eine Netzstruktur ergibt. Basis ist eine über Zeiger festgelegte Reihenfolge von Instanzen (Member), die einer Vorgängerinstanz (Owner) zugeordnet sind und nach Durchlaufen des Member-Satzes (Set) wieder an den Owner zurückverweisen. Anfragen an Netzwerkdatenbanken werden als Programme - in der Regel in COBOL - formuliert. Manche Anfragen sind auf einer gegebenen Netzwerkdatenbank mit ihrer starren, auf singuläre spezifische und statische Anwendungen ausgerichteten Struktur jedoch nur sehr aufwendig zu lösen. Es wird eine künstliche Reihenfolge induziert, die sich bei der Verwaltung dynamischer Datenbestände nachteilig auswirkt. Die Berücksichtigung vieler möglicher Abfragen macht wiederum den resultierenden Datenbankentwurf sehr komplex und erschwert die Fortführung des Datenbestandes, da ein Datensatz in alle Assoziationen eingebunden werden muss, an denen seine Objektklasse als Member oder Owner teilnimmt. Für eine detaillierte Beschreibung und Diskussion der Vorzüge und Schwächen des Netzwerkmodells sei an dieser Stelle auf [OLLE, 1981] und [DADAM, 2000] verwiesen.

3.2.3 Relationales Datenbankmodell

Relationale Datenbanken gelten als Datenbanksysteme der zweiten Generation und arbeiten nach dem von [CODD, 1970] im Jahre 1970 eingeführten Relationenmodell. Erste kommerziell verfügbare DBMS, die dieses

Modell unterstützen, kamen Anfang der 80iger Jahre auf dem Markt und etablierten sich insbesondere bei Neuentwicklungen schnell zum Marktführer für den Einsatz von Datenbanktechnik in Anwendungssystemen. RDBMS stellen bis heute die in den kommerziellen Bereichen (inkl. GIS) primär eingesetzten Systeme dar. Die Relationale Algebra gibt eine ausgereifte formale Grundlage für derartige Systeme vor und mit dem ANSI SQL-Standard existiert eine standardisierte Sprachschnittstelle für RDBMS. Kommerzielle relationale DBMS-Produkte haben einen langjährigen Reifeprozeß durchlaufen und sind führend hinsichtlich Robustheit, Effizienz und Portabilität. Das Relationenmodell ist an die übliche Datendarstellung in Tabellen angelehnt. Eine Relation ist im Gegensatz zu einer klassischen Tabelle jedoch an keine Reihenfolge der Zeilen gebunden und besteht aus:

- Relationenname
- Relationenschema: Attribute A_1 ... A_n
- Tupeln

Die Zeilen einer Tabelle entsprechen den Tupeln der Relation und bilden die gespeicherten Datensätze. Die Spalten, die analog zu den Zeilen an keine Reihenfolge gebunden sind, werden als *Attribute*, im Sprachgebrauch der Relationalen Algebra auch als *Domänen* bezeichnet. Jedem Attribut wird ein Datentyp zugewiesen, der durch einen Wertebereich und die auf diesen Werten definierten Operationen festgelegt ist. Einer Relation wird ein eindeutiger Relationenname zugeordnet. Das Relationenschema legt die Struktur der Tabelle durch die Vergabe der Attribute fest. Relationale Tabellen werden auch als Tupeltabellen bezeichnet.
Nach der ersten Normalform (1NF) dürfen Attribute nur atomare Werte, etwa Zahlenwerte oder Zeichenketten, nicht aber strukturierte Werte wie Mengen oder weitere Tabellen annehmen. Durch den Ausschluss von Kollektionstypen (vgl. Abschnitt 3.4.2 und 3.5.1) lassen sich im Relationenmodell somit nur flache, ungeschachtelte Tabellen definieren. Höherrangige Normalformen dienen nach [LONEY et al., 2003] der Beseitigung abnormer bzw. dysfunktionaler Beziehungen durch Auflösung (transitiver) Abhängigkeiten von Schlüsselteilen.
Ein Relationstyp R ist nach [KÖNIG-RIES, 2004] in zweiter Normalform (2NF), wenn jedes Nichtschlüsselattribut von jedem Schlüssel voll funktional abhängt. Nichtschlüsselattribute dürfen also nicht von Teilen eines Schlüssels abhängen.
Eine Relation ist in der Dritten Normalform, wenn sie in der Zweiten Normalform ist und jedes Nicht-Schlüssel-Attribut von keinem Schlüsselkandidaten transitiv abhängig ist.

In der Regel wird ein Objekt der Realen Welt durch eine Relation auf jeweils eines ihrer Tupel abgebildet. Bei strukturierten Anwendungsobjekten, deren Attribute keine atomaren Werte annehmen können, ist diese 1:1-Abbildung zwischen Objekten und Tupeln nicht möglich und muss durch geeignetes *Object Mapping* aufgelöst werden. Mögliche Varianten hierzu sind in [AMBLER, 2000] beschrieben.
Das relationale Datenbankmodell kennt ausschließlich Relationen als Abbildungsmittel, es gibt kein explizites Konstrukt für die Modellierung von Beziehungen. Somit lassen sich Beziehungen zwischen Datensätzen zweier Relationen nur über Wertegleichheit und entsprechende Schlüsselfestlegungen ausdrücken bzw. rekonstruieren. Aufgrund des notwendigen Duplizierens von Daten zur Beziehungsmodellierung wird das Relationenmodell als daten- bzw. wertebasiert bezeichnet.
Das relationale Datenbankmodell ist aufgrund seines breiten Einsatzes in nahezu jeder Literatur über kommerzielle Datenbanken beschrieben. Eine ausführliche Darstellung und Bewertung des Modells mit der Herleitung der Normalformen, eine Einführung in SQL:1999 sowie die Relationenalgebra als formale Grundlage finden sich beispielsweise in [GRILL, 1990] und [NEUMANN, 1996]. Die Schwächen des relationalen Datenbankmodells werden ausführlich in Abschnitt 3.3 und an relevanten Stellen im Rahmen der ausführlichen Diskussion des objektrelationalen Paradigmas (vgl. Abschnitt 3.4 und 3.5) diskutiert.

Ab Mitte der 70er Jahre war ein deutlicher Trend in Richtung **abstrakter Datenbankmodelle** erkennbar. Solche weniger implementierungsnahen Modelle erhöhen den Grad der physischen Datenunabhängigkeit, erzwingen aber gleichzeitig einen größeren Aufwand bei der Abbildung auf interne Strukturen bzw. bei der Abbildung von Anwendungsobjekten auf Datenbankobjekte.
Der bekannteste Vertreter derartiger abstrakter Modelle ist das *Entity-Relationship-Modell (ER-Modell)* nach [CHEN, 1976], in dem Datenbestände durch abstrakte Datensätze (Entities = Objekte oder Ereignisse), beliebige Beziehungen (Relationships) und Attribute modelliert werden. Das ER-Modell abstrahiert von den Konzepten

verbreiteter kommerzieller Datenbankmodelle und eignet sich daher vorzüglich zur Generierung von Datenbankschemata für verschiedene Datenbanksysteme. Es erlaubt eine einfache Modellierung, da es mit einem sehr geringen Vorrat an Modellierungskonstrukten auskommt und findet hauptsächlich für den Entwurf von Datenbanken (z.B. bei nahezu allen marktgängigen CASE-Tools) Verwendung.

3.2.4 Semantische Datenbankmodelle

Semantische Datenbankmodelle, die ab 1978 publiziert wurden, erweitern das ER-Modell um weitere Abstraktionskonzepte bzw. komplexe Attributstrukturen oder stellen Neuentwicklungen basierend auf diesen Konzepten dar. Damit stehen ausdrucksstärkere Modellierungskonzepte für eine differenziertere, natürlichere Modellierung zur Verfügung. Durch die Einführung von Strukturierungsmöglichkeiten wie Aggregation, Generalisierung und Gruppierung wird z.B. das ER-Modell zum *Strukturierten Entity-Relationship-Modell (SERM)* erweitert. Damit können auch komplexere Modellierungsaufgaben anschaulich dargestellt werden. Es existiert zwar noch kein Objektbegriff im strengeren Sinne, jedoch wird durch die Annäherung an objektorientierte Konzepte ein fließender Übergang zu den später aufkommenden objektorientierten Datenbanken geschaffen. Auch semantische Modelle werden primär als Entwurfsdatenmodelle eingesetzt. Eine detaillierte Übersicht zu Semantischen Datenbankmodellen und deren speziellen Derivaten gibt [HÖRNER, 2001].

3.2.5 Objektorientierte Datenbankmodelle

Ein gegenläufiger Trend zu den abstrakten Datenbankmodellen wurde durch die objektorientierten Datenbankmodelle eingeleitet, die unmittelbar auf objektorientierten Programmiersprachen wie C++ aufsetzen und ihr Typsystem direkt auf das Datenbankmodell abbilden. Objektorientierte Datenbank-Managementsysteme (OODBMS) unterstützen keine unterschiedlichen Darstellungen des Realweltausschnitts im Rahmen der Drei-Ebenen-Schema-Architektur, die interne Darstellung des Realweltausschnitts entspricht somit direkt der Darstellung im Anwendungsprogramm. Daneben existieren objektorientierte Entwurfsmodelle und Standardisierungsvorschläge, die die Kombination von programmiersprachenbasierten Modellansätzen und datenbankspezifischen Modellierungskonzepten vorangetrieben haben. Besonders zu nennen ist hier das Datenbankmodell des ODMG-93-Standards, bei dem SQL-Konzepte mit in die Entwicklung der objektbasierten Anfragesprache *Object Query Language (OQL)* eingeflossen sind. Entsprechend spiegelt sich die Erweiterung des relationalen Modells um objektorientierte Konzepte wie abstrakte Datentypen, Spezialisierungshierarchien für Tabellen und Typen sowie Objektidentifikatoren in SQL:1999 (SQL3) wider. Inhalte und Konzepte dieser, als „Standard von ORDBMS“ geltenden Version werden ausführlich in Abschnitt 3.4 behandelt. Eine detaillierte Betrachtung des Industriestandards *ODMG-93* der Objekt Database Management Group (ODMG), einem Zusammenschluss der wichtigsten Hersteller objektorientierter DBMS von 1991 gibt [CATTELL, 1994].

An dieser Stelle muss darauf hingewiesen werden, dass der aus den meisten Publikationen bekannte Begriff der *Objektorientierten Datenbankmanagementsysteme (OODBMS)* in Anlehnung an die objektorientierte Programmierung entstand. Da es sich genau betrachtet um Systeme zur Verwaltung von Objekten und nicht primär um objektorientierte Systeme handelt, wird vor allem in neueren Veröffentlichungen häufig auch von *Objektdatenbank Managementsystemen (ODBMS)* gesprochen.

Für ODBMS gibt es, ähnlich wie für Objektorientierung, derzeit keine offizielle, einheitliche Definition. In [ATKINSON et al.,1989] wurden jedoch spezielle Eigenschaften eines ODBMS gefordert. Dieses sog. *ODBMS-Manifesto* vermischt allgemeine Anforderungen an DBMS (vgl. Abschnitt 3.1) mit für Objektdatenbanken spezifischen und unterscheidet nach verbindlichen (mandatory) und optionalen Kriterien:

Als **mandatory features** bzw. **„golden rules“** werden im Manifesto folgende Kriterien aufgeführt. Die dem objektorientierten Paradigma zugeordneten Eigenschaften sind fett gedruckt:

- **Komplexe Objekte:** Komplex strukturierte Datenelemente müssen in einem ODBMS verwaltet werden können.

- **Objektidentität:** Jedes Objekt besitzt eine eigene, eindeutige und unveränderliche Identität (OID). Damit kann zwischen gleichen und identischen Objekten unterschieden werden. Zwei Objekte sind identisch, wenn Sie dieselbe OID besitzen. Sie sind gleich, wenn Sie unterschiedliche OIDs besitzen aber in ihren Attributen und deren Werten übereinstimmen.
- **Kapselung:** Im Sinne der Modularität soll das ODBMS Kapselung, d.h. den (alleinigen) Zugriff auf Objekteigenschaften über öffentliche Methoden, unterstützen.
- **Typen und Klassen:** Jedes Objekt ist einem bestimmten Typ oder einer Klasse zugeordnet.
- **Klassen und Typhierarchie:** Klassen und Typen sind in einer Spezialisierungshierarchie mit Inklusionsbeziehungen und Vererbung eingeordnet.
- **Überschreiben, Überladen, Dynamisches Binden, auch spätes Binden (late binding):** Diese klassischen Eigenschaften der Objektorientierung müssen in einem ODBMS realisiert sein. Vererbte Methoden können spezialisiert werden. Es können mehrere Methoden mit gleichem Namen definiert sein. Die Auswahl einer Methodenrealisierung erfolgt (zur Laufzeit) durch das ODBMS abhängig vom Typ eines Objektes.
- **Berechnungsvollständigkeit:** Das ODBMS besitzt eine berechnungsvollständige Programmiersprache zur Manipulation von Datenbankobjekten und Implementierung von Methoden. Dies ist in herkömmlichen relationalen Systemen nicht gegeben.
- *Erweiterbarkeit:* Vorgegebene Typen müssen durch selbstdefinierte Typen ergänzt werden können.
- *Persistenz:* Objekte werden über die Dauer einer Anwendung hinaus gespeichert.
- *Sekundärspeicherverwaltung:* Die Speicherung der Objekte wird durch das DBMS für den Anwender transparent und effizient durchgeführt (Speicherstrukturen und Zugriffspfade).
- *Synchronisation:* Ein Mehrbenutzerbetrieb muss möglich sein. Dabei bildet die Synchronisation neben dem Recovery eine Komponente der Transaktionsverwaltung.
- *Transaktionen und Recovery:* Ein Transaktionsmanagement nach dem ACID-Prinzip ist verwirklicht. Das ACID-Pradigma steht für die vier Eigenschaften *Atomarität*, *Konsistenz*, *Isolation* und *Dauerhaftigkeit*.
- *Ad-Hoc-Abfragen:* [ATKINSON et al.,1989] fordern eine einfache Möglichkeit für die Abfrage von Daten. Dabei sprechen Sie nicht notwendigerweise von einer Anfragesprache, wie z.B. SQL.

Als **optionale features**, den **„goodies“** werden im ODBMS-Manifesto folgende Kriterien aufgeführt. Die dem objektorientierten Paradigma zugeordneten Eigenschaften sind erneut fett gedruckt:

- **Mehrfachvererbung:** Eine Klasse kann Eigenschaften mehrerer Klassen direkt erben, um u.a. die Wiederverwendung von Methoden zu ermöglichen.
- **Typprüfung:** Eine statische Typprüfung zur Übersetzungszeit ist wünschenswert.
- *Verteilung:* Eine verteilte Datenhaltung sollte unterstützt werden.
- *Lange Transaktionen:* Neben den klassischen ACID-Transaktionen sollten auch lange Transaktionen ermöglicht werden.
- *Versionen:* Um Änderungen an Objekten über die Zeit dokumentieren zu können, sollten Versionen von Objekten verwaltbar sein.

Neben diesen, im ODBMS-Manifesto verankerten Anforderungen haben sich folgende Kriterien als für ODBMS wichtig herausgestellt:

- *Integrität:* Konsistenzbedingungen für Objekte sollten explizit formuliert und überwacht werden können.
- *Sichten:* Anwendungsspezifische Sichten auf den Datenbestand sollten formulierbar sein.
- *Schemaevolution:* Datenbestände sind sehr langlebig. Die Wahrscheinlichkeit für zusätzliche Anforderungen an das Datenbankschema wächst mit der Lebensdauer der Datenbank. Fehler und Unvollständigkeit treten vermehrt mit einer zunehmenden Komplexität des Datenbankschemas auf. Das Datenbankschema muss an neue Anforderungen inkrementell anpassbar sein. Die im Rahmen von SQL:1992 begrenzt mögliche Schemaevolution durch einfache *alter table*-Anweisungen und das Hinzufügen und Entfernen von Integritätsbedingungen ist unzureichend.
- *Zugriffskontrolle:* Der Zugriff auf den Datenbestand muss vom ODBMS kontrolliert werden.

Ein zweites Manifesto nach [STONEBRAKER et al., 1990] (Third-Generation Database System Manifesto) propagierte stattdessen die Evolution bestehender und über Jahrzehnte erprobter Datenbankmodelle anstelle einer insbesondere aus Nicht-Standard-Anwendungen motivierten Revolution. Die Autoren sprechen daher auch explizit nicht von *objektorientierten DBMS* sondern von *DBMS der dritten Generation* bzw. *post-relationalen DBMS*. Dieses Manifesto leitete in gewissem Sinne die Bewegung zu den ORDBMS ein.
Während ODBMS auf einem objektorientierten Datenmodell basieren, ist dies bei post-relationalen DBMS nicht unbedingt der Fall. Gemeinsam ist jedoch beiden Systemkategorien, dass sie die Konzepte der Vererbung und Kapselung beinhalten. Ein relationales DBMS kann in ein post-relationales DBMS überführt werden, wenn es Vererbung, benutzerdefinierte Datentypen und eine persistente Programmiersprache unterstützt. Auf der anderen Seite können ODBMS erst dann als ein DBMS der dritten Generation bezeichnet werden, wenn sie mindestens eine Abfragesprache, einen Abfrageoptimierer, SQL-Unterstützung und ein Regelsystem (z.B. Trigger und Constraints) anbieten.
Das Third-Generation Database System Manifesto enthält 13 Empfehlungen bezüglich der Funktionalität postrelationaler DBMS mit verstärkter Konzentration auf Anforderungen bezüglich der Datenbanksprachen. Eine Übersicht dieser Anforderungen gibt [KÜNG, 1994].

Ein drittes Manifesto nach Darwen und Date [DARWEN et al., 1995] greift den Stonebraker-Ansatz auf und unterstreicht das Potential einer Weiterentwicklung relationaler (insbesondere SQL) Technologie. Die dort betonten technologischen Vorteile von SQL-Systemen waren jedoch auch Motivation für die Entwicklung zukünftiger Objektdatenbanksysteme.

Anders als bei den Konzepten des relationalen Datenbankmodells verlief die Entwicklung von Objektdatenbankmodellen von Beginn an heterogen.
Die Bestrebungen entstammen vielen geistigen Urvätern, die unterschiedliche Zielrichtungen und damit Entwicklungstendenzen verfolgten. Ein allgemein gültiges, einheitlich definiertes Objektmodell existiert nicht. Diesen unterschiedlichen Datenbankmodellen liegt jedoch in der Regel eine bestimmte Menge objektorientierter Grundkonzepte als gemeinsame Basis zugrunde, mit denen ein möglichst anwendernahes Abbilden der realen Welt ermöglicht werden sollte.
Eine detaillierte Betrachtung von ODBMS, ihrer Objektdatenbankmodelle und der von ihnen unterstützten Konzepte gibt [HEUER, 1997].

Der kurze historische Abriss der Datenbankentwicklung und ihrer Modellierungskonzepte wird vervollständigt durch die Betrachtung objektrelationaler Datenbankmanagementsysteme (ORDBMS), die sich seit Mitte der 90iger Jahre auf dem Markt etabliert haben und nach [STONEBRAKER et al., 1999] aufgrund der immer komplexer und umfangreicher werdenden Daten in Datenbanksystemen als die DBMS der Zukunft gelten.

Die Motivation des objektrelationalen Paradigmas resultiert unmittelbar aus

- **den allgemeinen Anforderungen an Datenbanksysteme** (vgl. Abschnitt 3.1)
  und
- **den Defiziten / Nachteilen**
  - relationaler Datenbanksysteme
  - objektorientierter Datenbanksysteme

weshalb es zumeist auch als *das Beste aus beiden Welten* bezeichnet wird (vgl. etwa [PLABST, 2001]).
Relationale Datenbanken sind zwar ausgereift, leistungsfähig und theoretisch untermauert, der alleinige Einsatz des relationalen Konzepts wird vielen Anwendungsfällen jedoch nicht gerecht, da die Modellierung komplexer Aufgabenstellungen und Phänomene der realen Welt durch die Existenz von nur atomaren Datentypen nicht möglich ist.

Objektorientierte Datenbanken sind zwar genau hierzu in der Lage, erschwert wird hier jedoch häufig die Übernahme von in der Regel relational modellierten Datenbeständen. Ausserdem können ODBMS RDBMS nicht vollständig ersetzen, da sie in der klassischen Anwendung für relationale Systeme, der Verwaltung großer, einfach strukturierter Datenbestände, gegenüber den RDBMS konzeptionell im Nachteil sind. Dies ist nicht zuletzt auf das Fehlen bzw. bislang nicht ausreichend standardisierter Anfragesprachen wie OQL zurückzuführen.
Vor der detaillierten Behandlung des in ORDBMS angebotenen Modellierungspotentials in Abschnitt 3.4 und 3.5 werden im folgenden Abschnitt ausführlich die Schwächen des relationalen Datenbankmodells diskutiert.

3.3 Defizite relationaler Datenbankmodelle

Durch ihr relativ einfaches Datenmodell verbunden mit einer standardisierten Anfragesprache unterstützen RDBMS in hohem Maß das Konzept der Datenunabhängigkeit. Ihre Stabilität und die automatische Optimierung prädestinieren sie für die zentrale Verwaltung von Massendaten in klassischen Anwendungen. Nicht-Standard-Anwendungen stellen jedoch wesentlich höhere Anforderungen an die Modellierung, denen RDBMS nur unzureichend gerecht werden können.
Große Mengen komplex strukturierter, heterogener Daten, die beispielsweise während des Softwareentwicklungsprozesses, bei Multimedia- und nicht zuletzt bei GIS-Anwendungen verwaltet werden müssen, stellen RDBMS vor große Probleme. Im Folgenden werden anhand allgemeiner Merkmale von Nicht-Standard-Anwendungen die Defizite relationaler Datenbankmodelle konkretisiert.

1. **Komplexe Datenstrukturen**

   Die zu verwaltenden Daten können nicht direkt durch Tupel in flachen Relationen abgebildet werden. Stattdessen müssen sie, der allgemeinen Normalisierungstheorie folgend, zunächst zerlegt und auf viele Relationen verteilt, bei Anfragen über Multijoins aber wieder aufwendig zusammengesetzt werden. Insbesondere technische und raumbezogene Anwendungen umfassen komplexe Objekte, die häufig aus mehreren trivialen Objekten zusammengesetzt sind (Aggregation), jedoch als eigenständige Datenobjekte betrachtet werden müssen. Die Vielzahl an Fremdschlüsselbeziehungen zwischen den Relationen bzw. die mehrfachen Verbund-Operationen bei Anfragen sind unmittelbar mit erhöhtem Implementierungsaufwand und Performanzproblemen verbunden. Bei der Redefinition von gleichen Attributen bzw. Teilen von Relationenschemata in verschiedenen Tabellen erhöht sich das Risiko für Inkonsistenzen.

2. **Erweiterung des Typsystems**

   Das Typsystem eines gängigen RDBMS ist, entgegen der Forderungen an DBMS, strukturell sowie operational nicht erweiterbar und führt dazu, dass häufig wiederkehrende Datentypen einer Anwendung und typische auf den Werten ausführbare Operationen mehrmals definiert werden müssen. Nicht-Standard-Anwendungen benötigen flexible Mittel zur Typkonstruktion, wie sie etwa aus Programmiersprachen bekannt sind. Dies schließt die Möglichkeit der Definition von datentypspezifischen Operationen (Methoden) ein.

3. **Komplexe Integritätsbedingungen**

   Als Folge der komplexen Objektstrukturen müssen häufig komplexe Integritätsbedingungen berücksichtigt werden, für die die einfachen, vom RDBMS unterstützten Integritätsbedingungen nicht ausreichen.

4. **Semantische Modellierungskonzepte**

   Das Relationenmodell unterstützt keine semantischen und abstrakten Modellierungskonzepte wie Spezialisierung, Aggregation und Assoziation, wie sie beim Datenbankentwurf etwa in einem ER-Modell oder in UML abgebildet werden. Semantische Entwurfsinformationen gehen beim Transformationsprozess im Rahmen des logischen Datenbankentwurfs in der Regel verloren. In einer relationalen Datenbank muss die Entwurfssemantik mittels Relationen und Fremdschlüssel-Beziehungen nachgebildet oder in der Programmlogik nachgereicht werden, was umständlich und damit fehleranfällig erscheint:

   - Spezialisierung

     Die Transformation von in einer Spezialisierungsbeziehung stehenden Entity-Typen auf Relationenschemata erfolgt in der Regel durch Abbildung in unterschiedliche Relationenschemata mit gleichen Primärschlüsseln. Der Primärschlüssel des spezielleren Relationenschemas ist gleichzeitig Fremdschlüssel gegenüber dem allgemeineren Relationenschema. Die vollständige Information über ein spezielles Objekt lässt sich nur aufwendig über die Zusammenführung mehrerer Tupel mittels einer Verbund-Operation abfragen. Umgekehrt muss ein spezielles Objekt beim Speichervorgang zunächst zerlegt und auf unterschiedliche Relationen aufgeteilt werden.

   - Aggregation / Komposition

     Aggregationsbeziehungen zur Beschreibung von Entity-Typen als Komponenten eines übergeordneten Entity-Typs, können im Relationenmodell ebenfalls nur durch getrennte Speicherung von Aggregations- und Komponentenobjekten abgebildet werden. Das erforderliche Zusammensetzen beim Lesen des gesamten Aggregatobjektes muss erneut mittels Verbund-Operationen durchgeführt werden. Existentielle Abhängigkeiten zwischen Aggregat- und Komponentenobjekten (Komposition) sind eigenständig ggf. durch Mittel zur Erhaltung der referentiellen Integrität zu realisieren.

   - Assoziation

     Assoziationen werden im ER-Modell durch Relationships ausgedrückt und können durch verschiedene Kardinalitätsangaben näher charakterisiert werden (SERM). Die Transformation solcher Assoziationen auf das Relationenmodell kann nur durch das Primär-/Fremdschlüssel-Konzept erfolgen, was sich insbesondere bei m:n-Assoziationen nachteilig in Form von erforderlichen Zwischentabellen bzw. zweifachen Verbund-Operationen auswirkt.

5. **Navigierende Zugriffe**

   Zugriffe auf komplexere Datenstrukturen können in RDBMS lediglich durch aufwendige Berechnung von Verbunden, nicht aber durch einfache Navigation zwischen Objekten mittels Referenzen erfolgen. Das Verfolgen von Referenzen (Traversieren) wird in der Regel als effizientere Art des Zugriffs betrachtet und ist direkt mit Performance-Aspekten verbunden.

6. **Identifizierung von Anwendungsobjekten**

   Primärschlüsselattribute zur Identifikation von Anwendungsobjekten in relationalen Datenbanken können neben der Identifizierungssemantik zusätzlich Wertesemantik ausdrücken, also Eigenschaften von Anwendungsobjekten beschreiben. Diese Doppelfunktion führt bei Zustandsänderungen von Anwendungsobjekten zu einem Mehraufwand infolge der nötigen Wertanpassung bei Master- und Detailtabellen. Ferner kann in relationalen Datenbanken nicht zwischen *gleichen* und *identischen* Datenbankobjekten unterschieden werden, da aufgrund der Mengensemantik von Relationen gleiche Tupel nicht erlaubt sind. Diese Probleme lassen sich selbstredend durch die Definition von bzgl. der Wertesemantik unabhängigen reinen Identifikationsattributen (Surrogat-Schlüsseln) eliminieren, jedoch sollte dieser Administrations- und Entwicklungsaufwand automatisch von einem DBMS übernommen werden.

7. **Versionierung von Anwendungsobjekten**

   Ein Versionierungskonzept zur Verbesserung einer kooperativen, mehrbenutzerspezifischen Anwendungsentwicklung lässt sich in relationalen Datenbanken nur durch die Einführung von Zeitattributen und das Duplizieren von Datenbankobjekten nachbilden, da verschiedene Versionen eines Anwendungsobjektes denselben Identifikator tragen müssen. Ein DBMS, das ein Versionierungskonzept unterstützt, lässt diesen Mehraufwand gar nicht erst entstehen.

8. **Verhalten von Anwendungsobjekten**

   Die strikte Trennung von Daten und zeitlichen Abläufen (Verhalten) auf Basis der definierten Datenstrukturen in RDBMS wird den starken Abhängigkeiten zwischen Daten- und Verhaltensaspekten nicht gerecht. Verhalten basiert auf den vorhandenen Datenstrukturen und bewirkt Datenbankänderungen, bei denen Integritätsbedingungen eingehalten werden müssen. Abläufe, die unabhängig von Anwendungen sind, sollten zentral behandelt und somit vom DBMS verwaltet werden. Auf diese Weise ist eine zentrale Überwachung komplexer Integritätsbedingungen möglich.

9. **Impedance Mismatch**

   Die Unterschiede zwischen dem mengenorientierten, deklarativen SQL und den häufig prozeduralen, satzorientierten Programmiersprachen sowie die Inhomogenitäten aufgrund unterschiedlicher Typsysteme müssen anwenderseitig überbrückt werden. Diese Abbildung zwischen den Variablen der Programmiersprache und den Konstrukten der Anfragesprache führt zu erhöhtem Entwicklungsaufwand in Datenbankanwendungen.

Die wesentlichen Problembereiche bei der Nutzung relationaler Datenbanktechnologie für Nicht-Standard-Anwendungen können nach [SAAKE et al., 1997] zu strukturellen und dynamischen Aspekten zusammengefasst werden (vgl. Tab. 3.1). Die größten Probleme entstehen dadurch, dass das Relationenmodell zu wenig Semantik unterstützt. Dadurch ist man zur aufwendigen und fehleranfälligen Abbildung der Anwendungssemantik auf die Semantik des Relationenmodells gezwungen.

Problembereiche	
strukturelle Aspekte	semantische Modellierungskonzepte
	komplexe Datenstrukturen
	komplexe Integritätsbedingungen
	anwendungsspezifische Datentypen
	Identifikation
	navigierende Zugriffe
dynamische Aspekte	Verhaltensbeschreibung
	Versionen
	erweiterte Transaktionsmodelle
Impedance Mismatch	Kopplung mit Programmiersprache

Tab. 3.1: Problembereiche relationaler Technologie

[LONEY et al., 2003] nennen darüber hinaus Risiken, die der relationale Datenbankentwurf im Allgemeinen mit sich bringt.
Das relationale Prinzip ist relativ leicht verständlich, wodurch quasi aus dem Nichts Experten produziert werden, die zwar über viel Selbstvertrauen aber über wenig Erfahrung beim Aufbau realer Anwendungen verfügen. Die mittlerweile zur Verfügung stehenden Werkzeuge verleiten Entwickler dazu, sich sofort in die Modellierung zu stürzen und den Tests nur noch wenig Beachtung zu schenken.
Ein Artefakt aus historischen Applikationsentwicklungen sind Codes oder Abkürzungen, die eingesetzt wurden, um Plattenplatz zu sparen und die langsamen Maschinen von einst nicht zusätzlich zu belasten. Bei

der Einführung relationaler Datenbanksysteme wurden diese zumeist einfach übernommen, wodurch die Leistungsfähigkeit und Funktionen der relationalen Datenbanken praktisch vergeudet wurden. Oberstes Ziel beim Datenbank-Design sollte jedoch grundsätzlich darin bestehen, die Anwendung so verständlich und die Bedienung so einfach wie möglich zu gestalten, sprich Sprache statt Codes einzusetzen, zumal eine wirtschaftliche Rechtfertigung für Codes seit Jahren überholt ist.
Neben der ersten Normalform (1NF) als zwingende Voraussetzung zur Nutzung eines RDBMS ist zwar die Modellierung in höheren Normalformen wünschenswert, weil hierdurch Redundanzen vermieden werden. Der Grad der Normalisierung im Zuge des Analyseprozesses zur Beseitigung anormaler, dysfunktionaler Beziehungen hängt primär aber davon ab, ob eine weitergehende Kategorisierung bzw. die dabei entstehenden neuen Datenelemente wirklich einen zusätzlichen Nutzen für eine spezielle Anwendung bewirken. Mit dem Dekompositions- und Synthesealgorithmus sowie dem Sichtintegrationsverfahren existieren nach [HEUER, 1997] zwar formale Algorithmen um diese Normalformen zu erreichen. Dennoch können häufig Attribute unterschiedlicher Entitäten nicht getrennt werden oder es müssen umgekehrt Attribute völlig getrennter Entitäten in ein Relationenschema aufgenommen werden.
Im nun folgenden Abschnitt werden die wesentlichen, in SQL:1999 eingeführten objektrelationalen Modellierungskonzepte am Beispiel der zum Nationalpark Bayerischer Wald vorhandenen Datenbasis erläuert. Die Beschreibung der Realisierung objektrelationaler GIS-Anwendungen für den Nationalpark in Abschnitt 5.3 folgt dagegen dem SQL-Dialekt des der Umsetzung zugrunde liegenden ORDBMS Oracle 9i.

3.4 Objektrelationale Erweiterungen in Standard SQL:1999

Objektrelationale Datenbanksysteme sind nach [MEIER et al., 2000] Datenbanksysteme, die von einem relationalen Datenmodell ausgehen und objektorientierte Erweiterungen anbieten. [ABBEY et al., 2000] sprechen von einer *hybriden Datenbank*, die zwischen dem relationalen und objektorientierten Modell liegt. Die Datenhaltung erfolgt weiterhin primär in Tabellen, eventuell um Tabellenspalten mit erweiterten Datentypen ergänzt, und die Sprachschnittstelle folgt dem Sprachstandard SQL:1999 bzw. SQL:2003. Bei objektrelationalen Datenbanksystemen wird dem objektorientierten Paradigma oft nur teilweise entsprochen. Zudem weichen die SQL-Dialekte der auf dem Markt verfügbaren ORDBMS von z.B. Oracle, IBM DB2, Informix oder Postgres derzeit noch erheblich von Standard-SQL ab. Die Beispiele der im Folgenden vorgestellten objektrelationalen Konstrukte und deren Diskussion basieren daher ausschließlich auf Standard-SQL. Eine vollständige Betrachtung unterschiedlicher SQL-Dialekte und deren Vergleich mit SQL:1999 und SQL:2003 gibt [TÜRKER, 2003].

Das wesentliche Ziel der objektrelationalen Erweiterungen von SQL ist es, mehr Anwendungssemantik in die Datenbank einzubringen und damit Datenbankfunktionalität für anwendungsspezifische Datentypen und Funktionen verfügbar zu machen. Das DBMS soll das Wissen über die Semantik der Datentypen und Funktionen beispielsweise zur Optimierung von Anfragen und Synchronisation von Transaktionen ausnutzen.

SQL:1999 erweitert das relationale Datenbankmodell von SQL-92 insbesondere um:

- **neue Basisdatentypen** *(BOOLEAN, BLOB und CLOB)*
- **unbenannte Typkonstruktoren** *(ROW, ARRAY und REF)*
- **benannte Typkonstruktoren zur Definition von benutzerdefinierten Datentypen** *(Distinct Typen)* **und strukturierten Typen** (Objekttypen) **mit Subtypenbildung**
- **typisierte Tabellen mit Subtabellenbildung**
- **typisierte Sichten mit Subsichtenbildung**

Daneben enthält SQL:1999 diverse Erweiterungen, die unabhängig vom objektrelationalen Datenmodell sind und insbesondere benutzerdefinierte Routinen, Trigger sowie neue Anfrageprädikate umfassen. Eine ausführliche Beschreibung der wesentlichen modellunabhängigen Erweiterungen von SQL:1999 gibt [MATTOS, 2000].

3.4.1 Erweiterung des Typsystems um neue Basisdatentypen

Im SQL:1999-Standard, an den die DDL, DML und die Sprachkonstrukte für die Datenanfrage aller auf dem Markt befindlichen ORDBMS angelehnt sind, wurden weitere Basisdatentypen gegenüber SQL:1992 festgelegt. Zudem wurden die bestehenden Datentypen teilweise verändert. So wurde beispielsweise die interne Repräsentation der BOOLEAN-Werte im Standard festgelegt.

BOOLEAN unterstützt eine dreiwertige Logik mit den Wahrheitswerten *TRUE*, *FALSE* und *UNKNOWN*. Für boolesche Werte sind die Operatoren *NOT*, *AND*, *OR* und *IS* sowie die gängigen Vergleichsprädikate wie $<$ oder $<=$ zugelassen.

SQL:1999 stellt zwei Datentypen zur direkten Unterstützung großer Objekte (Large Objects (LOBs)) zur Verfügung. **Binary Large Object (BLOB)** ist ein Datentyp, um große Binärdaten wie Bilder direkt in der Datenbank abzulegen. Diese Objekte besitzen flexible Datenbankfunktionen zur Selektion und Manipulation, damit diese nicht auf dem Client-System bearbeitet werden müssen. Im Gegensatz zu den Datentypen *LONG* bzw. *LONG RAW* (speichern textuelle bzw. binäre Daten bis zu einer Größe von 2 GB) kann eine Tabelle mehrere LOB-Datentypen aufnehmen. Das ORDBMS unterstützt über die normalen Datenbankfunktionen die Datenintegrität und den gleichzeitigen Zugriff auf mehrere LOB-Spalten.

Character Large Object (CLOB) ist ein Datentyp, der für große alphanumerische Daten wie lange Texte verwendet werden kann. Auch für diesen Datentyp stehen spezielle Datenbankfunktionen (insbesondere Textsuche und Textmanipulation) zur Verfügung.

3.4.2 Erweiterung des Typsystems um unbenannte Datentypen

SQL:1999 führte das Konzept der *unbenannten Typkonstruktoren* ein. Ist die Struktur eines kollektionswertigen Typs in einer Tabellendefinition festgelegt, lassen sich später durch Aufruf des Tupelkonstruktors Tupel genau dieses Typs erzeugen. Die Wiederverwendbarkeit dieser Typen für die Definition weiterer Tabellen und übergeordneter Typen ist bei diesem Konzept aufgrund der nicht möglichen Namenszuweisung eingeschränkt. Hierfür muss auf (benannte) benutzerdefinierte Datentypen (vgl. Abschnitt 3.4.3) zurückgegriffen werden.
Es stehen folgende Typkonstruktoren zum Bilden von konstruierten, komplex geschachtelten Datentypen zur Verfügung:

- ROW Typen (Tupeltypkonstruktor)
- ARRAY Typen (Arraytypkonstruktor)
- REF Typen (Referenztypkonstruktor)

ROW Typen sind aus einer Reihe atomarer oder benutzerdefinierter Datentypen zusammengesetzt und können unmittelbar den Spalten einer Tabelle zugeordnet werden. Im Gegensatz zu Standard-SQL erlauben die meisten ORDBMS die Benennung von ROW Typen, was in deren Wiederverwendbarkeit resultiert. Diese Form von eigener Standardisierung beim Datenbankentwurf dient somit insbesondere der Konsistenzsicherung in der gesamten Datenbank.

```
alter table INVENTURKREIS add column inventurpunkt ROW (
      gk_rechts    DECIMAL (9,2),
      gk_hoch      DECIMAL (9,2),
      wgs84_x      DECIMAL (9,2),
      wgs84_y      DECIMAL (8,2),
      wgs84_z      DECIMAL (9,2),
      bemerkung    VARCHAR (50)
      );
```

Tupel bzw. Instanzen des entsprechenden Tupeltyps lassen sich dann einfach mit dem Tupelkonstruktor erzeugen:

```
ROW (4521732.60, 5411113.20, 4086094.60, 997435.90, 4780174.27, 'GPS-Messung 2003')
```

Der Zugriff auf einzelne Tupelfelder erfolgt durch den Punktoperator:

```
select inventurpunkt.gk_rechts
from   INVENTURKREIS;
```

Zwei Tupel sind mittels der sechs Vergleichsprädikate $<$, $<=$, $=$, $<>$, $>=$, $>$ vergleichbar, wenn sie dieselbe Anzahl an Feldern besitzen und alle Felder paarweise vergleichbare Datentypen aufweisen. Die Auswertung der Vergleichsoperatoren erfolgt paarweise und liefert als mögliche Ergebnisse *TRUE*, *FALSE* oder im Fall einer Nullwertbelegung eines Tupelfeldes *UNKNOWN*.

ARRAY Typen können mehrere Elemente aller verfügbarer Datentypen aufnehmen, solange dadurch keine explizit oder implizit geschachtelten Arrays entstehen. In SQL:2003 wurde diese Einschränkung aufgehoben (vgl. Abschnitt 3.5.1). Arrays (und Rows) als kollektionswertige Spalten setzen die Einschränkung der ersten Normalform (1NF) bei RDBMS ausser Kraft. 1:n-Beziehungen können unmittelbar in einer Tabelle abgebildet werden, wobei für n vorab eine Obergrenze festzulegen ist. Die Bestandsteilflächennummern eines Bestandes könnten dann wie folgt abgelegt werden:

```
alter table BESTAND add column
      bestandsteilflnr    VARCHAR (2)  ARRAY [6];
```

Die Anfrage

```
select bestandsteilflnr [2] from BESTAND;
```

gibt dann das zweite Element der arraywertigen Spalte *bestandsteilflnr* aus.

Geschachtelte Relationen dieser Art (NF^2-Relationen) werden als *PNF-Relationen* (Partitioned Normal Form) bezeichnet. Darunter werden jene NF^2-Relationen zusammengefasst, die sich entschachtelt immer durch eine äquivalente 1NF-Relation darstellen lassen. SQL:1999 sieht zum Entschachteln solcher PNF den *UNNEST-Operator* vor, mit dem sich Arrays in eine Tabelle transformieren lassen.

Neben Array Typen bietet z.B. Informix im Gegensatz zu Standard-SQL und anderen SQL-Dialekten als **weitere Kollektionstypen** *SET*, *LIST* und *MULTISET* an, die entgegen Arrays keine bestimmte Obergrenze für die 1:n-Kardinalität aufweisen. Sie können nach [TÜRKER, 2003] Basisdatentypen und komplexe Datentypen aufnehmen:

- SET (Mengentypkonstruktor zur Erzeugung von ungeordneten Listen ohne Duplikate)
- MULTISET (Multimengentypkonstruktor zur Erzeugung von ungeordneten Listen mit Duplikaten)
- LIST (Listentypkonstruktor zur Erzeugung von geordneten Listen ohne Duplikate)

MULTISET ist im Gegensatz zu den Mengentypkonstruktoren *SET* und *LIST* in SQL:2003 verfügbar (vgl. Abschnitt 3.5.1).

REF Typen werden mit dem Referenztypkonstruktor erzeugt. Über deren Instanzen (Referenzen) wird auf die Objekte eines referenzierten (strukturierten) Typs verwiesen. Eine Referenz kann später nur dann dereferenziert (aufgelöst) werden, wenn der zugehörige Referenztyp mit einer *Scope-Klausel* versehen wurde. In Standard-SQL muss der Wertebereich (Scope) einer Referenzspalte (im Gegensatz zu beispielsweise Oracle) zwingend auf eine einzelne Tabelle beschränkt werden, um die Referenzspalte später dereferenzieren zu können.

```
alter table FORSTAMT add column
      forstdirektion REF (FORSTDIREKTION_TY)
      scope forstdirektion;
```

Der Aufbau von Referenzen erfolgt dann durch den Referenzkonstruktor, indem einer Referenzspalte eine Instanz des entsprechenden Referenztyps (OID) zugewiesen wird. Die OID (Object Identity) ist ein vom ORDBMS automatisch bei der Erzeugung eines Objekts generierter, systemweit eindeutiger Surrogat-Wert, der nie geändert wird und solange Gültigkeit besitzt, wie auch das Objekt in der Datenbank existiert (Persistenz):

```
update FORSTAMT
set forstdirektion = (select f.oid
                      from forstdirektion f
                      where f.name = 'Niederbayern-Oberpfalz')
where name = 'Zwiesel';
```

Der *Pfeiloperator* ermöglicht das Verfolgen einer Referenz (Traversieren) in einem Pfadausdruck, der *DEREF-Operator* liefert den vollständigen Wert eines referenzierten Objekts:

```
select deref (forstdirektion) from FORSTAMT;
```

Durch OID und REF lassen sich somit Beziehungen zwischen Objekten explizit modellieren. Referenzen sind im Gegensatz zu Fremdschlüsseln des Relationenmodells streng typisiert. Eine Referenzspalte verweist auf einen Wert des entsprechenden Referenztyps, eine Fremdschlüsselspalte hingegen kann denselben Wert haben, wie eine andere beliebige Spalte einer Tabelle. Die beiden Modellierungskonzepte für Objektbeziehungen lassen sich nach [SAUER, 1998] wie folgt vergleichen:

Kriterium	**Zeilenreferenz**	**Primär-/Fremdschlüsselbeziehung**
Art der Beziehung	Explizit durch Referenzen. Die Referenzen einer Spalte können auf verschiedene Tabellen zeigen.	Beziehung wird dynamisch durch Wertevergleich (JOIN) rekonstruiert.
Kardinalitäten	1:1 und n:1 (1:n und n:m durch Kollektions-Typen)	1:1, 1:n und n:1 (n:m durch Zwischentabellen)
Unabhängigkeit von logischen Datenstrukturen	gering	hoch
Unabhängigkeit von Datenänderungen	Bei Schlüsselwertänderungen bleiben Referenzen gültig	Bei Schlüsselwertänderungen müssen alle Werte, auf die referenziert wird, geändert werden
Referentielle Integrität	Typprüfung und Scope-Anweisung verhindern ungültige Referenzen; Referenzen auf gelöschte Zeilen werden beim Zugriff erkannt	Referentielle Integrität kann jederzeit vom DBS sichergestellt werden
Zugriffsoptimierung	Durch Indizes und durch die in der Zeilenreferenz enthaltene interne Satzadresse	Durch einen Index
Besondere Eignung	Komplexe Beziehungsgeflechte	Einfache Beziehungen

Tab. 3.2: Vergleich zwischen Zeilenreferenz und Primär- / Fremdschlüsselbeziehung

3.4.3 Erweiterung des Typsystems um benannte, benutzerdefinierte Datentypen

In SQL:1999 wurde ebenfalls festgelegt, welche Möglichkeiten der Benutzer hat, eigene benutzerdefinierte Datentypen anzulegen, die namentlich bezeichnet und somit tabellen- und typübergreifend wieder verwendet werden können. Hierdurch wird eine sehr große Anzahl von Anwendungsfällen unterstützt, ohne dass der Datenbankanbieter hierfür jeden erforderlichen Datentyp implementieren muss. Die Anzahl der auf niedrigster Ebene verfügbaren Typen bleibt dadurch überschaubar.

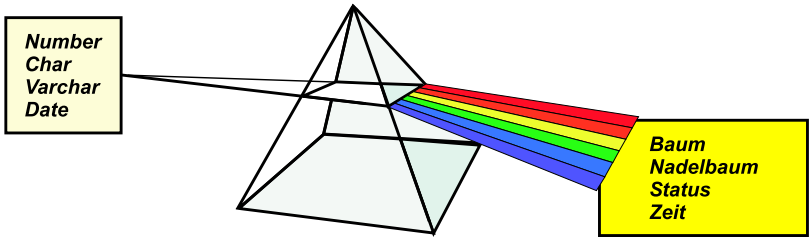

Abb. 3.3: Benutzerdefinierte Datentypen zur Definition anwendungsnaher Strukturen

3.4.3.1 Distinct Typen

Die elementarste Form eines *User Defined Type (UDT)* ist der **DISTINCT Typ**. Hierbei können für bereits existierende Basisdatentypen Kopien unter einem anderen Namen abgelegt werden. Verschiedene Argumente, die intern gleich dargestellt werden, können durch die Verwendung von Distinct Typen von einer fehlerhaften Benutzung geschützt werden. Als Beispiel sei die explizite Zuweisung eines Distinct Typen für Vorratsfestmeter

(Vfm) bzw. Erntefestmeter (Efm) aufgeführt, um bei Anfragen einen semantisch unzulässigen, versehentlich durchgeführten Vergleich von Holzmassen unterschiedlicher Gattungen auszuschließen. Vorratsfestmeter stellen das Holzvolumen stehender Bäume dar, bei Erntefestmetern sind dagegen Ernteverluste bereits eingerechnet:

```
create type VORRATSFESTMETER_TY as DECIMAL (5,2) final;
create type ERNTEFESTMETER_TY as DECIMAL (5,2) final;
```

Eine derartige Anfrage wird bei der Nutzung eigener Distinct Typen von der Datenbank infolge nichtidentischer Datentypen beim Einsatz eines Vergleichsoperators korrekterweise zurückgewiesen. Ein Vergleich zwischen einem Wert des Distinct Typen und einem Wert des dazugehörigen Quelltyps ist dagegen möglich, weil mit der Definition des Distinct Typen automatisch auch entsprechende Konvertierungs- und Cast-Funktionen generiert werden.

Das Arbeiten mit Distinct Typen kann aufgrund der strengen Typisierung zum Teil umständlich sein. So sind konsequenterweise keine arithmetischen Operatoren für Distinct Typen definiert, deren Quelltyp ein numerischer Datentyp ist. Im vorliegenden Fall ist die Addition zweier Vorratsfestmeter-Werte somit nicht erlaubt und kann nur durch eine (bidirektionale) Umwandlung zwischen Distinct Typen und Quelltypen mittels der Cast-Funktionen erreicht werden. Zudem sind Distinct Typen durch die Pflicht-Klausel *final* in Standard-SQL von einer Subtypenbildung ausgeschlossen.

3.4.3.2 Strukturierte Typen, Methoden und Typhierarchien

Abstract Data Types (ADT) erlauben neben der Neubenennung oder der Kombination von Basisdatentypen auch die Definition von Methoden und die Kapselung der internen Darstellung der Werte im Sinne der Objektorientierung. Der Zugriff auf die Attribute eines so erzeugten Objekts erfolgt dann ausschließlich über Methoden.

Abstrakte Datentypen stehen in SQL:1999 in Form von sog. **strukturierten Typen** zur Verfügung. Ein strukturierter Typ ist ein benutzerdefinierter Datentyp, der im Wesentlichen einem Objekttyp mit Attributen und Methoden entspricht, dessen Instanzen jedoch über keine inhärente Objektidentität (OID) verfügen, wie dies bei Objekttypen in Objektdatenbanken der Fall ist. In SQL:1999 können Instanzen eines strukturierten Typs jedoch mit einem OID-Attribut assoziiert und somit durch Referenzattribute referenziert werden, wenn sie Zeilen einer *typisierten Tabelle* darstellen. Damit ein strukturierter Typ als Typ einer Tabelle verwendet und eine OID-Generierung für seine Instanzen zugelassen werden kann, muss er explizit eine OID-Typspezifikation erhalten, die jedoch nicht nur systemverwaltete sondern auch benutzerdefinierte oder aus einer Attributwertekombination abgeleitete OIDs zulässt. Aus einer objektorientierten Sichtweise widersprechen aber sowohl eine benutzerdefinierte als auch eine abgeleitete OID-Generierung dem Grundgedanken einer unabhängigen, systemverwalteten Objektidentität. Zudem ist in SQL:1999 der Eindeutigkeitsbereich einer OID auf nur eine Tabellenhierarchie beschränkt und somit keine „echte“ OID im Sinne der Objektorientierung. Schließlich wird die Wiederverwendung von OIDs in Standard-SQL nicht explizit ausgeschlossen und OIDs lassen sich lediglich für Zeilen einer typisierten Tabelle, nicht aber für als Spaltenwerte eingesetzte Objekte generieren, die somit auch nicht referenzierbar sind (vgl. [TÜRKER, 2003]).

Ein strukturierter Typ kann als Subtyp eines anderen strukturierten Typs definiert werden. Jeder Subtyp hat genau einen direkten Supertyp, von dem er alle Attribute und Methoden erbt, somit ist eine Mehrfachvererbung ausgeschlossen. Ein strukturierter Typ, der keinen Supertyp besitzt, wird als Wurzeltyp bezeichnet, da er die Wurzel einer möglichen Typhierarchie bildet. Durch die Wiederverwendung strukturierter Typen als Attributtypen von anderen strukturierten Typen entstehen beliebig komplexe Datentypen, womit eine natürlichere Abbildung der Realweltobjekte auf Datenbankobjekte unterstützt wird.

```
create type GK_TY as (
       gk_rechts    DECIMAL (9,2),
       gk_hoch      DECIMAL (9,2)
       )
       not final;

create type INVENTURPUNKT_TY (
       invpunkt_id    DECIMAL (5),
       gk_koord       GK_TY,
       bemerkung      VARCHAR (50)
       )
       not final
       ref is system generated;
```

Bei der Erzeugung von Subtypen werden wie erwähnt alle Attribute und Methoden vom Supertyp geerbt, wobei Methoden überschrieben und eigene Methoden und Attribute definiert werden können (Spezialisierung). Ein strukturierter Typ kann mehrere Subtypen enthalten.

Die Vererbungshierarchie kann etwa am Beispiel der Spezialisierung von Baumarten bei der Modellierung von Inventurdaten im Nationalpark Bayerischer Wald demonstriert werden:

```
create type NADELBAUM_TY under BAUM_TY as (
       id                     INTEGER,
       baumalter              INTEGER,
       beschreibung           CLOB(50K),
       gattungen_verwandt     REF(NADELBAUM_TY) ARRAY [30],
       derbholz               VORRATSFESTMETER
       )
       not final;

create type FICHTE_TY under NADELBAUM_TY as (
       spezies    VARCHAR (30)
       )
       not final;

create type LAUBBAUM_TY under BAUM_TY as (
       id                     INTEGER,
       spezies                VARCHAR (30),
       beschreibung           CLOB(50K),
       gattungen_verwandt     REF(LAUBBAUM_TY) ARRAY [30]
       )
       not final;
```

Durch das Konzept der *Substituierbarkeit*, bei dem eine Instanz eines Subtyps in jedem Kontext benutzt werden kann, in dem eine Instanz eines Supertyps erwartet wird, kann ein Objektwert auch als Instanz eines Supertyps fungieren. Dazu ist eine temporäre Typanpassung entlang der Typhierarchie nötig, wodurch Methoden jeweils auf dem aktuell deklarierten Typ eines Objektwertes aufgerufen werden können.

METHODEN sind spezielle Funktionen, die an einen strukturierten Typ gebunden sind, anders als bei SQL-Funktionen erfolgt die *Deklaration* und *Definition bzw. Implementierung* einer Methode jedoch strikt getrennt. Methoden werden innerhalb der Definition eines strukturierten Typs deklariert und sind wie Prozeduren und

Funktionen im Sinne der Objektorientierung *überladbar*. Ein strukturierter Typ kann mehrere Methoden mit demselben Namen haben, sog. überladene Methoden müssen sich jedoch in mindestens einem Parametertyp oder in der Anzahl der Parameter unterscheiden, damit eine eindeutige Auswahl der verschiedenen Methodenimplementierungen möglich ist.

```
create type NADELBAUM_TY under BAUM_TY as
       (...)
       not final
       method VOLUMENBERECHNUNG() returns VORRATSFESTMETER;
       method VOLUMENBERECHNUNG(hoehe HOEHE, bhd BHD)
                                returns VORRATSFESTMETER;
```

Methoden dürfen im Subtyp überschrieben werden, ein Subtyp kann also eine Methode enthalten, welche dieselbe Signatur hat wie eine geerbte Methode. Zur Laufzeit wird die Implementierung einer überschriebenen Methode durch *late binding* aufgerufen. Die Auswahl der Implementierung hängt vom aktuell deklarierten Typ des Objektwertes ab, auf dem die Methode aufgerufen wird.

```
create type FICHTE_TY under NADELBAUM_TY as
       (...)
       not final
       overriding method VOLUMENBERECHNUNG(hoehe HOEHE, bhd BHD)
                                           returns VORRATSFESTMETER;
```

Die Signatur der *Methodenimplementierung* muss mit der Signatur der Methodendeklaration übereinstimmen. Der Methodenrumpf besteht aus einer SQL-Anweisung, die aus mehreren DDL-, DML-, sowie Anfrageanweisungen zusammengesetzt sein kann.
Im Folgenden ist die unterschiedliche Berechnung des Holzvorrats nach der Implementierung von überschreibenden Methoden illustriert. Der in der zweiten Methode gewählte Berechnungsansatz geht auf die Derbholz-Volumenberechnung von [FRANZ, 1971] zurück, nach der sich der Holzvorrat in Abhängigkeit von Baumart, Brusthöhendurchmesser und Baumhöhe approximieren lässt:

```
create method VOLUMENBERECHNUNG (hoehe HOEHE, bhd BHD)
       returns VORRATSFESTMETER
       for NADELBAUM_TY
       return VORRATSFESTMETER ((0.5 * cast(BHD as DECIMAL (4,2)))^2
                               * 3.14159 * cast(HOEHE as DECIMAL (4,2)));
```

```
create method VOLUMENBERECHNUNG (hoehe HOEHE, bhd BHD)
       returns VORRATSFESTMETER
       for FICHTE_TY
       begin
            declare a1 FLOAT (10), a2 FLOAT (10), a3 FLOAT (10);
            declare c11 FLOAT (4), c21 FLOAT (4), c31 FLOAT (4);
            declare hform FLOAT (10), v FLOAT (10);
            set c11 = -3.60E+00;
            set c21 = 1.80E+00;
            set c31 = -2.88E-01;
            declare c12 FLOAT (4), c22 FLOAT (4), c32 FLOAT (4);
            set c12 = 1.06E+00;
            set c22 = -1.29E+01;
```

```
        set c32 = 3.53E-02;
        declare c13 FLOAT (4), c23 FLOAT (4), c33 FLOAT (4);
        set c13 = 1.42E-01;
        set c23 = -5.83E-02;
        set c33 = 4.60E-03;
        set a1 = c11 + c21 * ln(cast(BHD as DECIMAL (4,2)))
                           + c31 * (ln(cast(BHD as DECIMAL (4,2))))^2;
        set a2 = c12 + c22 * ln(cast(BHD as DECIMAL (4,2)))
                           + c32 * (ln(cast(BHD as DECIMAL (4,2))))^2;
        set a3 = c13 + c23 * ln(cast(BHD as DECIMAL (4,2)))
                           + c33 * (ln(cast(BHD as DECIMAL (4,2))))^2;
        set hform = exp(a1 + a2 * ln(cast(HOEHE as DECIMAL (4,2)))
                       + a3 * (ln(cast(HOEHE as DECIMAL (4,2))))^2;
        set v = (cast(BHD as DECIMAL (4,2)))^2 * (3.14159 / 40000) * hform;
        if (v > 2) THEN
           signal sqlstate '75001';
           set Message_Text = 'Derbholzbeitrag über Durchschnitt';
        end if;
        self.derbholz = VORRATSFESTMETER(v);
        RETURN self.derbholz;
    end;
```

Methoden von ADT können in vielen ORDBMS längst auch in C++ oder Java programmiert werden, wodurch die Grundlage zur Unterstützung komplex strukturierter Datenbestände wie Geometrie-, Multimedia-Daten oder Zeitreihen mit eigenen Vergleichs-, Sortierungs- und Manipulationsoperatoren geschaffen wurde sowie das Angebot an Indizierungsmechanismen erweitert werden konnte ([STONEBRAKER et al., 1999]).

3.4.4 Typisierte Tabellen und Tabellenhierarchien

Typisierte Tabellen bilden neben den strukturierten Typen den Kern der objektrelationalen Erweiterungen von SQL. Eine typisierte Tabelle entspricht im Wesentlichen einer Objekttabelle, deren Typ durch einen strukturierten Typ festgelegt wird. Die Spalten einer typisierten Tabelle ergeben sich aus den Attributen des strukturierten Typs, wobei die erste Spalte immer das hier zwingend erforderliche OID-Attribut enthält. Die Zeilen einer typisierten Tabelle entsprechen Objekten des strukturierten Typs. Typisierte Tabellen unterscheiden sich von Tupeltabellen durch folgende Eigenschaften:

- Die Zeilen einer typisierten Tabelle sind über die OID-Spalte referenzierbar.
- Auf den Zeilen einer typisierten Tabelle sind Methoden aufrufbar.
- Eine typisierte Tabelle kann als Subtabelle einer anderen typisierten Tabelle definiert werden.

Entsprechend der Bezeichnung des ersten Typs einer Typhierarchie bei strukturierten Typen werden typisierte Tabellen, die keine Supertabellen besitzen, als *Wurzeltabellen* bezeichnet. Eine solche Tabelle bildet die Wurzel einer möglichen Tabellenhierarchie.

```
create table NADELBAUM of NADELBAUM_TY
       (
            ref is oid system generated,
            baumalter with options not null,
            gattungen_verwandt with options scope nadelbaum
       );
```

Die OID-Spalte wird beim Einfügen eines Objekts in die typisierte Tabelle mit einem eindeutigen und unveränderbaren Wert versehen. Die Scope-Klausel bestimmt die typisierte Tabelle, auf deren Objekte eine Referenzspalte verweisen darf und ist daher unabdingbar für Referenzspalten, da sie sonst nicht dereferenzierbar sind.

Jede *Subtabelle* hat genau eine direkte Supertabelle, wodurch eine Mehrfachspezialisierung ausgeschlossen ist. Dabei muss der Typ der Subtabelle immer ein direkter Subtyp des Typs der Supertabelle sein, jedoch müssen die Typ- und Tabellenhierarchien nicht eins-zu-eins übereinstimmen (vgl. Abb. 3.4, Entkopplung der Typ- und Tabellenhierarchien):

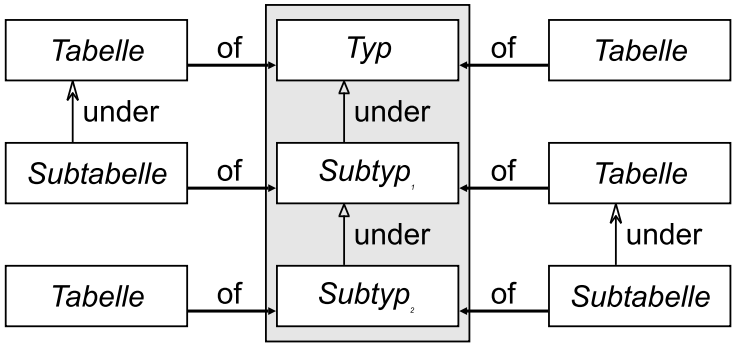

Abb. 3.4: Zusammenhang zwischen Typ- und Tabellenhierarchie

Eine Subtabelle erbt die OID-Spalte und alle Spaltenoptionen der Supertabelle. Sie kann darüber hinaus eigene Spaltenoptionen erhalten:

```
create table FICHTE of FICHTE_TY under NADELBAUM
       (
            spezies with options not null,
            check(baumalter < 20)
       );
```

Durch das Konzept der Substituierbarkeit (vgl. Abschnitt 3.4.3.2) kann die Spalte einer Tupeltabelle Instanzen verschiedener strukturierter Typen (einer Typhierarchie) enthalten (Substituierbarkeit auf Spaltenebene). Das Substituierbarkeitsprinzip gilt aber auch für Referenztypen, wodurch eine typisierte Tabelle OIDs von verschiedenen Referenztypen aufnehmen kann. Die Zeilen repräsentieren verschiedene Arten von Objekten, auf denen je nach Spezialisierung des zugrundeliegenden strukturierten Typs unterschiedliche Methoden aufgerufen werden können (Substituierbarkeit auf Zeilenebene). Jedoch sind die subtypspezifischen Attribute der zugehörigen Instanzen des jeweiligen Subtyps nicht in der Tabelle darstellbar, denn die Spalten der (typisierten) Tabelle sind durch die Attribute des Tabellentyps statisch fixiert. Die Umsetzung von Objekten und deren Substituierbarkeit ist beispielsweise in Oracle durch das abstraktere Objektmodell mit inhärenten OIDs und der Möglichkeit, die Substituierbarkeit der Objekte einer typisierten Tabelle (Objekttabelle) einzuschränken oder ganz auszuschließen, wesentlich konsequenter umgesetzt. Entgegen SQL:1999 unterstützt Oracle jedoch keine Subtabellenbildung.

3.4.5 Typisierte Sichten und Sichtenhierarchien

Typisierte Sichten basieren analog zu typisierten Tabellen auf einem strukturierten Typ. Eine typisierte Sicht entspricht im Wesentlichen einer Objektsicht, deren Typ durch einen strukturierten Typ festgelegt wird. Analog zu typisierten Tabellen stellt die erste Spalte einer typisierten Sicht immer die OID-Spalte dar. Die restlichen Spalten entsprechen den Attributen des festgelegten strukturierten Typs. Die Unterschiede zwischen typisierten Sichten und Tupelsichten verhalten sich analog zu denen zwischen typisierten Tabellen und Tupeltabellen (vgl. Abschnitt 3.4.4):

- Die Zeilen einer typisierten Sicht sind über die OID-Spalte mittels Referenzspalten referenzierbar.
- Auf den Zeilen einer typisierten Sicht sind Methoden aufrufbar.
- Eine typisierte Sicht kann als Subsicht einer anderen typisierten Sicht definiert werden.

Typisierte Sichten, die keine Supersicht besitzen, werden als Wurzelsichten bezeichnet und bilden die Wurzel einer möglichen Sichtenhierarchie.

Die Anfrage einer typisierten Sicht unterliegt einigen Einschränkungen:

- Der Ergebnistyp des Anfrageausdrucks muss mit dem Typ der Sicht verträglich sein.
- Bei Sichten, die auf einer benutzerdefinierten OID-Generierung basieren, muss die Projektionsliste mit dem Aufruf des Referenzkonstruktors beginnen, d.h. zunächst muss der Wert der ersten Spalte der typisierten Sicht (OID-Spalte) berechnet werden.
- Die FROM-Klausel darf nur *eine* Tabelle bzw. Sicht enthalten. Ist diese Tabelle bzw. Sicht selbst typisiert, so muss sie mit dem Schlüsselwort ONLY auf ihre flache Extension (Wurzel ohne mögliche Subtabellen, - sichten) eingeschränkt werden.

```
create type NADELBAUM_VIEW_TY as
       (
            standort                GK_TY,
            id                      INTEGER,
            baumalter               INTEGER,
            gattungen_verwandt      REF(NADELBAUM_VIEW_TY)
       )
not final
ref using integer;


create view VERJUENGUNG of NADELBAUM_VIEW_TY
       (
            ref is oid user generated,
            gattungen_verwandt with options scope VERJUENGUNG
       )
as select NADELBAUM_VIEW_TY(id),
          standort,
          id,
          baumalter,
          NADELBAUM_VIEW_TY(gattungen_verwandt->id)
from only(NADELBAUM)
where baumalter < 10
with check option;
```

Jede Subsicht hat genau eine direkte Supersicht, wodurch eine Mehrfachvererbung ausgeschlossen ist. Analog zu Subtabellen gilt auch hier die Forderung, dass der Typ der Subsicht ein direkter Subtyp des Typs der Supersicht sein muss. Ferner muss die Tabelle, die der Subsicht zugrunde liegt, eine Subtabelle der Tabelle sein, die der Supersicht zugrunde liegt. Die folgende Abbildung illustriert die Abhängigkeiten zwischen Tabellen- und Sichthierarchien:

Eine Subsicht erbt die OID-Spalte und alle Spaltenoptionen der Supersicht. Sie kann darüber hinaus eigene Spaltenoptionen erhalten:

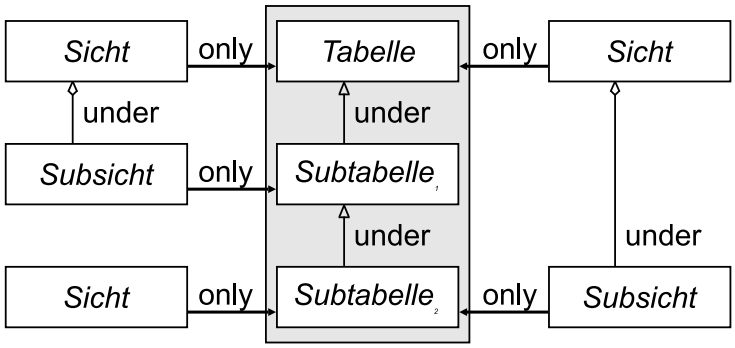

Abb. 3.5: Zusammenhang zwischen Tabellen- und Sichthierarchie

```
create type FICHTE_VIEW_TY under NADELBAUM_VIEW_TY
       (
            zustand     VARCHAR (30))
not final;


create view GESCHWAECHTE_JUNGFICHTEN of FICHTE_VIEW_TY
under VERJUENGUNG
as select FICHTE_VIEW_TY(id),
          standort,
          id,
          baumalter,
          FICHTE_VIEW_TY(gattungen_verwandt->id)
from only(FICHTE)
where zustand = 'geschwaecht';
```

Die in SQL:1999 zum Sprachumfang ergänzten objektrelationalen Konzepte wie benutzerdefinierte Datentypen und Methoden bilden die Grundlage für die Wiederverwendbarkeit von Datenstrukturen und die Implementierung neuer Basisdatentypen zur direkten Verwaltung von Geodaten und XML-Daten in kommerziellen Datenbanken (vgl. Abschnitt 3.6 und 3.5.2). Für weiterführende Ausführungen über die in SQL:1999 verfügbaren DDL-Strukturen zur Datendefinition, Anfragen und Änderungsoperationen sei an dieser Stelle auf [PRIEBE, 2003] verwiesen.

3.5 Neuerungen in Standard SQL:2003

Neben Fehlerkorrekturen sieht SQL:2003 gegenüber SQL:1999 Verbesserungen und Erweiterungen in drei wesentlichen Bereichen vor:
Zunächst existieren weitere Features des objektrelationalen Modells, hier speziell die Möglichkeit zur Erzeugung von ungeordneten Kollektionen mit Duplikaten (vgl. Abschnitt 3.5.1). Ausserdem wurde die Pflichtklausel NOT FINAL bei der Subtypenbildung in SQL:1999 zur optionalen Klausel FINAL in SQL:2003 erweitert, d.h. die Typhierarchie kann nun explizit unterbrochen werden.
Zum Zweiten wurden eine Reihe analytischer Funktionen für OLAP-Anwendungen in den Standard aufgenommen, um erweiterte Methoden zur Ad-hoc-Analyse multidimensionaler Informationen in SQL-Datenbanken zur Verfügung zu stellen.
Die dritte und wohl bedeutendste Neuerung umfasst mit SQL:2003 - Part 14 einen vollständig neuen Sprachumfang zur Abbildung zwischen SQL und XML (vgl. Abschnitt 3.5.2).
Einen übersichtlichen Vergleich der Versionen SQL:1992, SQL:1999 und SQL:2003 mit ihren wesentlichen Neuerungen gibt [PISTOR, 2003]. Für eine explizite Darstellung der Unterstützung dieser Erweiterungen in Oracle 10g sei auf [ORACLE (4), 2003] verwiesen.

3.5.1 Multimengentypkonstruktor

SQL:2003 sieht einen Multimengentypkonstruktor *MULTISET* zur Erzeugung von ungeordneten Kollektionen, die Duplikate zulassen vor. Stattdessen stehen der Mengentypkonstruktor *SET* als auch der Listentypkonstruktor *LIST* (vgl. Abschnitt 3.4.2) in SQL:2003 nicht zur Verfügung. Mittels Multimengen lassen sich etwa sämtliche Inventurpunkte und Bestandsteilfächen eines Bestandes in die Haupttabelle aufnehmen:

```
create table BESTAND (
       bestandskey                  DECIMAL (10),
       inventurpunkte               ROW (gk_rechts    DECIMAL (9,2),
                                         gk_hoch      DECIMAL (9,2),
                                         bemerkung    VARCHAR (50)) MULTISET,
       flaeche                      DECIMAL (8,2),
       bestandsteilflaechen         VARCHAR (2) MULTISET
       );
```

An dieser Stelle sollte noch einmal explizit erwähnt werden, dass SQL:2003 die Einschränkung von SQL:1999, wonach der Arraytypkonstruktor nicht geschachtelt werden kann, aufhebt. Array- und Multimengentypkonstruktoren sind also nun beliebig kombinierbar.

Verfügbar sind ferner einige *Multiset-Operatoren* zum direkten Vergleich der Elemente zweier Multimengen, die etwa die Vereinigung (UNION), die Differenz (EXCEPT) bzw. den Durchschnitt (INTERSECT) der Ausgangsmengen in einer neuen Multimenge zurückgeben. Mittels DISTINCT lassen sich dabei Duplikate aus den Multimengen eliminieren. Der Multimengeninklusionstest mittels SUBMULTISET liefert den Boolean-Wert TRUE, wenn alle Elemente einer Menge M_1 in M_2 enthalten sind.
Zum Entschachteln und Zugriff auf die Elemente einer multimengenwertigen Spalte dient wie bisher bei arraywertigen Spalten der Operator UNNEST:

```
select bestandskey, b.bestandsteilflaeche
from   BESTAND, unnest (bestandsteilflaechen) b (bestandsteilflaeche);
```

3.5.2 SQL/XML zur integrierten Verwendung von SQL und XML

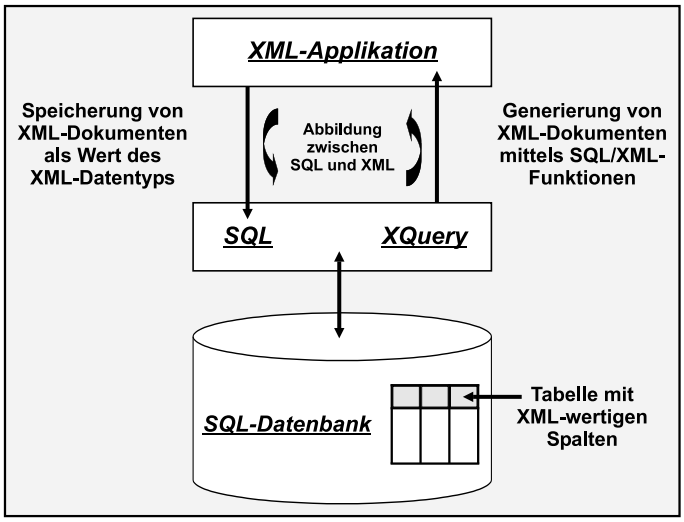

Abb. 3.6: Architektur von SQL/XML

Nachdem sich XML als Standard für den Datenaustausch über das Internet durchgesetzt hat und demnach das Datenvolumen, das als XML-Daten strukturiert wird, stetig gewachsen ist, ist auch der Bedarf gestiegen, XML Daten in einer zuverlässigen, sicheren und skalierbaren Umgebung, sprich in einer Datenbank, zu speichern, zu verwalten und zu verarbeiten.

Von der Industrie werden derzeit zwei Wege eingeschlagen, eine derartige XML-Unterstützung anzubieten. Zum einen werden sogenannte „native“ XML-Datenbanksysteme entwickelt (vgl. [SEEMANN, 2003]), die vollständig auf die Speicherung und Verarbeitung von XML-Daten ausgerichtet sind. Zum anderen werden die zur Speicherung und Verwaltung großer Datenmengen am Markt etablierten RDBMS bzw. ORDBMS um XML-Funktionalität erweitert.

Der Einsatz nativer XML-Datenbanksysteme bietet sich an, wenn ausschließlich XML-Daten zu speichern und zu verarbeiten sind. Soll jedoch auf XML-Daten und SQL-Daten gemeinsam über eine SQL-Schnittstelle zugegriffen werden, muss ein „XML-fähiges“ objektrelationales Datenbanksystem eingesetzt werden, da native XML-Datenbanksysteme i.d.R. keinerlei SQL-Unterstützung anbieten.

Mit SQL/XML wurde ein neuer Teil in SQL:2003 aufgenommen, der die integrierte Verwendung von SQL und XML in einer SQL-Datenbank vorsieht (vgl. Abb. 3.6 nach [TÜRKER, 2003]). Für die Integration von XML-Daten in die Datenbank sowie für die Definition von XML-Dokumenttypen wird explizit die Erweiterbarkeit des Typsystems eines ORDBMS genutzt.

Die Definition von XML-Dokumenttypen durch *XML-Schema*, das im Gegensatz zur *Document Type Definition (DTD)* eine echte Typisierung mit einem umfassenden Typsystem unterstützt, geschieht auf Grundlage von Basisdatentypen und Typkonstruktoren für die Erzeugung von benutzerdefinierten sowie komplexen Datentypen. Neben einem neuen Basisdatentyp *XML* zur Speicherung und Verarbeitung von XML-Werten und -Dokumenten in einer SQL-Datenbank bietet SQL/XML eine Reihe von vordefinierten Funktionen zur Generierung von XML-Werten aus herkömmlichen Tabellen oder anderen XML-Werten. Durch die Einbettung der XML-Anfragesprache *XQuery* in SQL sind Anfragen auf XML-Dokumente möglich.

Aus Sicht der Geoinformatik lässt sich durch den Datentyp XML die fachliche Bedeutung oder Beschreibung eines Geoobjekts in strukturierter Form zusammen mit seiner Objektgeometrie (vgl. Abschnitt 3.6) und seinen direkten Attributen in einer Tabelle verwalten. Ein Geoobjekt stellt nach [BOLLMANN et al., 2001]

> *„eine elementare Einheit zur Modellierung der realen Welt und zur Implementierung in Geoinformationen dar. Ein Geoobjekt wird dabei für jedes Phänomen der realen Welt gebildet, das aus fachlicher Sicht ein hinreichendes Eigenleben führt.“*

Eine ausführliche Diskussion, durch welche Datentypen sich Geoobjekte hinreichend genau beschreiben lassen, gibt [LOTHER, 2003].

Die Verfügbarkeit des Datentyps XML ermöglicht demnach eine geschlossene Objektbeschreibung und eine wesentlich intuitivere Darstellung aller Objekt-relevanten Daten. Tab. 3.3 illustriert dies am Beispiel der FFH-Richtlinie zum Schutz und Erhalt von Lebensarten und Lebensräumen im Nationalpark Bayerischer Wald (vgl. [KIENER et al., 2002]).

```
create table LEBENSRAUMTYPEN (
       lrt_code        VARCHAR (4),
       lrt_name        VARCHAR (80),
       beschreibung    XML,
       geom            ST_POLYGON
       );
```

LEBENSRAUMTYPEN

LRT_CODE	LRT_NAME	BESCHREIBUNG	GEOMETRIE
91D0	Moorwälder	<Beschreibung Datum="19.08.2004"> <Vegetation> <Auspraegung>Birken-Moorwald</Auspraegung> <Auspraegung>Latschen-Moorwald</Auspraegung> <Auspraegung>Waldkiefern-Moorwald</Auspraegung> <Pflanzen> <Zwergstraeucher> <Pflanze>Heidelbeere</Pflanze> <Pflanze>Preiselbeere</Pflanze> <Pflanze>Rosmarinheide</Pflanze> </Zwergstraeucher> </Pflanzen> </Vegetation> <Kennzeichen>kleinstandörtliche Vielfalt</Kennzeichen> <Geologie> <Bodentyp>feucht-nasses Torfsubstrat</Bodentyp> <Naehrstoffhaushalt>gering</Naehrstoffhaushalt> <Wasserhaushalt>hoch</Wasserhaushalt> </Geologie> <Rechtsgrundlage>geschützt nach Art.13, BayNatSchG</Rechtsgrundlage> <Repraesentanzen> <Repraesentanz>südliches Alpenvorland</Repraesentanz> <Repraesentanz>oberpfälzisch-bayerischer Wald</Repraesentanz> </Repraesentanzen> </Beschreibung>	...
9180	Schlucht- und Hangmischwälder (Tilio - Acerion)	<Beschreibung Datum="21.07.2004"> <Vegetation> <Auspraegung>Ahorn-Eschen-Schluchtwald</Auspraegung> <Auspraegung>Blockschuttwald</Auspraegung> <Auspraegung>Blaugras-Winterlinden-Wald</Auspraegung> <Pflanzen> <Kraeuter> <Pflanze>Silberblatt</Pflanze> <Pflanze>Dorniger Schildfarn</Pflanze> <Pflanze>Schwalbenwurz</Pflanze> </Kraeuter> </Pflanzen> </Vegetation> <Standorte> <Standort>kühl-feucht bis frisch</Standort> <Standort>trocken-warm</Standort> </Standorte> <Exposition>Steilhanglagen</Exposition> <Kennzeichen>räumliche Vielfalt an Strukturen</Kennzeichen> <Kennzeichen>üppige Krautschicht</Kennzeichen> <Rechtsgrundlage>geschützt nach Art.13(d), BayNatSchG</Rechtsgrundlage> <Repraesentanzen> <Repraesentanz>südliches Alpenvorland</Repraesentanz> <Repraesentanz>oberpfälzisch-bayerischer Wald</Repraesentanz> <Repraesentanz>Rhön</Repraesentanz> <Repraesentanz>Fränkische Schweiz</Repraesentanz> </Repraesentanzen> </Beschreibung>	...
...			

Tab. 3.3: Erweiterte Beschreibung von Geoobjekten

SQL/XML definiert neben dem neuen Basisdatentyp XML und einigen XML-Funktionen zur Generierung von XML-Dokumenten aus SQL-Daten konkrete Abbildungen zur Transformation der SQL-Daten in XML-Daten. Diese erfolgt durch zwei Dokumente:
Ein XML-Schema-Dokument beschreibt das Schema der Tabelle mit allen Spaltennamen-Typ-Paaren und dem Typ der Tabelle. Ein XML-Dokument gibt den Inhalt der Tabelle, d.h. die Werte der Tabellenzeilen wieder. Grundlage aller Transformation ist ein von SQL/XML definiertes Typ-Mapping, bei dem jeder SQL-Datentyp auf einen bestimmten XML-Schema-Typ abgebildet wird. Der standardisierte Austausch von XML-Werten zwischen dem Datenbanksystem und den XML-Anwendungsprogrammen erfolgt mittels Zeichenketten. Alternativen zu dieser Form des Austauschs, die die jeweiligen Charakteristika der Anwendungsprogramme berücksichtigen, werden ausführlich von [MÜLLER, 2004] diskutiert. Eine übersichtliche Beschreibung der in SQL/XML verfügbaren XML-Funktionen gibt [SCHMIDHAUSER, 2003].

3.6 SQL/MM-Spatial zur Verwaltung von Geodaten in ORDBMS

Auf Grundlage der objektrelationalen Erweiterungen von SQL:1999 entstand mit *SQL/Multimedia* ein separater Standard, der Zusatzpakete für die Verarbeitung von Multimediadaten definiert (vgl. [ANSI/ISO/IEC, 1999]). Dieser Standard setzt sich aus fünf Teilen zusammen, wovon „Part 3: Spatial“ das Anwendungspaket für die Verarbeitung von Vektordaten spezifiziert.
Dieses - im Allgemeinen als *SQL/MM-Spatial* bezeichnete - Erweiterungspaket stellt Datentypen und zugehörige Methoden zum Speichern und Verarbeiten von Geometrie-Objekten in der Datenbank zur Verfügung. Dabei bildet der strukturierte Typ *ST_GEOMETRY* die Wurzel einer Typhierarchie (vgl. Abb. 3.7). Der Typ ST_GEOMETRY ist nicht instanzierbar und legt die gemeinsamen Methoden aller Geometrie-Datentypen fest. Diese Methoden werden gewöhnlich in den Subtypen durch subtypspezifische Implementierungen überschrieben und umfassen:

- Methoden zum Abfragen des Objektzustands (a)
- Methoden, die geometrische Operatoren implementieren (b)
- Methoden, die räumliche Beziehungen ermitteln (c)

```
create type ST_GEOMETRY as
       (
            st_privatedimension             SMALLINT default -1,
            st_privatecoordinatedimension   SMALLINT default 2,
       )
       not instantiable
       not final
       method ST_GEOMETRYTYPE() returns VARCHAR(64),                          (a)
       method ST_INTERSECTION(a ST_GEOMETRY) returns ST_GEOMETRY,             (b)
       method ST_DISTANCE(a ST_GEOMETRY) returns DOUBLE PRECISION;            (c)
```

Der Subtyp ST_CURVE enthält spezifische Methoden, die typisch für Linienobjekte sind, etwa Methoden zum Berechnen der Länge eines Linienzugs oder zum Umwandeln eines Linienzugs in eine Gerade:

```
create type ST_CURVE under ST_GEOMETRY
       not instantiable
       not final
       method ST_LENGTH() returns DOUBLE PRECISION,
       method ST_CURVETOLINE() RETURNS ST_LINESTRING;
```

Die instanzierbaren Subtypen (z.B. ST_LINESTRING) speichern die Koordinatenwerte ihrer Stützpunkte in Attributen vom Typ ARRAY und besitzen ihrerseits wiederum spezifische Methoden, wie etwa die Möglichkeit zum Zugriff auf den n-ten Stützpunkt eines Linienzugs. Die eigentliche Definition der Methode erfolgt wiederum getrennt zur Deklaration (vgl. Abschnitt 3.4.3.2).

Die Produkte der führenden Datenbankhersteller zum Speichern und Verarbeiten von raumbezogenen Daten, wie etwa Oracle Spatial bieten im Vergleich zu SQL/MM-Spatial ähnliche Funktionalität, basieren aber auf firmenspezifischen Funktionen und Objekten. Hybride Indexstrukturen, die in der Regel in Form eines Primär- und Sekundärfilters zur Förderung des effizienten Zugriff auf räumliche Daten implementiert sind, ergänzen die Funktionalität.
Die von OGC erstellte bzw. erweiterte und als ISO 19125-2 bekannte *Simple Features Specification for SQL* ist dem SQL/MM-Spatial Standard sehr ähnlich, was vor allem im Vergleich der Datentypen zum Ausdruck kommt. Im Gegensatz zu SQL/MM-Spatial ist die SFSQL jedoch nicht auf eine objektrelationale bzw. objektorientierte

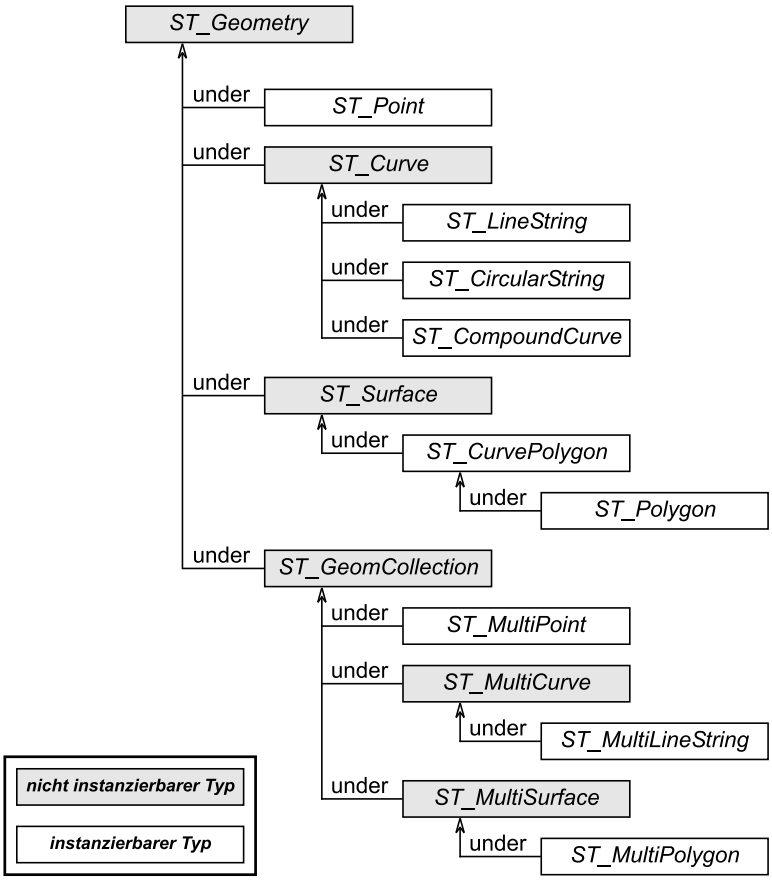

Abb. 3.7: Typhierarchie für Geometrieobjekte

Datenbank angewiesen, d.h. die Geometriedaten müssen nicht als Objekte in der Datenbank abgelegt werden, sondern können auch numerisch oder binär gespeichert werden.

Für weiterführende Vergleiche von SQL/MM-Spatial mit anderen führenden Standards für die Speicherung von Geodaten wie der SFSQL, der schweizer Datenstrukturbeschreibungssprache und Transferformat *INTERLIS* und herstellerspezifischen Implementierungen unter Berücksichtigung ihrer geometrischen Datentypen, Speicherformen, Operationen und Funktionen sei auf [ANDEREGG, 2002] verwiesen. Die Realisierung von objektrelationalen GIS-Anwendungen für den Nationalpark Bayerischer Wald, die in Abschnitt 4.3 ausführlich beschrieben sind, erfolgt auf Grundlage von Oracle 9i Spatial. Das objektrelationale Schema zur Verwaltung von Geometriedaten in Oracle Spatial ist in [ORACLE (2), 2002] umfassend dokumentiert.

3.7 Zusammenfassung und Bewertung

Die Erweiterung relationaler Datenbanksysteme um objektorientierte Strukturen resultiert unmittelbar in der Verfügbarkeit von benutzerdefinierten Datentypen und Methoden zur flexibleren Abbildung komplexer, realer Objekte, insbesondere für Nicht-Standard-Anwendungen. Flexible Mittel zur Typkonstruktion sowie die Definierbarkeit von datentypspezifischen Operatoren erlauben die Wiederverwendbarkeit von häufig wiederkehrenden Datentypen und typischen auf den Werten ausführbaren Operationen einer Anwendung, was sich positiv auf die Konsistenzsicherheit innerhalb der gesamten Datenbank auswirkt. Zudem unterstützt das objektrelationale Datenbankmodell die direkte Abbildung von semantischen Entwurfsinformationen, so dass auf aufwendige Multijoins beim Lesen von z.B. Aggregations- und Kompositionsobjekten verzichtet werden kann.

Für die objektrelationale Anwendungsentwicklung ergeben sich enorme Vorteile durch die Möglichkeit, Objekte persistent in der Datenbank zu speichern. Benutzerdefinierte Datentypen und Routinen können etwa in Java definiert und wie jedes andere SQL-Konstrukt in der Datenbank genutzt werden.

Die Erweiterbarkeit des Typsystems ist Grundlage für die Verfügbarkeit vordefinierter Basisdatentypen zur Verwaltung von Geodaten sowie zur nativen Speicherung von XML-Werten und -Dokumenten.
Räumliche Funktionen und Operatoren werten das ORDBMS von einem Medium für die Speicherung komplex strukturierter Objekte und ihrer Beziehungen zu einem alternativen Werkzeug für die Analyse raumbezogener Daten auf. Nachdem die Beschreibung von Geoobjekten bislang auf die Wiedergabe ihrer geometrischen und attributiven Eigenschaften beschränkt war, erlaubt die Definition von XML-Spalten in Tabellen nunmehr die Vervollständigung der Beschreibung etwa durch die strukturierte Dokumentation ihrer fachspezifischen Bedeutung.

Kapitel 4

Migration getrennt gehaltener Geodaten in ein Relationales Datenbanksystem

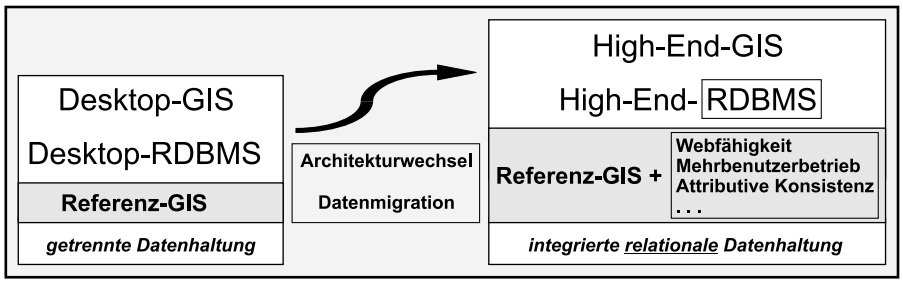

Abb. 4.1: Leistungsgewinn durch integrierte, relationale Datenhaltung

Das folgende Kapitel beschreibt die logische und physikalische Umsetzung des integrierten relationalen Datenhaltungsansatzes nach Abschnitt 2.1 (Ansatz 2). Den Rahmen der empirischen Untersuchungen bildet der Ausbau des Referenz-GIS „Nationalpark Bayerischer Wald“ (vgl. Abschnitt 2.4) zu einem Web-fähigen High-End-GIS (= Referenz-GIS +), das interdisziplinäre GIS-Anwendungen miteinander verknüpft und für GIS-Laien nutzbar macht. Die Konzeption dieses Geodatenservers (GDS) folgt damit dem in [SCHILCHER et al., 2001] formulierten Ansatz, dass sich verschiedene Fachdisziplinen durch den Einsatz von GIS-Technologie über das Internet nutzbringend zusammenführen lassen.

Neben der Beschreibung der bei der Migration der forstlichen und touristischen Fachdaten (Tab. 2.1 und 4.1) erzeugten Datenbankstrukturen steht die Implementierung einer Zugriffsschicht zur Unterstützung benutzerfreundlicher, internetbasierter Zugriffe auf den Geodatenserver im Vordergrund. Die Zugriffsschicht fördert bei lesenden Transaktionen die Transparenz des aktuell verfügbaren Datenangebots und reduziert den zentralen Fortführungsaufwand durch die effiziente Steuerung von dezentralen, schreibenden Transaktionen. Unabdingbare Basis hierfür ist die Umsetzung eines zentralen Metadatenkonzepts. Das bestehende relationale Datenmodell der Metadatenbank des Referenz-GIS wurde entsprechend konzeptionell erweitert und für den integrierten Datenhaltungsansatz optimiert. Das Kapitel endet mit der Beschreibung ausgewählter GIS-Anwendungen, die unmittelbar auf den erzeugten DB-Strukturen aufsetzen.

Im Folgenden wird zunächst das GIS-Systemkonzept des GDS vorgestellt.

4.1 GIS-Systemkonzept

Das GIS-Systemkonzept des GDS besteht im wesentlichen aus den Komponenten *Geodaten*, *Metadatenkonzept* und *Systemplattform*, sowie einer *Zugriffsschicht* zur gezielten Distribution der vorgehaltenen Geodaten bzw. zur fachanwendungsspezifischen Fortschreibung des Datenbestandes. Anhang B zeigt die vollständige Abbildung des GIS-Systemkonzepts unter dem Gesichtspunkt der Verknüpfung interdisziplinärer GIS-Anwendungen.

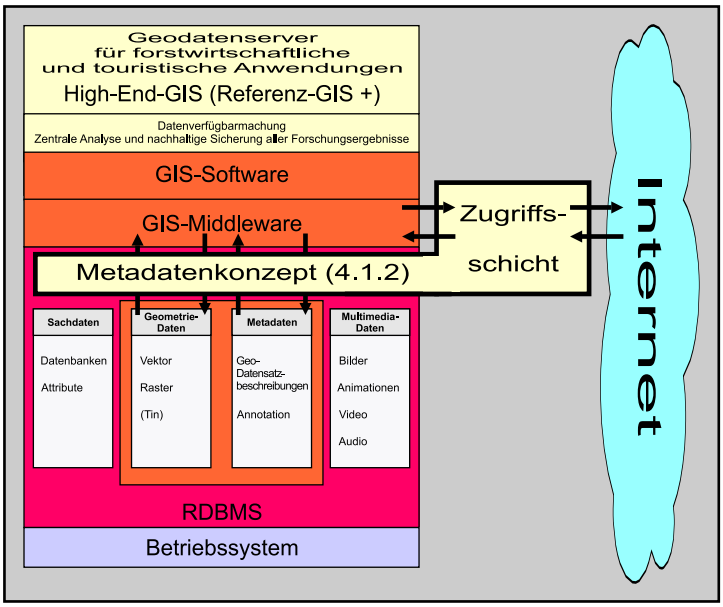

Abb. 4.2: High-End-GIS (Referenz-GIS +) auf Basis integrierter, relationaler Datenhaltung

4.1.1 Geodaten

Der Datenbestand des Geodatenservers deckt sich im wesentlichen mit den Ausgangsdaten des Referenz-GIS (vgl. Abschnitt 2.4), ist jedoch um touristische Fachdaten der Projektregion Nationalpark Bayerischer Wald erweitert. Eine detaillierte Übersicht aller touristischer Datenquellen ist [NEUMEIER, 2004] zu entnehmen.

	Thema	Thementyp	Quelle
Touristische Points of Interest	Adresse	Adressthema	ALB (Flurstücksdatensatz, Ortsdatenbank)
	Bahnhof	Vektorthema	Referenz-GIS „Nationalpark Bayerischen Wald“
	Beherbergung	Adressthema	Zweckverband der Nationalparkgemeinden
	Baudenkmal	Vektorthema	EUREGIO Bayerischer Wald-Böhmerwald, Šumava
	Bushaltestellen	Vektorthema	EUREGIO Bayerischer Wald-Böhmerwald, Šumava
	Gastronomie	Adressthema	Zweckverband der Nationalparkgemeinden
	Gemeinde	Vektorthema	Bayerische Ortsdatenbank
	Geologie	Vektorthema	Waldwildnis e.V.
	Museum	Vektorthema	Waldwildnis e.V.
	Radtour	Vektorthema	Nationalpark Radwegekarte
	Straßenname	Vektorthema	Digitale Flurkarte
Rasterdaten	Übersichtskarte TVO	Rasterthema	Tourismusverband Ostbayern
	SPOT pan Mosaik	Rasterthema	National Image and Mapping Agency (USA)
	TK100	Rasterthema	Referenz-GIS „Nationalpark Bayerischen Wald“
	DFK Zwiesel	Rasterthema	Bayerische Vermessungsverwaltung
Intermodales Routing	Teleatlas Straßennetz	Vektorthema	Zukunft Biosphäre GmbH
	HAFAS ID	Attribute	Gevas Software GmbH

Tab. 4.1: Datenbestände des Benutzerschemas TOURISMUS des Geodatenservers

Für die Realisierung einer internetfähigen Forschungsplattform, die einen Großteil der in Abschnitt 2.2 definierten konzeptionellen Anforderungen erfüllt (vgl. Tab. 4.2), ist ein integriertes Datenhaltungskonzept unerlässlich. Dabei erfolgt die Verwaltung von Geometrie- und Sachdaten in relationalen Tabellen mit Hilfe eines RDBMS auf Grundlage seiner spezifischen, atomaren Datentypen.

√ ausreichend erfüllt (√) bedingt erfüllt x unzureichend erfüllt

Konzeptionelle Anforderungen aus Abschnitt 2.2	**erfüllt**
Zentrale Benutzerverwaltung und feingranulare Zugriffsrechte	√
Flexibler Austausch von Geodaten und Metadaten über das Web	√
Skalierbarkeit der Systemplattform	√
Verfügbarkeit von zentralen Backup- und Recovery-Mechanismen	√
Mehrbenutzerzugriff und Transaktionskonzept für Geometriedaten (lange Transaktionen)	√
Ressourcen-Transparenz bzgl. des Datenangebots für lesende Zugriffe	√
Zuverlässigkeit des Gesamtsystems	√
Zusammenführung und Analyse von Geodaten	(√)
Flexible Modellierungsmittel zur Abbildung von 2D und 3D inkl. Zeitbezug	(√)
Wahrung der Konsistenz zwischen Geometrie-, Sach- und Metadaten	(√)
Offener Zugriff auf die Geodatenbasis	x
Standardisiertes und konsistentes Handling von Annotation	x
Modellierbarkeit bekannter komplexer Objekte und deren Beziehungen	x

Tab. 4.2: Erfüllung konzeptioneller Anforderungen an das Geodatenmanagement durch integrierte, relationale Datenhaltung

Eine ausführliche Bewertung der Leistungsfähigkeit des integrierten, relationalen Datenhaltungsansatzes sowie eine Diskussion der durch ORDBMS erzielbaren Mehrwertaspekte folgt in Kapitel 6.

4.1.1.1 Integrierter Relationaler Datenhaltungsansatz

Nach der Vorherrschaft von, für das schnelle Editieren und Schreiben sowie für die schnelle grafische Darstellung optimierten, proprietären Datenformaten für Vektordaten (z.B. ESRI-Shape-Format), existieren seit geraumer Zeit herstellerspezifische, relationale Datenbankmodelle für das integrierte datenbankbasierte Management dieser Geometriedaten. Entsprechendes gilt für die Verwaltung von großen Rasterdatenbeständen mittels RDBMS. Diese Datenmodelle werden über die Middleware-Komponente des jeweiligen GIS-Herstellers erzeugt und verwaltet. Demnach fungiert die GIS-Middleware als Applikationsserver zwischen der GIS-Software und dem RDBMS und liefert bei raumbezogenen Anfragen von Seiten der Clients die entsprechenden Geometriedaten zurück (vgl. Ansatz 2 bzw. 3, Abschnitt 2.1).

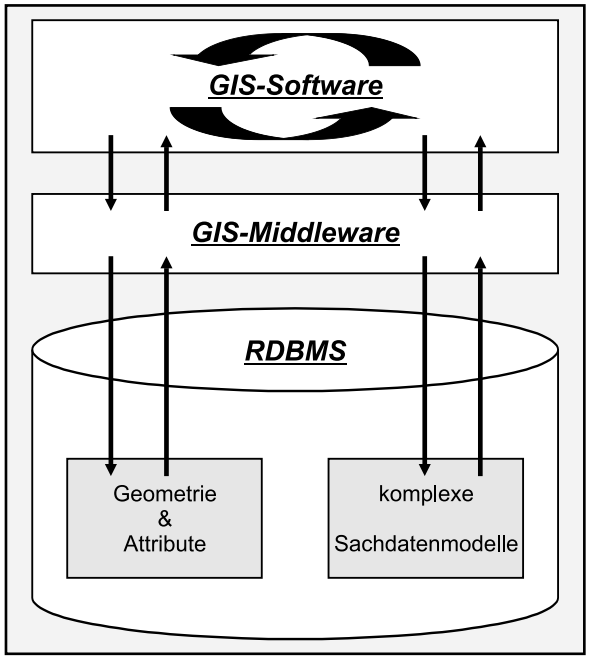

Abb. 4.3: Integrierter Datenhaltungsansatz des GDS

Direkte Attribute, die ursprünglich im Geometrie-Datenformat eines GIS-Herstellers gespeichert sind, werden beim integrierten Datenhaltungsansatz nunmehr in relationalen Tabellen abgelegt, die über Fremdschlüsselbeziehungen mit den Tabellen verknüpft sind, die die zugehörigen Geometrieobjekte aufnehmen, sofern für die Speicherung der Geometriedaten auf binäre Datentypen des RDBMS (z.B. BLOB, LONG RAW) zurückgegriffen wird. Bevorzugt man dagegen die unmittelbar von einem ORDBMS angebotenen, abstrakten Datentypen zur Verwaltung der Geometriedaten, so werden diese zusammen mit den direkten Attributen in lediglich einer objektrelationalen Tabelle gespeichert (vgl. Abschnitt 5.1.1).
Indirekte, den Geometriedaten über 1:1-Verknüpfungen manuell zugeordnete Attribut-Tabellen werden ebenso in relationalen Tabellen abgelegt.
Alle weiteren alphanumerischen Sachinformationen des GDS werden als komplexe Sachdaten klassisch nach dem relationalen Modell abgebildet und über das RDBMS in der Regel in der gleichen Datenbankinstanz wie die Geometriedaten verwaltet. Der Zugriff auf diese forstlichen und touristischen Sachinformationen erfolgt standardisiert via SQL über das RDBMS oder alternativ mittels entsprechender ODBC-Treiber über die GIS-Software. Darüber hinaus steht eine Vielzahl an Produkten von Drittanbietern für den Zugriff auf Sachdaten in RDBS zur Verfügung (z.B. Quest Software, [QUEST, 2004]).

4.1.1.2 Strukturierung des Datenbestandes

Die hohe Flexibilität eines Dateisystems zur räumlichen, thematischen und datentypspezifischen Strukturierung von Geometriedaten kann durch relationale Tabellen eines RDBMS als einzig verfügbare physikalische Ebene selbstredend nicht erreicht werden.
Bei der Migration eines hochstrukturierten Geometriedatenbestandes eines Dateisystems in ein integriertes Datenhaltungskonzept muss sich daher die hierarchische Ordnerstruktur des Filesystems durch eine geeignete Nomenklatur-Konvention unmittelbar in der Benennung der Tabellen wiederfinden, sofern man den Strukturierungsgrad vollständig erhalten möchte.
Dies kann etwa durch die Abbildung der absoluten Pfade und / oder eindeutiger Identifikatoren auf den zu vergebenden Tabellennamen eines Geodatensatzes erfolgen.
Die Nomenklatur für die Bezeichnung der Vektor- und Rasterdatensätze des GDS basiert auf der Sequenz logischer Abkürzungen für jede Hierarchieebene sowie eines für jeden Geodatensatz eindeutigen, vierstelligen Identifikators (vgl. Abb. 4.4). Der Identifikator ist als Metainformation der Metadatenbank des Referenz-GIS „Nationalpark Bayerischer Wald“ (vgl. Abschnitt 2.4) verfügbar.
Die aus der Nomenklatur hervorgehende Kurzbezeichnung der Geodatensätze ist zudem aufgrund der Beschränkung der Benennung von Tabellen auf 30 Character der zugrunde liegenden Oracle-Datenbank erforderlich.

Komplexe Sachdatentabellen des GDS werden dem gleichen Datenbank-Benutzerschema zugeordnet wie die Geometriedatensätze. Die Metadatentabellen werden dagegen physikalisch getrennt in einem separaten Schema verwaltet.
Die vollständige Beschreibung des Migrationsverfahrens sowie der konzeptionellen Erweiterung der vorliegenden Sachdatenmodelle erfolgt in Abschnitt 4.2.

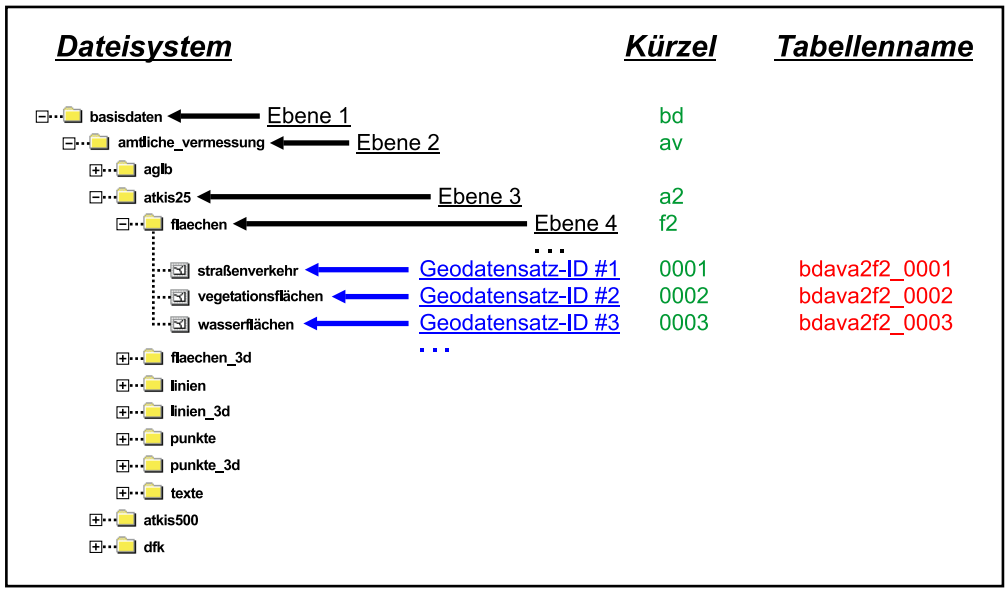

Abb. 4.4: Ableitung einer Nomenklatur aus dem vorliegenden Dateisystem

4.1.2 Metadatenkonzept und Zugriffsschicht

Das Metadatenkonzept dient der umfassenden, thematischen und qualitativen Beschreibung aller im GDS vorgehaltenen Geodatensätze.
Die Metainformationen sowie die Beschreibung der fachlichen Bedeutung der eingepflegten Geodaten stellen die Voraussetzung zur gezielten Recherche und zum effizienten Austausch von Geoinformationen zwischen interdisziplinären Anwendungen und deren Nutzung dar.
Aus Anwendersicht sollten Metadaten sowohl direkt mit ihren zugehörigen Geodatensätzen verknüpft als auch zentral für das effiziente, anwendungsspezifische Auffinden von Geodatensätzen vorliegen. Das Metadatenkonzept des GDS basiert daher auf einer Fusion folgender Ansätze:

1. **Datensatz-begleitende Metainformationen:**

   Datensatz-begleitende Metainformationen werden parallel zu jedem im GDS verfügbaren Geodatensatz in Form einer binär kodierten XML-Datei über die GIS-Software in einer Systemtabelle der GIS-Middleware gespeichert.

2. **Zentraler internetfähiger Geodatenkatalog:**

   Der Geodatenkatalog ist als internetfähige Metadatenbank organisiert. Diese Metadatenbank setzt auf dem vorhandenen Metadatenmodell des Referenz-GIS auf und ist gemäß des integrierten Datenhaltungsansatzes des GDS sowie der Anforderung des flexiblen fachübergreifenden Datenaustauschs konzeptionell erweitert (vgl. Abschnitt 4.2.4).

Um die synchrone Fortschreibung des Datenbestandes des GDS inklusive seiner Metadaten durch Fachanwender und die sofortige Verfügbarkeit neuer Geodatensätze für alle registrierten Benutzer zu gewährleisten, erfolgt ein geregelter, automatisierter und konsistenter Abgleich zwischen den beiden Metadaten-Komponenten. Die Inhalte der Metadaten sind konform zu den Vorgaben des Federal Geographic Data Committee (FGDC).

Das Metadatenkonzept des GDS ist eng an eine *Zugriffsschicht* gekoppelt, die erstens für den oben beschriebenen Abgleich zwischen den Metadaten-Komponenten sorgt, und zweitens durch die Berücksichtigung der gewährten Leserechte dem Anwender nach aussen hin seine individuelle Sicht auf den Datenbestand des GDS präsentiert.
Das Zugriffskonzept ist also benutzerspezifisch angelegt, d.h. es werden beim lesenden Zugriff nur die Geodatensätze angezeigt, für die der entsprechende Anwender Zugriffsrechte erhalten hat.
Eine ausführliche Beschreibung der Implementierung des Metadatenkonzepts erfolgt in Abschnitt 4.2.4.

4.1.3 Systemplattform

Die System-Plattform des Referenz-GIS + entspricht in ihrem Aufbau dem aktuellen Stand der Technik. Sie lässt sich in die drei Grundkomponenten Hardware, GIS-Software und DBMS gliedern.

4.1.3.1 Hardware und Betriebssystem

GIS-Software sowie GIS-Anwendungen laufen auf einer geeigneten Hardwarekonfiguration unter einem bestimmten Betriebssystem. Unter dem Hardware-Begriff werden dabei alle physischen Bestandteile der GIS-Systemplattform subsummiert. Die Hardwarekonfiguration der Forschungsplattform wird entsprechend der Client-/Serverumgebung wiederum durch drei Komponenten bestimmt.

Als **Geodatenbankserver** dienen ein leistungsfähiger Windows NT4-Server auf Intel-Basis mit 2 x PIII 850 MHz, 1 GB RAM und 4 x 18 GB Festplattenspeicher (Raid Level 5) sowie ein Red Hat Enterprise Linux Advanced Server 3.0 auf Intel Basis mit P4 2,4 GHz, 768 MB RAM und 40 GB Festplattenspeicher.

Als Backup-Server kann ein Windows NT4-Server auf Intel-Basis mit 2 x PIII 1 GHz, 1 GB RAM und 90 GB Festplattenspeicher im LAN des Fachgebiets Geoinformationssysteme genutzt werden.

Die **GIS-Arbeitsplätze der Clients**, die auf die Datenbestände der Forschungsplattform zugreifen, sind Windows NT4-Workstations auf Intel-Basis. Da das Geoprocessing für konkrete Auswertungen i.d.R. lokal stattfindet, müssen auch die Client-Rechner über ausreichende Ressourcen verfügen. Die wichtigsten Eckdaten der GIS-Arbeitsplatzmaschinen lassen sich durchschnittlich mit 2 x PIII 850 MHz, 512 MB RAM, 20 GB Festplattenspeicher angeben.

Die interne **Vernetzung** (LAN) der Hardware der Forschungsplattform erfolgt hybrid sternförmig mittels 2 GBit-Glasfaserleitung zwischen einer 1 GBit/100 MBit-24-Port-Switch und dem Geodatenbankserver sowie 600 MBit-Kupfer-Drahtleitungen (Twisted Pair) zwischen der Switch und den Arbeitsstationen. Die externe Vernetzung erfolgt über eine 100 MBit-Drahtleitung an einen Hochleistungs-Netzknoten des internen Hochschulnetzes (internes WAN) und von dort glasfaserbasiert weiter an das Leibnitz Rechenzentrum (LRZ) des Deutschen Forschungsnetzes (DFN) und dadurch an das Internet (externes WAN) [HUBER (2), 2002].

4.1.3.2 GIS-Software und Erweiterungen

Die **GIS-Software** des Referenz-GIS + basiert auf der Produktfamilie der Fa. ESRI (Environmental Systems Research Institute). Dieser Entscheidung lag kein Evaluierungsprozess verschiedener Systeme zu Beginn der Arbeiten zugrunde, sondern resultierte unmittelbar aus der Tatsache, dass ArcView GIS 3.x für den Aufbau des Referenz-GIS „Nationalpark Bayerischer Wald“ herangezogen worden war und somit der zu migrierende Basisdatenbestand im ESRI-Shape-Format vorlag. Zudem setzt die Nationalparkverwaltung Bayerischer Wald (NPV) als Kooperationspartner der TU München seit der GIS-Einführung im Jahr 1996 ebenfalls dieses Desktop-GIS ein.

Die GIS-Software umfasst zwei Komponenten der etablierten 3-tier Architektur für den GIS-Einsatz im Mehrbenutzerbetrieb.
Die Middleware-Komponente ArcSDE der Fa. ESRI bildet die Schnittstelle zur Geodatenbank und wird als Applikationsserver in der Version 8.3 eingesetzt. ArcSDE läuft unter dem Betriebssystem Windows NT4 auf dem gleichen Server wie das DBMS, was der klassischen Konfiguration entspricht. Für die Evaluierung der OR-Datenbankkonzepte (vgl. Kapitel 5) wird stattdessen eine Remote-Variante gewählt. Hier läuft ArcSDE auf einem Windows-NT 4-Client-Rechner und greift über das LAN des Fachgebiets Geoinformationssysteme auf eine separate Datenbank zu, die von einem Linux-Datenbankserver gehostet wird.
Als ArcSDE-Clients fungieren das High-End GIS ArcInfo Desktop 8.3 bzw. für lesende Zugriffe von Seiten der GIS-Applikationen auch ArcView 8.3. Der Funktionsumfang der GIS-Software wird ergänzt durch die beiden Erweiterungen ArcGIS Spatial Analyst und ArcGIS 3D Analyst sowie extern entwickelten und frei verfügbaren

Systemerweiterungen auf VBA-Basis. Die in Abschnitt 4.3 skizzierte Zugriffsschicht zur Kontrolle des lesenden und schreibenden Zugriffs auf den GDS basiert auf zwei an der TU München entwickelten Java-Applikationen.

Die eingesetzten GIS-Softwarekomponenten umfassen:

ArcSDE:
ArcSDE ist ein geographischer Applikationsserver, der Vektor-, Raster- und Metadaten in einem DBMS zentral verwaltet und für eine Vielzahl von Nutzern parallel verfügbar macht. Neben binären Datentypen können für die Speicherung der Geodaten in einer Datenbank mit eigenen Erweiterungen zum Geodatenmanagement auch die jeweiligen raumbezogenen Datentypen und Indizierungsmethoden des DBMS direkt genutzt werden. ArcSDE unterstützt das *ArcGIS Geodatabase Modell*, ein komplexes Geodatenmodell zur Verwaltung verschiedener Objektstrukturen. ArcSDE ermöglicht den Mehrbenutzerbetrieb auf Basis von langen Transaktionen und Versionsverwaltung und bietet darüber hinaus Funktionalität zur temporären, integritätserhaltenden Entkoppelung bzw. Reintegration von Teildatenbeständen etwa zum Zwecke der mobilen Felderfassung. [ESRI (1), 2005]

ArcInfo: ArcInfo stellt als High-End GIS das funktional umfassendste GIS-System der ArcGIS Desktop Produktfamilie von ESRI dar. Es bietet eine Fülle an Funktionen im Bereich des Geoprocessing, der Analyse von Geodaten sowie der Datenkonvertierung von Vektor-, Raster- und CAD-Formaten und kann über die Objektbibliothek ArcObjects angepasst und erweitert werden. ArcInfo bietet ferner Tools zur Verwaltung von Metadaten, Topologien und Versionen an und ermöglicht in Kombination mit ArcSDE den gesicherten Mehrbenutzerbetrieb für umfangreiche Geodatenbestände in DBMS. [ESRI (1), 2005]

ArcGIS Spatial Analyst: ArcGIS Spatial Analyst ergänzt ArcGIS Desktop Produkte wie ArcInfo im Bereich der Rasterfunktionalität und stellt Funktionen zur raster-basierten, räumlichen Modellierung und Auswertung zur Verfügung. Durch Raster/Vektorkonvertierung, das Berechnen und Analysieren von Oberflächen oder durch Rasterverschneidung lassen sich neue Informationen aus Raster- und Grid-Daten gewinnen. [ESRI (1), 2005]

ArcGIS 3D Analyst: ArcGIS 3D Analyst erweitert das ArcGIS Basispaket um die dritte Dimension. Die Funktionalitäten umfassen insbesondere die Generierung, die Analyse und die anschauliche Visualisierung und Animation von Oberflächendaten, die in den ISO-Standard VRML exportiert und somit auch durch externe Software weiterverarbeitet werden können. Mit ArcGIS 3D Analyst können GRID-, TIN-, CAD- sowie Rasterdaten, ArcInfo Coverages und 3D-Shapefiles verwendet und über die Visualisierungsapplikation ArcScene gemeinsam dargestellt und verarbeitet werden. [ESRI (1), 2005]

4.1.3.3 Datenbankmanagementsystem

Die Implementierung der relationalen und objektrelationalen Datenbankkonzepte erfolgt auf Basis des Datenbankmanagementsystems Oracle, Enterprise Edition, in den Releases 8.1.7 bzw. 9.2.0.
Oracle ist weit mehr als eine Datenbank, Oracle stellt auch ein integriertes Paket zur datenbankbasierten Anwendungsentwicklung dar. Es enthält eine Vielzahl an Tools und ganze Programmierumgebungen, die neben der Datenbankanwendung installiert werden können. Das DBMS verfügt neben einfachen, kommandozeilenorientierten Werkzeugen zur Ausführung von SQL-Befehlen auch über komfortablere Clients mit grafischen Oberflächen wie z.B. der Oracle Management Konsole, die u.a. zur Administration, Migration und Auslastungsanalyse einer Datenbank eingesetzt werden kann. Es existieren Programmierschnittstellen für die wichtigsten Entwicklungsumgebungen wie Java (JDBC), ODBC, Perl (DBI), C/C++. Mit SQLJ können mittels der mitgelieferten Oracle Virtual Machine statische und auch dynamisch erzeugte SQL-Befehle in Java eingebettet werden [BOHN, 2003]. Oracle unterstützt die wichtigsten objektorientierten Neuerungen von SQL:1999 und SQL:2003 (vgl. Abschnitte 3.4 und 3.5).

Mit der Datenbankerweiterung Oracle Spatial können Vektordaten basierend auf dem nativen Datentyp *SDO_GEOMETRY* unmittelbar in der Datenbank verwaltet, gepflegt und analysiert werden. Die zu diesem Zweck notwendigen Algorithmen sind direkt im Datenbankkern implementiert und können aus allen SQL-fähigen Anwendungen heraus genutzt werden. Ein um räumliche Operatoren und geometrische Funktionen ergänzter

SQL-Sprachumfang ermöglicht geometrische Abfragen auf räumlich indizierte, triviale und komplexe Objektgeometrien [ORACLE (7), 2003].
Entsprechend stehen mit der Oracle Version 10g ein neuer Datentyp *SDO_GEORASTER* (vgl. Abschnitt 5.1.1) sowie eine Reihe von Methoden zum direkten Rasterdatenmanagement inkl. ihrer Metadaten unter Nutzung des Datentyps XMLTYPE zur Verfügung [ORACLE (5), 2003].
Weitere bedeutende Neuerungen sind die Erweiterung der räumlichen Operatoren und Funktionen (u.a. Export von Geometrie-Objekten in GML, Version 2.1.1) sowie die Einführung eigener Speichermodelle zur Repräsentation von Netzwerken und Topologie [ORACLE (2), 2004].

4.2 Übernahme von Datenbeständen

Die Grundlage der Migration bilden die Vektor-, Raster- und Grid-Datenbestände des Ausgangsdatenbestands (vgl. Tab. 4.3). Dagegen sind die integrierten Datenhaltungskonzepte der führenden GIS-Hersteller bislang nicht für die Speicherung von TIN-Daten konzipiert, da aufgrund des hohen Speicherbedarfs die On-The-Fly-Berechnung aus den zugrundeliegenden Basisdaten priorisiert wird.
Im Zuge des Aufbaus des Referenz-GIS (vgl. Abschnitt 2.4) wurden Geodaten aus verschiedenen heterogenen Datenquellen in eine zentrale GIS-Forschungsplattform integriert. Um die Strukturen und Inhalte der einzelnen Datenmodelle der genutzten heterogenen Datenquellen aufbrechen und in einem einheitlichen Modell zusammenführen zu können, war nach [HUBER (2), 2002] die thematische Klassifizierung der übernommenen Geodaten nötig. Die thematische Klassifizierung umfasst Informationen zu Datenquelle, Objekt- und Themenbereich eines Geodatensatzes. Diese Informationen sind als Metadaten in einem Relationenmodell abgebildet.

Als Basis der Migration des Geodatenbestandes in ein integriertes Datenhaltungskonzept liegen somit singuläre Geodatensätze vor, die über Schlüsselspalten mit ihren Metainformationen verknüpfbar sind. Nur im Verbund mit ihren zugehörigen Metainformationen lassen sich die Geodaten auch nach der Übernahme noch modellspezifisch bewerten. Die Migration der Geodaten erfolgt demnach Datensatz- und nicht modellbasiert, d.h. im physikalischen Modell des Ausgangsdatenbestands existieren keine logischen Beziehungen zwischen Geoobjekt-Klassen, die bei der Migration berücksichtigt werden können / müssen.

Eine *modellbasierte Migration* im Sinne der exakten formalen Beschreibung der Geoobjekt-Klassen durch eine konzeptionelle Beschreibungssprache mit automatischer Herleitung von Transferformaten (modellbasierter Datentransfer), wie etwa bei der Schweizer Norm *Interlis*, wäre für die Übernahme der vorliegenden singulären Geodatensätze alternativ zum hier beschriebenen Verfahren möglich (vgl. [INTERLIS, 2004]), sofern Regeln für die automatische Herleitung des Transferformats (XML/GML bei Interlis) sowie Prozessoren zum Export und Import der Geodaten in das bzw. aus dem Transferformat existieren ([GNÄGI et al., 2004]).

Im Folgenden wird die (System-interne) Abbildung des vorliegenden Geodatenbestandes auf Relationen für Vektordaten und Rasterdaten erläutert.

4.2.1 Migration von Vektordaten

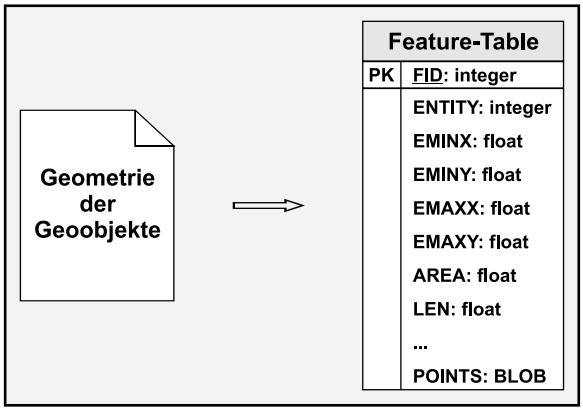

Abb. 4.5: Abbildung der Objektgeometrie auf Relationen

Bei der Konvertierung von proprietären Vektorformaten in herstellerspezifische Datenbankstrukturen - wie im vorliegendem Fall die Migration eines im ESRI-Shape-Format vorliegenden Vektordatenbestandes in das ArcSDE-Datenmodell - wird jeder Geodatensatz (Geoobjekt-Klasse) auf genau eine Feature-Tabelle abgebildet. Die Abbildung der Geometrieobjekte auf die Tupel der Relation ist wohldefiniert und bijektiv, d.h. jedes Geometrieobjekt wird auf genau ein Tupel der Feature-Tabelle abgebildet. Die geometrische Ausprägung der Geoobjekte wird als Koordinaten-Array ausgegeben und komprimiert in einer Tabellenspalte *(POINTS)* auf Basis der Datentypen BLOB oder LONG RAW gespeichert [ESRI, 2001].

Die Speicherung der Geometrie mittels binärer Datentypen lässt sich wie folgt begründen:

Für die Einhaltung der ersten Normalform der relationalen Datenbank müssten Geometrieobjekte atomisiert und auf viele verschiedene Tabellen aufgeteilt werden, was unmittelbar in teueren Rechenoperationen bei der Reaggregation und Visualisierung von Geometriesegmenten resultiert. Da relationale Tabellen nicht geordnet sein müssen (Relation), wäre zudem die Einführung einer Domäne „Sequence“ erforderlich, um die korrekte Reihenfolge der Stützpunkte zu wahren.(vgl. früheres *Normalized Schema* der ArcSDE).

Entsprechendes gilt bei räumlichen Anfragen, wo in Vergleichsoperationen tausende solcher Verbindungen in Echtzeit miteinander verglichen werden müssen ([BARTELME, 2000]).
Statt einem streng nach relationalen Datenbank-Standards aufgebauten Datenmodell bietet sich stattdessen die Speicherung der Geometrie in BLOBs (Binary Large Objects) an. Diese Datensätze sind vom DBMS nicht näher aufschlüsselbar und werden als Ganzes gelesen, geschrieben oder ausgetauscht. Sie können lediglich von der Anwendung her interpretiert werden, wodurch sich die hohe Systemabhängigkeit erklärt.

Die Feature-ID *FID* ist Primärschlüssel der Tabelle, ist somit für jedes Geoobjekt bzw. für jede Tabellenzeile eindeutig und wird von der GIS-Software in Interaktion mit dem RDBMS gepflegt. Über die FID sind direkte Attribute (vgl. Abb. 4.6) und räumlicher Index (vgl. Abb. 4.7) mit der Geometrie der Geoobjekte verknüpft. Im Zuge der Konvertierung werden vom GIS-System implizite Metadaten wie *MBR* (Minimum Bounding Rectangle), Anzahl der Stützpunkte, Geometrietyp, Fläche und Umfang bzw. Länge der Geoobjekte abgeleitet und zur Beschleunigung räumlicher Anfragen in der Feature-Tabelle gespeichert. Bei der Fortführung der Geometrien werden diese Informationen vom GIS-System automatisch neu berechnet.

Die direkten Attribute der Geoobjekte werden ebenso bijektiv auf Tupel einer separaten Tabelle (Business-Tabelle) abgebildet. Die genaue Umsetzung des Mappings der numerischen und textuellen Datentypen der Dbase-Datei auf ihre Pendants des eingesetzten RDBMS bleibt dabei für den Anwender verborgen. Das Feld *SHAPE* ist Fremd -schlüssel für die Verknüpfung der Attribute mit ihren zugehörigen Geometrie-Objekten der Feature-Tabelle. Beim ESRI-Shape-Format ist diese Verknüpfung durch eine dritte, zwingend erforderliche Datei, die den Index der Objektgeometrien aufnimmt (vgl. Abb. 4.7), realisiert (vgl. [ESRI, 1998]).

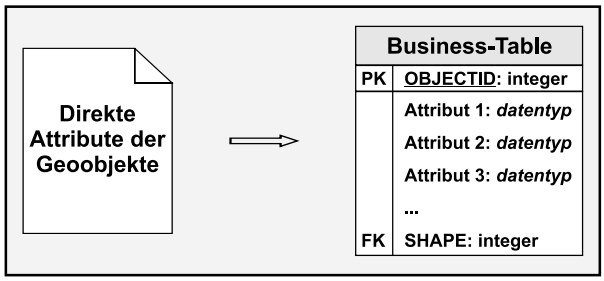

Abb. 4.6: Abbildung der direkten Attribute auf Relationen

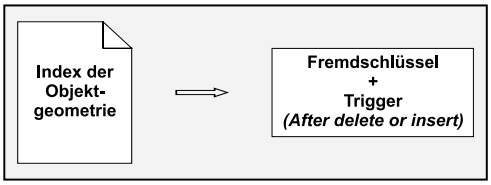

Abb. 4.7: Abbildung der Verknüpfungssemantik

Die Funktion dieser Datei wird im Relationenmodell unmittelbar durch den oben beschriebenen Constraint erfüllt. Alternativ dazu werden Trigger als Mittel der aktiven Datenbank für die Wahrung der Konsistenz eingesetzt. Im Falle, dass sämtliche Attribute eines Geoobjekts aus der Business-Tabelle durch eine entsprechende Transaktion gelöscht werden, wird die referentielle Integrität über einen Trigger sichergestellt, d.h. der zugeordnete Geometriedatensatz wird automatisch aus der Feature-Tabelle entfernt. Der konträre Fall, d.h. die Eliminierung eines Tupels aus der Feature-Tabelle erfordert dagegen zwingend den Einsatz des dem Relationenmodell zugrunde liegenden GIS-Systems, um Inkonsistenzen zu vermeiden. Wird die Transaktion stattdessen direkt über SQL angestoßen, führt dies zu Inkonsistenzen, da die Attribute nach dem Löschen der zugehörigen Geometrie-Objekte in der Business-Tabelle verbleiben.

Die Festlegung des räumlichen Bezugssystems durch die Parametrisierung des geodätischen Datums und der Projektion erfolgt auf Fileebene durch eine weitere Datei. Die Parameter dieser Datei werden bei der Konvertierung auf Datenbankstrukturen im Feld *SRTEXT* der Tabelle „Spatial_References“ gespeichert. Maßstab und Längeneinheiten werden in entsprechenden Feldern verwaltet. Die Verknüpfung mit Feature- und Business-Tabelle erfolgt über das Primärschlüssel-Feld *SRID* und eine übergeordnete Tabelle „Layers“, in der alle Vektordatensätze registriert sind. Ist die Information bzgl. des räumlichen Bezugssystems eines Geodatensatzes auf Fileebene nicht verfügbar, so kann diese auch nach erfolgter Migration durch die GIS-Software explizit zugewiesen werden.

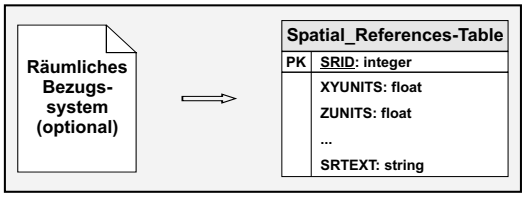

Abb. 4.8: Abbildung des räumlichen Bezugssystems

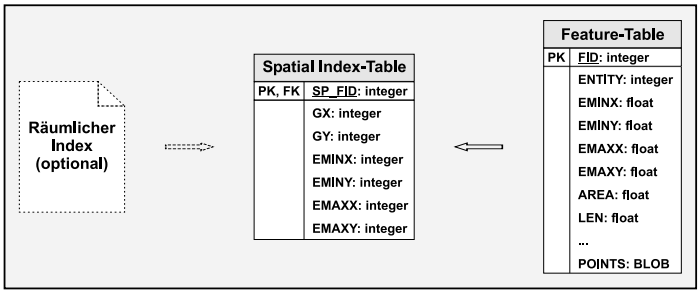

Abb. 4.9: Neuberechnung des räumlichen Index

Der räumliche Index wird anders als die oben angeführten attributiven und geometrischen Strukturen der Geoobjekte nicht aus der auf Fileebene verfügbaren Binärdatei übertragen, sondern im Zuge der Migration neu aus den geometrischen Informationen der Feature-Tabelle berechnet. Die Felder *GX* und *GY* speichern dabei die Koordinaten der Kacheln eines einfachen, regulären Rasters. Es können bis zu drei solcher *GRID-Levels* unterschiedlicher Zellgröße synchron genutzt werden. Die Spatial-Index-Tabelle erhält für jede Kachel-Feature-Kombination einen Eintrag. Der Join der Spatial Index-Tabelle mit der Feature-Tabelle erfolgt über den Fremdschlüssel *SP_FID*. Darüber hinaus werden eine Reihe von Indizes als Datenbankobjekte abgelegt, die die FID-, LEN-, bzw. die AREA-Spalte der zugehörigen Feature-Tabelle indizieren.

Indizes auf Attribute werden ebenfalls neu auf Basis der zugrunde liegenden Datenbanktabellen erzeugt und als klassische Datenbankobjekte im gleichen Benutzerschema abgelegt. Vom GIS-System werden automatisch die wichtigsten Attributspalten (SHAPE, OBJECTID) indiziert.

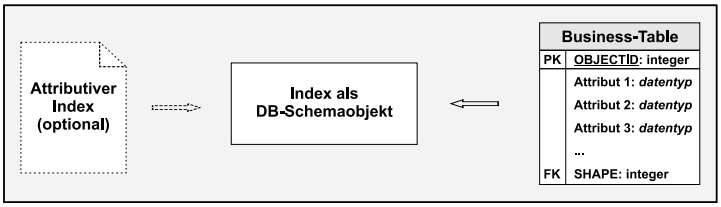

Abb. 4.10: Wiederaufbau des attributiven Index

Zur Unterstützung des Mehrbenutzerbetriebs durch Versionierung werden bei der GIS-basierten Fortführung der direkten Attribute der Business-Tabelle Änderungen nicht direkt in der Business-Tabelle sondern in separaten *Add-* und *Delete-Tabellen* gespeichert. Die attributiven Änderungen sind nur sichtbar, wenn die Version abgeglichen wurde, was den Einsatz der entsprechenden GIS-Software erfordert.

Die hier skizzierten Relationen für die datenbankbasierte Verwaltung von Vektordaten stellen lediglich den Kernausschnitt eines komplexen, proprietären Datenmodells dar, das die elementare Basis für die Implementierung und Nutzung erweiterter GIS-Funktionalität bildet. Darunter sind insbesondere zu nennen:

- Implementierung von Integritätsbedingungen für Geodaten (Check-Constraints, Attribut-Domänen)
- Modellierung von Beziehungen zwischen Geoobjekten für spezielle GIS-Anwendungen
- Validierung topologischer Beziehungen von Geoobjekten
- Integrierte Verwaltung von Geodaten und zugehöriger Metadaten
- Versionierung von Geodaten als Ansatz zum Mehrbenutzerbetrieb und zur Historienführung

4.2.2 Migration von Raster- und Griddaten

Die Basis zur Übernahme von Rasterdaten bilden singuläre Rasterdateien amtlicher und privatwirtschaftlicher Anbieter im TIFF-Format (vgl. Tab. 2.1). Der Raumbezug wird überwiegend extern über Worldfiles hergestellt (Geotransformation).
Grid-Daten, etwa diverse DGM-Produkte wie Hangneigungs- und Expositions-Klassifizierungen, ergänzen den vorliegenden Ausgangsdatenbestand.

Das ArcSDE-Datenmodell ist bzgl. des Rasterdaten-Managements ähnlich konzipiert wie für die Verwaltung von Vektordaten (vgl. [ESRI (1), 2002]). Rasterdatensätze sind analog zu Vektorlayer in einer übergeordneten System-Tabelle (Raster_Columns) registriert, die nicht im Benutzerschema sondern im SDE-Schema gespeichert ist. Der Konvertierungs-Prozess erzeugt für jede Rasterdatei jeweils eine Business-Tabelle als Master- und spezielle Detailtabellen, die Metadaten über das Rasterbild und seine Bänder (u.a. Bounding Box, Pixeltiefe) sowie

die Pixeldaten selbst (Feld *BLOCK_DATA*) speichern. Die korrekte Geocodierung des Rasterbildes wird als Detailinformation in den Metadaten-Tabellen abgelegt. Raster-Statistiken und Farbpalette als weitere Informationen zu den Bändern eines Rasterbildes werden gesondert in der Auxiliary-Tabelle verwaltet (vgl. Abb. 4.11).

Spezielle Constraints und Trigger sorgen nach einem Update der Business-Tabelle, etwa bei der Erzeugung von Rasterkatalogen, für die konsistente Fortführung der Detailtabellen. Die exakte Implementierung der Integritätsbedingungen bleibt für den Anwender jedoch verborgen. Für die weiterführende Betrachtung des Rasterdaten-Managements mit ArcSDE sei auf [ESRI (2), 2002] verwiesen.

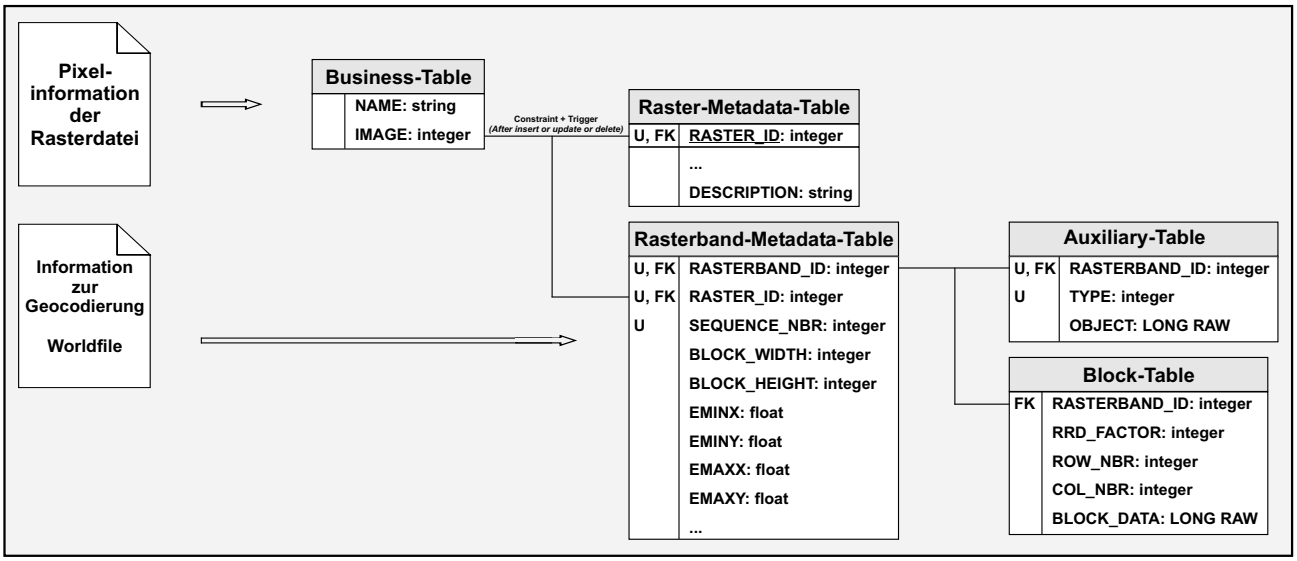

Abb. 4.11: Abbildung von georeferenzierten Rasterdaten auf ein herstellerspezifisches Relationenmodell

Das Mengengerüst des Geodatenbestandes als Basis der Migration in relationale Datenbankstrukturen umfasst insgesamt 828 Geodatensätze, darunter 585 Vektorthemen, wovon wiederum 196 auf 3D-Vektoren entfallen.
In der folgenden Übersicht wird auf die Unterscheidung nach 2D- und 3D-Vektoren verzichtet, da deren logische Abbildung auf die skizzierten Relationenschemata in beiden Fällen identisch ist. Das physikalische Modell der Abbildung als Differenzierungskriterium bleibt beim integrierten Datenhaltungsansatz für den Anwender verborgen.

	Anzahl der Geodatensätze	**Anzahl der Geoobjekte**	**Datenvolumen in [MB]**
Vektoren			
Punkte	102	3.779.891	315
Linien	220	994.398	971
Flächen	263	650.698	635
Rasterdaten	225	-	12.930
Grid	18	-	32
Gesamt	828	5.424.987	14.883

Tab. 4.3: Mengengerüst der migrierten Geodaten

Die Konvertierung erfolgt auf Kommandozeilenebene oder durch Wizards optional im Batchbetrieb. Die Benennung der Business-Tabellen der konvertierten Geodatensätze folgt dabei der in Abschnitt 4.1.1.2 vorgestellten Nomenklatur.
Das Zielschema des Datenbestandes ist gemäß des auf seiner Basis entwickelten Anwendungsspektrums *FORST*. Touristische Fachdaten (vgl. Abschnitt 4.1.1) sind entsprechend dem Zielschema *TOURISMUS* zugeordnet.

4.2.3 Migration von Sachdaten

Die zu migrierenden Sachinformationen umfasssen indirekte, den Geometriedaten über 1:1-Verknüpfungen manuell zugeordnete Attribute sowie komplexe, relational modellierte Sachdaten (von Desktop-RDBMS nach High-End-RDBMS).

Die **Abbildung von indirekten Attributen** auf Relationen verläuft analog zu der in Abschnitt 4.2.1 skizzierten Form, jedoch erfolgt keine Registrierung der Business-Tabellen in der Systemtabelle „Layers“. Dies spiegelt sich für den Anwender in Form unterschiedlicher logischer Sichten auf die erzeugten Datenbankstrukturen, als *Feature-Klassen* (Vektorlayer) bzw. *Objektklassen* (indirekte Attribute), wider.

Die Verknüpfung indirekter Attribute mit der Objektgeometrie erfolgt klassisch auf Basis identischer Schlüsselfelder ihrer Business-Tabellen, wie Abb. 4.12 für die Wetterstationen des Nationalparks illustriert:

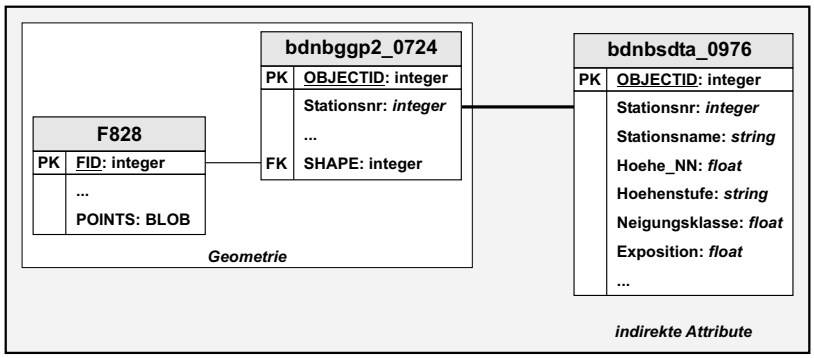

Abb. 4.12: Verknüpfung von Objektgeometrie und indirekten Attributen

Kern der **Migration relationaler Datenbankmodelle** von einem Quell- in ein Ziel-RDBMS ist das Mapping ihrer verfügbaren Basisdatentypen (vgl. Tab. 4.4) sowie die logische Abbildung und Benennung der Constraints und Indizes.

Die Abbildung der Datentypen zwischen Quell- und Ziel-RDBMS, die im Rahmen des im Folgenden beschriebenen, pragmatischen Migrationsansatzes herangezogen wurde, lässt sich wie folgt zusammenfassen:

Quell-RDBMS (MS Access 2000)	**Ziel-RDBMS (Oracle Database 8.1.7)**	**Bemerkung**
BYTE	NUMBER(3)	ganze Zahlen zwischen 0 und 255
(LONG)INTEGER	NUMBER(n)	ganze Zahlen; die Feldgröße wurde anhand des größten Feldwertes ermittelt
DECIMAL	FLOAT(126)	DECIMAL: Fließkommazahl mit dezimaler Präzision ≤ 28 FLOAT: Fließkommazahl mit binärer Präzision ≤ 126 ($\hat{=}$ dezimaler Präzision von ≤ 38)
SINGLE	FLOAT(63)	Zahl mit dezimaler Präzision ≤ 19
DOUBLE	FLOAT(126)	Zahl mit dezimaler Präzision ≤ 38
TEXT	VARCHAR2(n)	Zeichenkette variabler Länge (maximale Länge: 4000 Zeichen)
JA/NEIN-Feld	NUMBER(1)	Simuliertes Boolean-Feld: mit Wert ’0’ (Nein/False) und ’1’ (Ja/True)
DATUM/UHRZEIT	DATE	Datums- und Zeitangaben in Sekundengenauigkeit

Tab. 4.4: Datentypmapping zwischen Quell- und Ziel-RDBMS

Die 1:1 Migration von relationalen Datenbankstrukturen ist trivial, jedoch bei erforderlichen strukturellen Anpassungen infolge DBMS-spezifischer Charakteristika bzw. Einschränkungen, etwa bei der Bezeichnung von Tabellen

und ihren Attributen, mit Mehraufwand verbunden. Insbesondere bei der breiten, interdisziplinären Nutzung sollte die Semantik der Datenbankobjekte trotz der RDBMS-spezifischen Einschränkung bei deren Benennung direkt aus der Namensgebung erkennbar sein.

Die Umsetzung kann prinzipiell auf zwei Arten erfolgen:

1. Pragmatischer Ansatz:

   - Kategorische Trennung von struktureller und Werte-basierter Migration
   - Ausführliche Analyse des Quellmodells unter besonderer Betrachtung der zugrundeliegenden Datentypen und Integritätsbedingungen
   - Ableitung des Datentypmappings und Definition einer konsistenten Namenskonvention nach theoretischen (vgl. z.B. [KLINE et al., 2001]) und praktischen Grundsätzen (systemabhängig)
   - SQL-basierte (DDL) Redefinition des logischen Modells im Zielsystem entsprechend der Namenskonvention und des Datentypmappings
   - Migration der Daten in das Zielsystem, etwa durch ODBC-basierten Zugriff des Quellsystems auf das Zielsystem

   Dieser Ansatz ist essentiell, wenn ein Höchstmaß an Flexibilität in Bezug auf die individuelle Anpassung der Modellstruktur, insbesondere bei der Benennung der Constraints und Indizes, erforderlich ist.

2. Einsatz von Migrationstools

   - Automatische Generierung des logischen Modells des Zielsystem aus dem Quellmodell
   - Eingeschänkte manuelle Eingriffe in das Zielmodell bzgl. Datentypen und physikalische Speicherparameter möglich
   - Sukzessive oder vollständige Migration von Modellstrukturen und Daten

   Auf dem Markt verfügbare Migrationstools (z.B. Oracle Migration Workbench, [ORACLE (6), 2003]) bieten sich insbesondere zur schnellen Verfügbarmachung von in Fremdsystemen vorliegenden relationalen Sachdatenbeständen an. Je nach Umfang und Art der erforderlichen strukturellen Eingriffe kann auch ein kombinierter Ansatz aus 2. und 1. in Betracht gezogen werden.

Bei der Migration der vorliegenden Access-Datenbanken (vgl. Tab. 4.5) nach Oracle wurden, je nach Bedarf nach Erweiterung des konzeptionellen Modells, beide Ansätze verfolgt. Die Migration und Anpassung des Metadatenmodells, die den pragmatischen Ansatz erforderte, wird explizit in Abschnitt 4.2.4 erläutert.

Bezeichnung	Beschreibung	Anzahl der Relationen	Anzahl der Tupel
Sachdaten	Forstliche Inventur- und Begangsdaten der Jahre 1981, 1986, 1991, 1996 und 1998	21	394.320
Klimadaten	Klimadaten der Jahre 1947-2000, erhoben an 14 Wetterstationen	20	170.020
Liegenschaftsdaten (AGLB)	Daten des Automatisierten Grund- und Liegenschaftsbuchverfahrens	10	1.260
Verjüngung	Verjüngungsdaten der Jahre 1998-2000, erhoben in 24 Versuchsflächen	14	30.754

Tab. 4.5: Mengengerüst der migrierten Sachdatenbanken

Im Zuge der Übernahme der forstlichen Inventurdaten ist das konzeptionelle Datenbankmodell aufgrund struktureller Änderungen der zugrundeliegenden Forsteinrichtungsdatenbank des Bayerischen Staatsministeriums für Landwirtschaft und Forsten (BayStMLF) überarbeitet und modifiziert worden (vgl. [EICHINGER, 2003]). Der

aktuelle Datenbestand umfasst zudem die Parameter der letzten Waldinventur 2002. Die vollständige Abbildung des entsprechenden logischen Datenmodells enthält Anhang C.

Die übernommenen forstlichen und klimatologischen Sachdaten werden ergänzt durch touristische Fachdaten über das aktuelle Fremdenverkehrsangebot in den Nationalparkgemeinden. Das zugrunde liegende komplexe Sachdatenmodell ist mit geokodierten Adressen der Bayerischen Vermessungsverwaltung verknüpft und so konzipiert, dass aktuelle Veranstaltungen und Angebote von den örtlichen Betreibern eigenständig über das Web eingepflegt werden können. Das vollständige touristische Datenmodell ist in [NEUMEIER, 2004] dokumentiert.

4.2.4 Migration von Metadaten

Aus Anwendersicht sollten Metadaten sowohl direkt mit ihren Geodatensätzen verknüpft als auch zentral für das effiziente, anwendungsspezifische Auffinden von Geodatensätzen vorliegen. Das Metadatenkonzept des integrierten relationalen Datenhaltungsansatzes des Geodatenservers basiert daher aus der Kombination eines zentralen, internetfähigen Geodatenkatalogs und Datensatz-begleitenden Metainformationen (vgl. Abschnitt 4.1.2).

4.2.4.1 Migration und konzeptionelle Erweiterung des Metadatenmodells

Das Relationenmodell als Basis des zentralen, datenbankbasierten Metadatenkonzepts des Referenz-GIS stellt die Grundlage für die Übernahme von Metadaten in den Geodatenserver dar. Das Datenmodell ist gemäß des pragmatischen Migrationsansatzes aus Abschnitt 4.2.3 konzeptionell erweitert, um dem veränderten Datenhaltungskonzept (getrennt $\Longleftrightarrow$ integriert) unmittelbar zu genügen.
Die Neustrukturierung des konzeptionellen Metadatenmodells umfasst daher zunächst die Redefinition des physikalischen Zugriffs von absoluten Pfaden eines Dateisystems auf die vollständige Notation des den Nutzdaten zugeordneten Benutzerschemas einer Datenbank (vgl. Abb. 4.13).

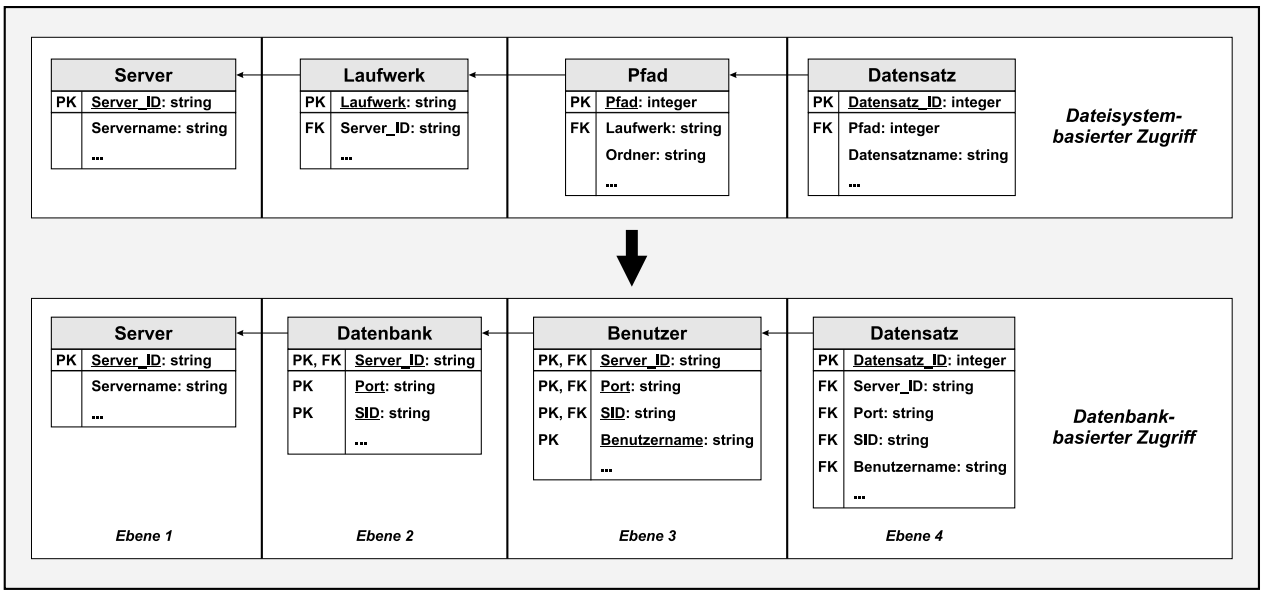

Abb. 4.13: Remodellierung des physikalischen Datenzugriffs

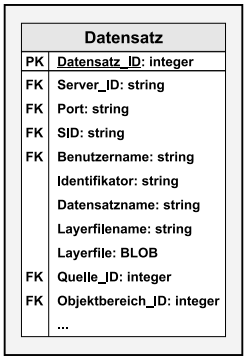

Abb. 4.14: Erweiterte Attributierung von Geodatensätzen

Für die in Abschnitt 4.3 beschriebene Implementierung einer Zugriffsschicht, die transparente, lesende und schreibende Zugriffe über das Internet unterstützt, ist die konzeptionelle Erweiterung der zentralen Metadatentabelle *Datensatz* erforderlich. Das Attribut *Datensatzname* nimmt die Bezeichnung der Geodatensätze entsprechend der Nomenklatur (Abschnitt 4.1.1.2) auf (z.B. *bdnbrlp2_0331*). Durch den 4-stelligen Identifikator (alphanummerischer Schlüssel) ist die eindeutige Zuordnung der Geometrie-Datensätze zu den Einträgen der Metadatenbank (zentraler Geodatenkatalog) gewährleistet.

Die effiziente Suche nach anwendungsrelevanten Geodaten erfordert jedoch die vollständige und semantisch aussagekräftige Bezeichnung aller Geodatensätze. Demnach beschreibt das Attribut *Layerfilename* einen Geodatensatz unmittelbar durch seinen Langnamen (z.B. *einzelbäume stehend 1998*), der dem als Basis der Migration vorliegenden Grafikformat (z.B. **.shp oder *.tif*) entspricht. Das Attribut *Layerfile* speichert ein Layerfile pro Geodatensatz als Binary Large Object (BLOB) direkt als grafische Metadateninformation.

Ein *Layerfile* verwaltet die kartographische Ausprägung eines Geodatensatzes sowie den entsprechenden physikalischen Zugriffspfad. Der Zugriff auf die Geometriedaten des Geodatenservers kann somit nicht nur direkt über die jeweiligen Business-Tabellen sondern - bei entsprechenden Zugriffsrechten - auch indirekt über die zugeordneten Layerfiles erfolgen (vgl. Abschnitt 4.3).

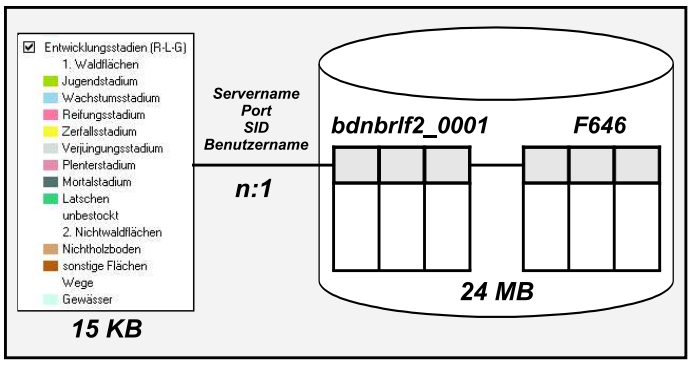

Abb. 4.15: Beziehung zwischen Layerfile(s) und Geometriedaten mit exemplarischen Speicherdimensionen

Zwischen Geodatensatz und Layerfile besteht generell eine 1:n-Kardinalität, d.h. es können theoretisch beliebig viele Layerfiles und damit mehrere kartographische Repräsentationen bzw. Klassifizierungen aus dem zugrunde liegenden Geodatensatz abgeleitet und genutzt werden, ohne dass die entsprechenden Geometriedaten redundant gehalten werden müssen.

Die Tabelle *Datensatz* stellt als Mastertabelle der Detailtabellen zur Modellierung von Herkunft und (geometrischer) Qualität der Geodatensätze die zentrale Relation des Metadatenbankmodells dar. Nach der konzeptionellen Erweiterung bietet sie die Grundlage für die Implementierung einer Zugriffsschicht, die den lesenden und schreibenden Online-Zugriff von Fachapplikationen unterstützt (vgl. Abschnitt 4.3). Die Abbildung des relevanten Teils des logischen Metadatenmodells enthält Anhang D.

4.2.4.2 Konvertierung von Datenbank-basierten Metadaten in XML-Daten

Seit geraumer Zeit bieten die Produkte führenden GIS-Hersteller die Funktionalität, - begleitend zu Geometriedaten - auch deren zugehörige Metadaten konform zu den führenden Metadatenstandards des FGDC und der ISO etwa XML-basiert zu verwalten. Die XML-Dateien werden beim integrierten Datenhaltungsansatz binär kodiert in Systemtabellen gespeichert und sind über Fremdschlüsselbeziehungen mit den Businesstabellen der Geometriedaten verknüpft. Für den Anwender ergibt sich dadurch eine erweiterte logische Sicht auf Geodaten und ihre beschreibenden, qualitativen Eigenschaften, was für die Selektion von relevanten Datensätzen essentiell ist. Nach Fortführung von Geodatensätzen werden *Implizite Metadaten* wie etwa das Minimum Bounding Rectangle (MBR) von der GIS-Software automatisch neu abgeleitet und die bisherige Metainformation aktualisiert (Synchronisation).

Im Zuge der Umsetzung des Metadatenkonzepts des GDS wurden auf VB-Basis gemäß einer von der GIS-Software vorgegebenen, FGDC-konformen Document Type Definition (DTD) Metadaten-Elemente (XML-Tags) erzeugt, diese mit den entsprechenden Werten der Metadatenbank versehen und als XML-Dateien ausgegeben. Im Anschluss erfolgte deren Zuweisung an die zugehörigen Geodatensätze durch explizite Imports.

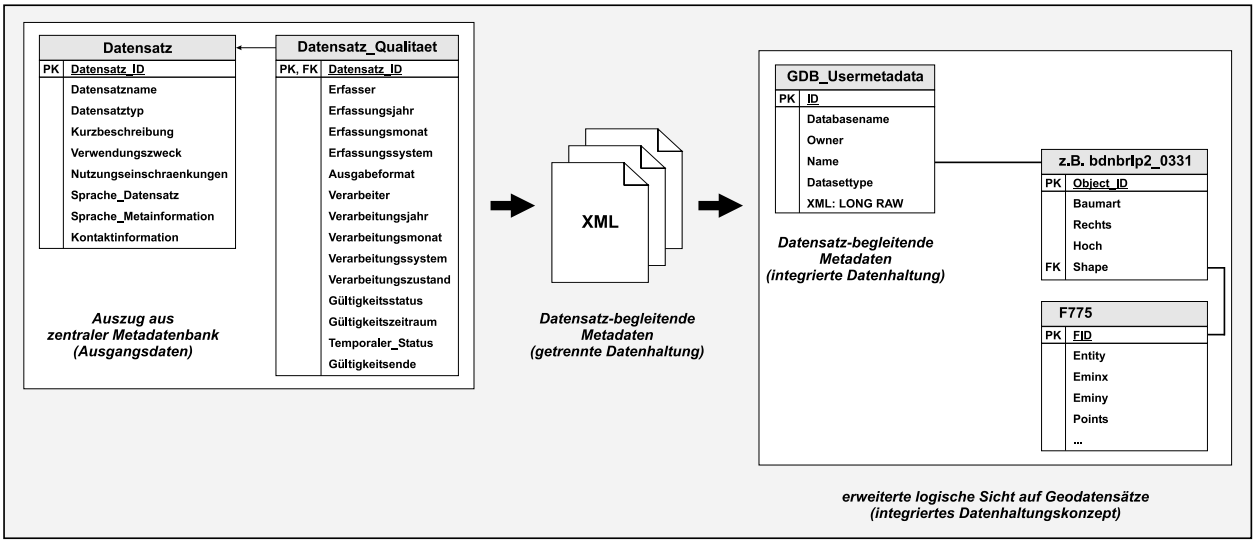

Abb. 4.16: Erzeugung Datensatz-begleitender Metadaten (XML) für ein Integriertes Datenhaltungskonzept

4.3 Implementierung des Zugriffkonzepts

Die dem entwickelten Referenz-GIS + zugrunde liegende GIS-Architektur beruht auf dem optimistischen Sperrkonzept für Geodatenbanken *(Optimistic Locking)*, d.h. schreibende Transaktionen zum Zwecke der Fortführung von Geodaten sind für jeden Benutzer zu jeder Zeit durch Versionierung möglich. Der integrierte Datenhaltungsansatz mit entsprechenden Systemtabellen (Add- und Delete-Tables) ermöglicht dabei die Einführung von benutzereigenen Versionen, ohne den Datenbestand redundant halten zu müssen. Für den Fall, dass nach Abgleich der Versionen Konflikte auftreten, sind diese durch die Benutzer zu lösen. Die Beurteilung der Wertigkeit einzelner Versionen im Konfliktfall obliegt somit den jeweiligen Benutzern selbst.
Allgemein erfolgt der Zugriff auf Geodaten klassisch über Thin-Clients (z.B. Geodaten-Viewer) bzw. Fat-Clients (Desktop-GIS) des GIS-Herstellers, wobei schreibende Zugriffe aus lizenzpolitischen Gründen lediglich über die High-End-Produkte des GIS-Herstellers möglich sind.
Alternativ besteht auch die Möglichkeit, via SQL und die Middleware-Software auf Geodatentabellen zuzugreifen.

Zur Transparenzsteigerung bei der Selektion von relevanten Geodaten einerseits und zur Minimierung des zentralen Fortführungsaufwands andererseits wurde der klassische Zugriff auf den Geodatenserver mit einer eigens entwickelten Zugriffsschicht kombiniert. Die Zugriffsschicht basiert unmittelbar auf den in Abschnitt 4.2.4 erläuterten Metadatenstrukturen, d.h. die Selektion von Datensätzen bei lesenden Zugriffen bzw. schreibende Transaktionen durch Fachanwender zur Erweiterung des Fachdatenbestandes erfolgt sowohl auf Grundlage von Metadaten der zentralen Datenbank (Geodatenkatalog) als auch der erzeugten XML-Dateien.

4.3.1 Realisierung lesender Zugriffe

Der lesende Zugriff auf Geodaten des Geodatenservers erfolgt über eine Client-seitige Java-Applikation, die auf Grundlage der gewährten Leserechte den Download von Layerfiles erlaubt und somit die anwenderfreundliche, individuelle Bereitstellung der verfügbaren Geodaten ermöglicht. Die Zugriffsberechtigungen werden nach der Einbringung neuer Fachdaten (schreibende Transaktionen, vgl. Abschnitt 4.3.2) vom jeweiligen Besitzer der Daten explizit und individuell für andere Benutzer (Projektpartner) in Form von Objektberechtigungen auf die Business- und Feature-Tabellen mittels der GIS-Software erteilt.

Die für (ganze) Tabellen gewährten Objektberechtigungen für registrierte Benutzer können aus der Data Dictionary View USER_TAB_PRIVS abgefragt und darauf aufbauend Geodatensätze benutzerspezifisch präsentiert werden. Auf Grundlage der thematischen Klassifizierung des zentralen Geodatenkatalogs werden die Layerfiles der Geodatensätze dann hierarchisch nach Datenquellentyp, Themenbereich, Datenquelle, Objektbereich und Geometrietyp strukturiert. Gleichzeitig können für jeden Datensatz die zugehörigen Datensatz-begleitenden

Metadaten aus der Systemtabelle GDB_USERMETADATA ausgelesen und präsentiert werden. Nach dem Download der Layerfiles lassen sich bei korrekter Datenbank-Verbindung die Geodaten visualisieren und ggf. lokal speichern, was für das Geoprocessing insbesondere bei begrenzter Bandbreite bedeutsam ist.

Die Kombination der beiden Metadaten-Ansätze sorgt demnach für die bestmögliche Strukturierung des Datenangebots einerseits und für die aus Anwendersicht wünschenswerte erweiterte logische Sicht der Geodaten und ihrer Metadaten andererseits. Der zentrale Geodatenkatalog kann alternativ auch direkt über das Internet via SQL bzw. ODBC-basiert über die GIS-Software oder geeignete Tools von Drittanbietern angefragt werden. Entsprechendes gilt für den Zugriff auf komplexe Sachdaten des Geodatenservers.

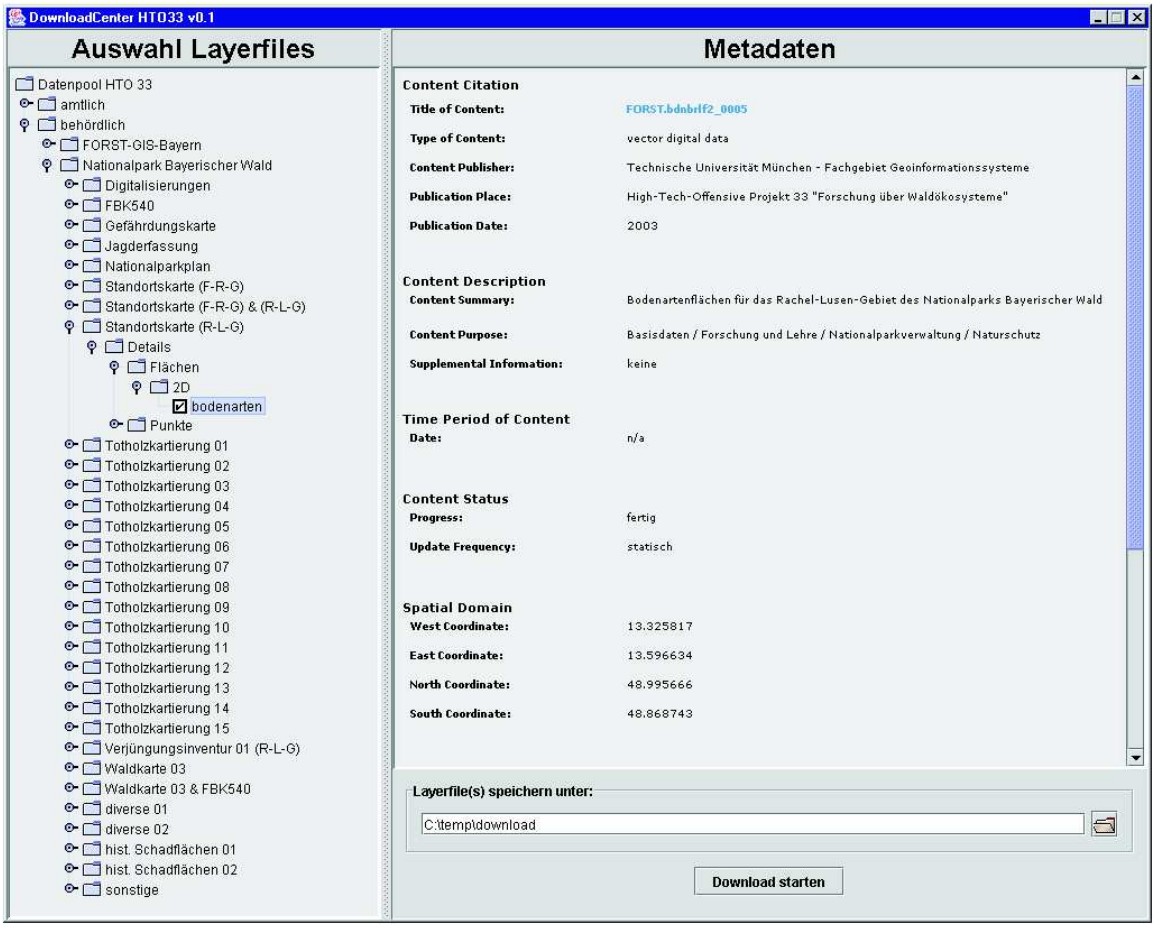

Abb. 4.17: Metadaten-gestützte Selektion von relevanten Geodaten

4.3.2 Realisierung schreibender Zugriffe

Zur Sicherung und nachhaltigen Nutzung der Erkenntnisse aus forstwissenschaftlichen Analysen und Auswertungen sowie zur Erweiterung der Datenbasis über das Waldökosystem des Nationalparks sollen die Ergebnisse in Form neuer Geo- und Sachdatensätze in den Geodatenserver eingepflegt werden.

Hierfür wurde eine weitere Java-Applikation entwickelt, die eigene schreibende Transaktionen gezielt unterstützt. Auf diese Weise wird die Web-Fähigkeit der GIS-Plattform dazu genutzt, um den Prozess der Erweiterung des Datenpools zu dezentralisieren und auf die hinsichtlich der Beschreibung der Fachdaten einzig kompetenten Anwender zu übertragen. Von den jeweiligen Fachanwendern sind dazu in vordefinierter Form die zur Beschreibung des jeweiligen Geodatensatzes wesentlichen Metainformationen als datensatzbegleitende Metadaten anzugeben. Diese werden über die Java-Applikation automatisch in die zentrale Metadatenbank (Geodatenkatalog) übertragen, um für nachfolgende lesende Zugriffe Dritter unmittelbar berücksichtigt werden zu können.

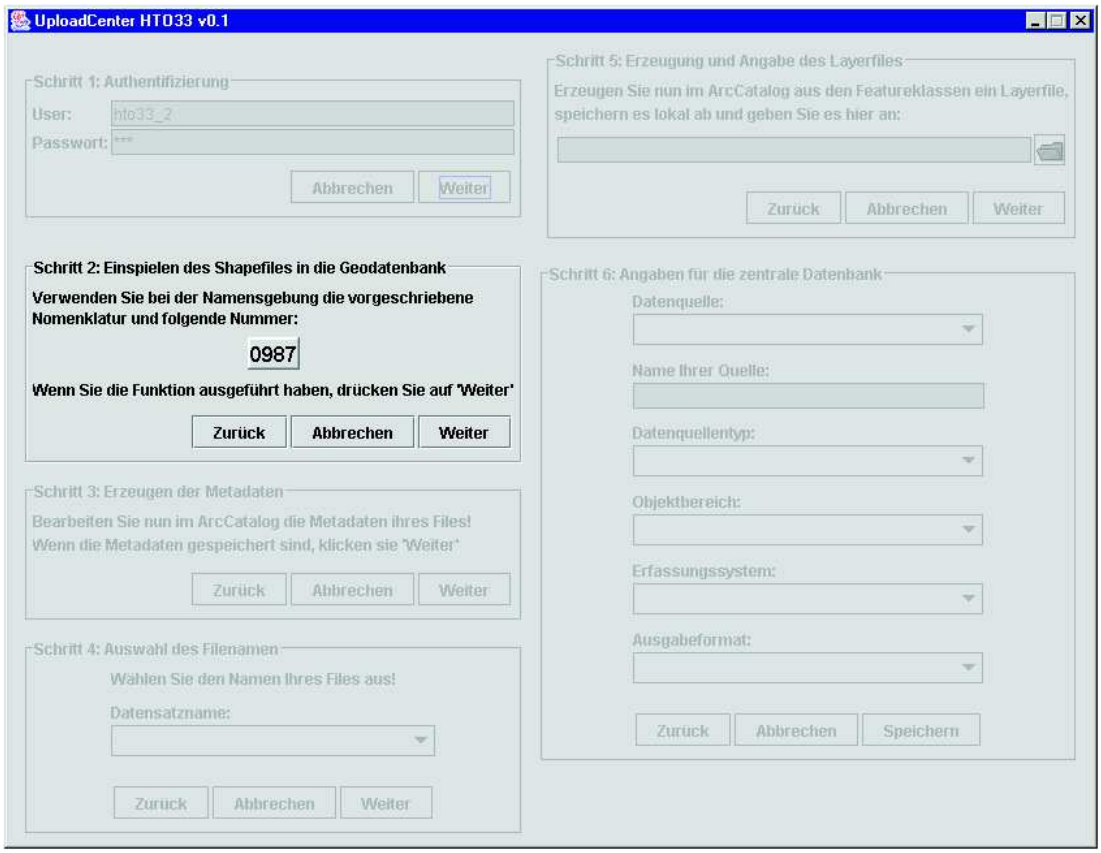

Abb. 4.18: Eigenständige Einbringung von Analyseergebnissen in den Geodatenserver (Referenz-GIS +)

4.4 Ausgewählte Anwendungen des RDBMS-basierten Datenhaltungsansatzes

Neben deutlich verbesserten Voraussetzungen zur breiteren Nutzung und intelligenten Fortführung der Geodatenbasis, die ein integriertes relationales Datenmanagement bietet, ist vor allem der Mehrwert für die Anwendungsentwicklung von Interesse. Die im Vergleich zum getrennten Datenhaltungskonzept erweiterten Möglichkeiten des integrierten Ansatzes zur realitätsnahen Modellierung von (räumlichen) Objekten basieren primär auf der Tatsache, dass sich durch die Abbildung von Geometrieobjekten auf Tabellenstrukturen Constraints und Trigger auch auf die Attribute von Geodaten anwenden lassen. Somit können bekannte Beziehungen zwischen Geometrieobjekten persistent in der Datenbank gespeichert und referentielle Integritäten etwa über die ON DELETE CASCADE-Klausel kontrolliert werden. Der folgende Abschnitt zeigt am Beispiel des Wegenetzes im Nationalpark, welches Potential der integrierte relationale Ansatz hinsichtlich der Modellierung von (raumbezogenen) Objekten, insbesondere zu deren konsistenten Fortführung bietet.

4.4.1 Konsistente Fortführung des Wegenetzes im Nationalpark Bayerischer Wald

Ausgangssituation:

Das Straßen- und Wegenetz im Nationalpark Bayerischer Wald ist sehr ausgedehnt und vielfältig. So steht den Besuchern zur Erholung ein Netz von mehr als 300 km gut markierten Wanderwegen, fast 200 km Radwegen und rund 80 km Skiloipen zur Verfügung. Das Gesamtnetz umfasst neben den öffentlichen Verkehrswegen also eine Reihe weiterer Widmungen wie Rücke-, Wander-, Reit- und Radwege sowie Loipen und Forststraßen, die es vollständig, hinsichtlich der Fortführung effizient (redundanzfrei und konsistenzerhaltend) und für GIS-Anwendungen flexibel im GIS zu modellieren gilt.

Lösungsansatz:

Da viele Wege nicht ausschließlich einer Klasse sondern zwei oder mehreren Widmungen zugeordnet werden können, ist die Verwaltung als eigenständige Layer nicht a priori zwingend. Vielmehr ist die Aggregation von Wegen unterschiedlicher Widmungen zu einer übergeordneten Klasse, z.B. *Wege*, und die Ausscheidung von Straßenarten durch die Definition von Subtypen auf Basis identischer Attributwerte eines Subtypfeldes sinnvoll. Auf diese Weise lassen sich Subtypen verschiedener Straßenarten bzw. Interessensgruppen flexibel miteinander kombinieren. Beispielhaft seien hier Reitwege genannt, die zugleich auch als Radwege genutzt werden. Durch die Ausscheidung von Kombinationen aus unterschiedlichen Straßentypen kann unmittelbar das Konfliktpotential des Wegenetzes dargestellt werden, da auf diesen Wegstrecken unterschiedliche Interessensgruppen aufeinander treffen. Zudem wird das gesamte Straßen- und Wegenetz redundanzfrei gespeichert, wodurch der Fortführungsaufwand minimiert und das Risiko bzgl. Inkonsistenzen reduziert wird.

Typischen oder quasistatischen Attributen von Straßentypen wie etwa deren Straßenbelag oder -breite werden Default-Werte oder Bereichs-Domänen (Range Domains) bzw. Wertelisten (Coded-Values) zugewiesen, die den Datenbestand bei der Fortführung automatisch auf Regelkonformität überprüfen (Check-Constraints). Auf diese Weise lassen sich Fehler bei der Attributeingabe grundsätzlich ausschließen.

Bei Verlängerung oder Verkürzung von z.B. Radwegen werden deren numerische Attribute wie etwa die mittleren Instandhaltungskosten durch die Definition von Teilungs- bzw. Vereinigungsmethoden (Split, Merge Policies) prozentual in Abhängigkeit der neuen Weglänge berechnet.

Mehrwert im Vergleich zu getrennter Datenhaltung:

Die integrierte Verwaltung von Geometrie und direkten Attributen in relationalen Datenbanktabellen eröffnet die Möglichkeit zur Nutzung des Potentials der aktiven Datenbank (Constraints und Trigger) für Geodaten. Im vorliegenden Anwendungsszenario kann das Wegenetz durch Subtypenbildung bereits auf logischer Modellebene klassifiziert und damit realitätsnäher, redundanzfrei und persistent in der Datenbank abgebildet werden. Während Check-Constraints die stetige attributive Konsistenz der Vektordaten garantieren, ermöglichen AFTER UPDATE Trigger nach geometrischen Änderungen am Wegenetz die automatische Neuberechnung von numerischen Attributwerten auf Grundlage von vom Anwender explizit definierten Beziehungen zwischen Geometrie und Attributen.

4.4.2 Modellierung des Igelbussystems im Nationalpark

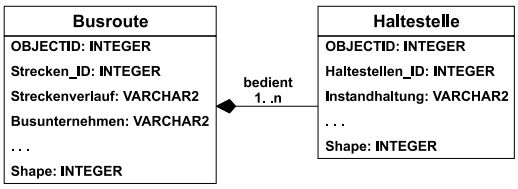

Abb. 4.19: Existentielle Abhängigkeit zweier Geoobjekt-Klassen

Ausgangssituation:

Um das Verkehrsaufkommen und damit die Schadstoffbelastung bzw. überfüllte Parkplätze im Nationalpark zu vermeiden, sind bestimmte Straßen für den Autoverkehr gesperrt. Stattdessen sind seit 1996 umweltfreundliche, Erdgas-betriebene sog. Igelbusse im Einsatz, die die Besucher von ihrem Urlaubsquartier in das Schutzgebiet bringen. Die Busse bewegen sich dabei auf festen Routen und fahren bestimmte Haltestellen an. Das Datenmodell eines GIS sollte verschiedene Geoobjekt-Klassen nicht nur singulär abbilden sondern auch deren Beziehungen, die in diesem Szenario in Form existentieller Abhängigkeiten zwischen den Haltestellen und seinen zugeordneten Busrouten besteht.

Lösungsansatz:

Existentielle Abhängigkeiten zwischen Geoobjekt-Klassen, wie hier den Bushaltestellen und den Routen des Igelbussystems, können bei integrierter, relationaler Datenhaltung unmittelbar in der Geodatenbank in Form von persistenten, benutzerdefinierten Beziehungen modelliert werden. Die Beziehung ist nichtattributiv, d.h. die Beziehung basiert nicht auf Attributwerten sondern auf der Abhängigkeit an sich: Eine Bushaltestelle kann nicht ohne das Wegenetz existieren.
Zur exakten Modellierung der Beziehung reichen die einfachen Kardinalitäten oft nicht aus, deshalb kann die Beziehung durch Regeln präzisiert, etwa auf Subtypen begrenzt werden. Weiterhin können der Beziehung selbst ebenfalls Attribute zugeordnet sein.

Mehrwert im Vergleich zu getrennter Datenhaltung:

Die Möglichkeit zur persistenten Modellierung von Beziehungen zwischen Instanzen unterschiedlicher Geoobjekt-Klassen (Busrouten und Haltestellen) eröffnet die Chance zur ganzheitlichen Abbildung des jeweiligen Realwelt-Ausschnitts. Werden nach der Definition des Geodatenmodells neue Instanzen einer Geoobjekt-Klasse erzeugt, so prüft das GIS die definierten Abhängigkeiten und erzwingt die Verknüpfung mit einer bereits vorhandenen Instanz der in Beziehung stehenden Klasse bzw. fordert das Anlegen einer neuen Instanz dieser Klasse.

4.4.3 Dynamische Kartographische Visualisierungen

Auf Grundlage der erzeugten relationalen Datenstrukturen sind eine Reihe von Kartographischen 2D- und 3D-Darstellungen entstanden, die das breite Anwendungsspektrum der Geofachdatenbasis wiederspiegeln. Infolge des stark proprietären Relationenmodells, in das Geodaten bei der Migration abgebildet werden, lassen sich Visualisierungen ausschließlich mit GIS-Produkten des GIS-Herstellers erzeugen. Selbst entwickelte *Thin Clients* zur Visualisierung von Geodaten, die etwa auf Java basieren, nutzen zur Interpretation der Datenstrukturen ebenfalls die Objektbibliotheken des GIS-Herstellers.

Bereits bestehende Visualisierungen, die auf den klassischen Geometrieformaten basieren, sind nach dem Migrationsprozess weiterhin vollständig nutzbar, da das veränderte physikalische Modell der Geodaten explizit durch die GIS-Software berücksichtigt werden kann. Spezifische Eigenschaften von Visualisierungen wie etwa die maßstabsabhängige Darstellung oder die Klassifizierung von Geobjektklassen sind Metadaten der entsprechenden Kartenwerke und damit vollständig unabhängig vom gewählten Datenhaltungsansatz.
Der hohe Aufwand, der im Rahmen der empirischen Untersuchungen bei der Wiederherstellung bereits vorhandener Visualisierungen - primär bei der Generierung der exakten Symbolik - entstand, ist vielmehr im Wechsel der GIS-Produktfamilie mit vollständiger Neuausrichtung der Entwicklungsstrategie (u.a. modulare Architektur bestehend aus Bibliotheken aus Software-Komponenten und Integration bedeutender IT-Standards [ESRI, 2004]) durch den GIS-Hersteller begründet.

4.5 Fazit

Nach langer Vorherrschaft proprietärer GIS-Dateiformate hat sich die zuverlässige und performante Verwaltung von Geodaten in Datenbanken insbesondere durch die Möglichkeit zur komprimierten Abbildung von Vektor- und Rastergeometrien auf binäre Basisdatentypen eines RDBMS sowie durch effiziente Indizierungsmechanismen und verfügbare, darauf angepasste Funktionalität zur Durchführung von raumbezogenen Abfragen etabliert. Der unmittelbare Nutzen für die Geoinformatik ist vielfältig:

1. **Breitere Nutzung der Geodaten**

   Die Benutzerverwaltung und die Möglichkeit zur Zuweisung von System- und Objektberechtigungen in Datenbanken sind Voraussetzung zur gezielten Distribution von Daten in Netzwerken (LAN/WAN). Die Folge ist ein Quantensprung bei der breiten Nutzbarmachung von Geodaten. Gefördert wird dies durch die synchrone Speicherung von Metadaten in BLOBS, die die vorgehaltenen Geodatensätze insbesondere qualitativ beschreiben und damit die logische Gesamtsicht des Anwenders um eine elementare Ebene erweitern.

2. **Bessere funktionale Abbildung konzeptionaler Anforderungen in die Datenbank**

   Durch die Möglichkeit, Constraints auf direkte Attribute von Geoobjektklassen anzuwenden, wird die Lücke zwischen realer Welt und dem logischen Modell der Abbildung zu einem erheblichen Anteil geschlossen. Anstelle von Punkten, Linien und Flächen werden deutlicher als bei einfachen Grafikformaten „reale“ Geoobjekte wie etwa Inventurpunkte, Radwege oder Waldbestände modelliert.
   Die genauere, logische Modellierung von Geoobjekten und ihren Beziehungen wie die Definition von existentiellen Abhängigkeiten zwischen Instanzen von Objektklassen stellt einen ersten bedeutenden Schritt zur ganzheitlichen Abbildung von Realwelt-Ausschnitten dar. Geodatensätze müssen nicht mehr singulär in

der Datenbasis vorgehalten werden, sondern sind zu einem Gesamtmodell verknüpft. Durch die Definition von Gültigkeitsbereichen ergibt sich gleichzeitig eine massive Steigerung der attributiven Konsistenz.

3. **Bündelung von Geoobjekt-Komponenten zur Förderung des transparenten Zugriffs**

   Durch die Speicherung der Nutzdaten und ihrer Metadaten sowie zugehöriger System-interner Informationen (Benutzer, Zugriffsrechte, etc.) in Relationen mit flacher Extension und einfachen Sichten können Geoobjekt-Komponenten relativ leicht gebündelt und dem GIS-Anwender individuell als logische Gesamtsicht präsentiert werden. Die Zusammenführung der einzelnen Ebenen auf Basis des Relationenmodells ermöglicht die Entwicklung anwendungsspezifischer Benutzerschnittstellen zur transparenten Selektion bzw. kontrollierten, dezentralen Fortführung von Geodatenbeständen.

4. **Verstärkung der Herstellerabhängigkeit**

   Bei der Abbildung von GIS-Dateiformaten in das Relationenmodell des GIS-Herstellers ESRI wird die Systemabhängigkeit gewissermaßen noch verstärkt, da sich die entstehenden Datenstrukturen nur durch die GIS-Produkte des Herstellers bzw. deren Objektbibliotheken ausreichend interpretieren lassen. Zur Nutzung der Geodaten in einem Fremdsystem müsste demnach zunächst eine rückwärtige Konvertierung in das Ausgangs-Grafikformat erfolgen, das im Falle eines Industriestandards dann direkt vom Zielsystem gelesen oder zumindest in ein geeignetes Zielformat konvertiert werden kann.
   Die Hersteller-Abhängigkeit wird besonders offensichtlich bei der Betrachtung der Konsistenz zwischen Geometrie und direkten Attributen von Geoobjektklassen. Diese kann einzig durch den Einsatz der GIS-Software, nicht jedoch im Falle des direkten Zugriffs via SQL dauerhaft garantiert werden, was durch die Verteilung von Attributen und Geometrie auf verschiedene Tabellen noch verstärkt wird.

5. **Unzureichende Modellierbarkeit komplexer Objekte**

   Die Kernanforderung, die effiziente Modellierbarkeit komplexer Objekte im Allgemeinen und aller in Abschnitt 2.2 aufgeführten Geoobjekt-Komponenten im Speziellen, wird durch die RDBS-basierten Datenbankkonzepte nicht ausreichend erfüllt. So wird etwa der exakten Beschreibung der fachlichen Bedeutung von Geoobjekten in der aktuellen GIS-Software, die auf relationalen Speicherstrukturen aufsetzt, bislang nur innerhalb der Kurzzusammenfassung der Datensatz-begleitenden Metadaten Rechnung getragen, d.h. diese ist lediglich Teil des Binärstroms der Metadaten und kann nicht gezielt über SQL angefragt werden. Volltext, der klassisch als Hotlink mit den Geodatensätzen verknüpft ist und ebenfalls unstrukturiert bleibt, stellt deshalb ebenfalls keine wirkungsvolle Alternative dar.

6. **Mangelnde Strukturierbarkeit großer Geodatenbestände**

   Der konkreten Anforderung, große Geodatenbestände möglichst flexibel strukturieren zu können, werden flache Tabellen als einzig verfügbare physikalische Ebene im Relationenmodell nur unzureichend gerecht. Stattdessen muss die Klassifizierung nach thematischen oder regionalen Aspekten auf Benutzer- bzw. Datenbankinstanz-Ebene stattfinden.

Eine ausführliche Bewertung der Leistungsfähigkeit des Relationenmodells für das effiziente Management von Geodaten insbesondere im Hinblick auf die Adaption der konzeptionellen Anforderungen aus Abschnitt 2.2 sowie ein Vergleich mit dem Potential des objektrelationalen Ansatzes erfolgt in Kapitel 6.

Kapitel 5

Migration getrennt gehaltener Geodaten in ein Objektrelationales Datenbanksystem

Die Abbildung von proprietären GIS-Dateiformaten auf das Relationenmodell eines Herstellers führt wie in Kapitel 4 aufgezeigt zu bedeutenden Leistungssteigerungen der Geodatenhaltung. Dies gilt speziell für die Modellierbarkeit von Geoobjekten und deren Beziehungen sowie die Verfügbarmachung von Geodaten. Der relationale Ansatz offenbart aufgrund seines proprietären Datenbankmodells noch Schwächen hinsichtlich des offenen Zugriffs, bzw. erfordert zur vollständigen Erhaltung der Konsistenz zwischen Geometrie und direkten Attributen den zwingenden Einsatz der entsprechenden GIS-Hersteller-Software.

Durch die Nutzung der von den ORDBMS-Herstellern angebotenen Strukturierten Typen (Objekttypen) zur direkten Speicherung von Vektordaten in Tabellen kann dieses Defizit zum Teil überwunden werden. Neue, auf Grundlage der Erweiterbarkeit des Typsystems entstandene Basisdatentypen etwa zur Speicherung von XML-Daten in der Datenbank bieten sich für die erweiterte Modellierung von Geoobjekten an. Für den Einsatz von ORDBS als Basis der Datenhaltung spricht ihre hohe Flexibilität bei der direkten Modellierung von komplexen Sachdatenobjekten und deren Beziehungen.

Mit dem Hype, der ab Mitte der 90er Jahre rund um die Objektrelationale Datenbankwelt entstand, wurde die Integration von ORDBMS in GIS zu einem der Forschungsschwerpunkte der Geoinformatik. Mittlerweile ist die anfängliche Begeisterung über OR merklich gewichen, was nicht zuletzt auf die Entwicklung von XML hin zum de facto Standard für die Beschreibung komplexer Strukturen zurückzuführen ist (vgl. [SCHEUCH et al., 2004]). Die führenden GIS-Hersteller unterstützen bislang lediglich Objekttypen zur Speicherung von Vektorgeometrien, das weitere in den Abschnitten 3.4 und 3.5 aufgezeigte Modellierungspotential bleibt aber noch gänzlich ungenützt. In Bezug auf die für GIS wichtige Rasterdatenverwaltung ist bereits die Unterstützung der entsprechenden Basisdatentypen von Seiten der führenden GIS-Hersteller angekündigt, so dass nun auch Rasterdaten sowohl über die GIS-Software als auch direkt über die SQL-Schnittstelle des ORDBMS gelesen und geschrieben werden können.

Das folgende Kapitel 5 beschreibt die Migration des vorliegenden Ausgangsdatenbestandes eines Desktop-GIS (Referenz-GIS) in ein integriertes objektrelationales Datenhaltungskonzept (Referenz-GIS ++) nach Abschnitt 2.1 (Ansatz 4) (vgl. Abb. 5.1) unter besonderer Betrachtung der dabei entstehenden Datenbankstrukturen. Es werden mögliche Schwerpunkte für die erweiterte Nutzung von ORDBMS in der Geoinformatik, die noch bestehenden Defizite bei der Integration in GIS sowie der generelle Leistungsgewinn dieses Ansatzes thematisiert. Ferner wird das Analysepotential, das der ORDBS-basierte Ansatz dank der implementierten räumlichen Operatoren und Funktionen bietet, am Beispiel konkreter GIS-Anwendungen im Testgebiet Nationalpark Bayerischer Wald demonstriert.

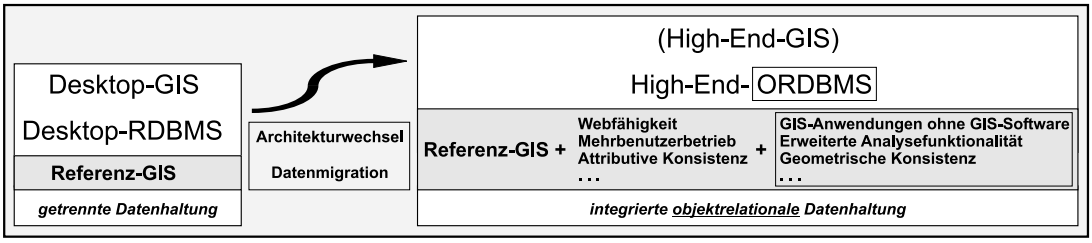

Abb. 5.1: Leistungsgewinn durch integrierte, objektrelationale Datenhaltung

5.1 Übernahme von Datenbeständen

5.1.1 Migration von Vektordaten

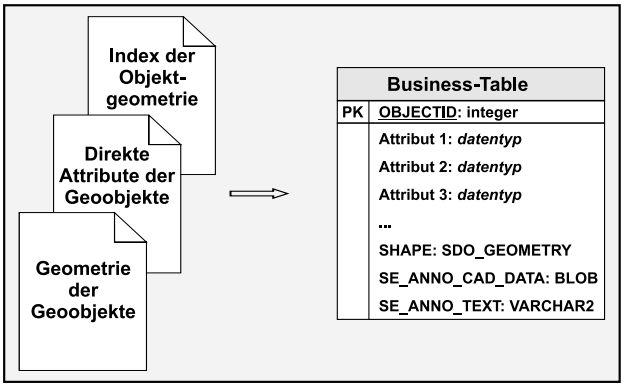

Abb. 5.2: Abbildung von Vektorformaten auf *Spatial Tables*

Alternativ zum eigenen binären Speicherformat *(SDEBINARY)* bietet die im Rahmen dieser Arbeit eingesetzte Spatial Database Engine von ESRI (ArcSDE) die direkte Unterstützung der räumlichen Objekttypen der ODBMS Oracle, IBM DB2 und IBM Informix für die Speicherung von Vektordaten an. Die Middleware-Komponente ist dazu vorab bzgl. des gewünschten Speicherformats zu konfigurieren.
Bei der Konvertierung des Vektorformats in objektrelationale Datenbankstrukturen werden die Stützpunkte der Geometrie als Koordinaten-Array zusammen mit den direkten Attributen eines Geoobjekts in einer Tabellenzeile gespeichert. Im Datenmodell des GIS-Herstellers ESRI existiert demnach keine Feature-Tabelle wie beim integrierten relationalen Ansatz unter Verwendung des Datentyps SDEBINARY, was sich positiv auf die Konsistenzerhaltung bei offenen Zugriffen via SQL auswirkt. Abb. 5.2 skizziert die im Rahmen dieser Arbeit aus ESRI Shape-Dateien via ArcSDE erzeugten *Spatial Tables* auf Basis des Oracle-Objekttyps SDO_GEOMETRY.
Oracle Spatial unterstützt bislang nicht das Speichern von Betextungen (Annotation), Splines oder CAD-Daten. ArcSDE unterstützt dagegen Shape-Dateien, die CAD-Information und ein spezielles Speicherschema für Annotation beinhalten, das noch in früheren SDE-Versionen Verwendung fand ([ESRI (2), 2005]).
Um den Zugriff von CAD-Clients auf sämtliche eigenen Formate in der Datenbank über ArcSDE zu garantieren, werden im objektrelationalen Speichermodell zusätzliche Binär- und Text-Felder erzeugt. Beim relationalen Ansatz werden diese Informationen im Binärstrom der Feature-Tabelle, zusammen mit der Geometrie abgelegt (vgl. Abschnitt 4.2.1).

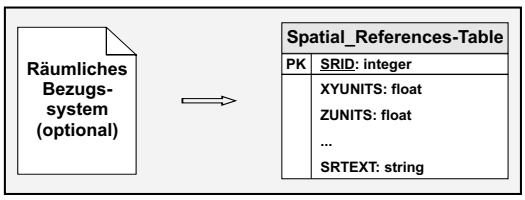

Abb. 5.3: Abbildung des räumlichen Bezugssystems

Die Parameter der ASCII-Datei zur Definition des geodätischen Datums und der Projektion werden bei der Konvertierung auf Datenbankstrukturen analog zum relationalen Ansatz im Feld *SRTEXT* der Tabelle „Spatial_References“ gespeichert. Maßstab und Längeneinheiten werden in entsprechenden Feldern verwaltet. Die Verknüpfung mit der Business-Tabelle erfolgt über das Primärschlüssel-Feld *SRID* und der übergeordneten Tabelle „Layers“, in der alle Vektordatensätze registriert sind. Ist die Information bzgl. des räumlichen Bezugssystems eines Geodatensatzes auf Fileebene nicht verfügbar, so kann diese auch nach erfolgter Migration durch die GIS-Software explizit zugewiesen werden.

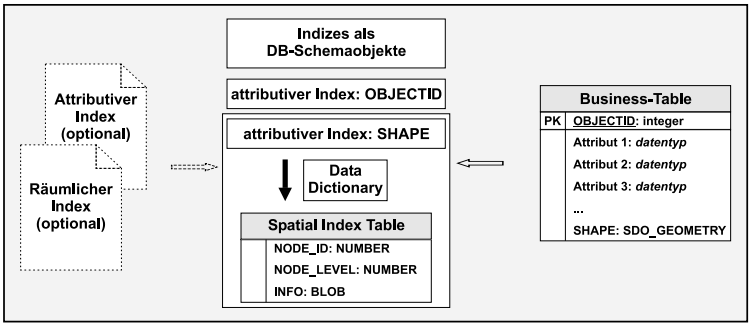

Abb. 5.4: Räumliche und attributive Indizierung

Die zur räumlichen Indizierung verfügbaren Indextypen sind unmittelbar abhängig vom zugrunde liegenden ORDBMS. Oracle Spatial bietet neben dem gängigen R-Tree, der die Hierarchie auf die Minimum Bounding Rectangles (MBR) der Einzelgeometrien einer Geoobjekt-Klasse indiziert, wahlweise einen linearen Quadtree oder ein hybrides Indizierungsverfahren an, die eine automatische Kachelung eines Layers bewirken, bis die kleinste Kachelgröße bzw. die maximale Kachelanzahl pro Einzelgeometrie erreicht ist (vgl. [ORACLE (1), 2002]).

Abb. 5.4 skizziert die bei der Indizierung mittels R-Tree entstehenden Datenbankstrukturen. Dabei werden analog zum relationalen Ansatz die Felder OBJECTID und SHAPE von der GIS-Software klassisch indiziert. Die Information des R-Tree wird in einer separaten Tabelle gespeichert, die im Datenbankschema des jeweiligen Benutzers liegt und deren Bezeichnung sich über Abfrage der Data Dictionary View *USER_SDO_INDEX_METADATA* ermitteln lässt.

Zur Unterstützung des Mehrbenutzerbetriebs werden attributive und geometrische Änderungen nicht direkt in der Business-Tabelle gespeichert sondern bis zum Abgleich verschiedener Versionen in separaten *Add-* und *Delete-Tabellen* geführt (vgl. Abschnitt 4.2.1).

Aufgrund seiner weiten Verbreitung sind für den Industriestandard ESRI-Shape einige Konverter (z.B. *JShapeXi* oder das Oracle-Tool *shp2sdo*) verfügbar, mit denen sich derartige Spatial Tables auch ohne den Einsatz der GIS-Software direkt erzeugen und befüllen lassen. Der Konvertierungsprozess ist beim zuletzt genannten Werkzeug allerdings etwas umständlich und deshalb auch zeitaufwendiger. Der Konverter erzeugt lediglich SQL-DDL-Skripte, Control- und Daten-Dateien (enthalten die Stützpunkte der Geometrie), die anschließend vom Anwender zum Erzeugen und Befüllen der Tabellen genutzt werden können. Das Befüllen der Tabellen erfolgt dabei mittels SQL*Loader, einem Oracle-Standardwerkzeug zum Einladen von großen Datenmengen, die als ASCII-Daten vorliegen. Damit solche auf dieser Basis erzeugten Spatial Tables, die als sog. *Third Party Geometries* bezeichnet werden, mit der GIS-Software gelesen und geschrieben werden können, ist die Registrierung der Tabellen im SDE-Datenbankmodell über entsprechende Administration-Commands erforderlich.

5.1.2 Rasterdatenverwaltung in Oracle 10g

Oracle hat im ersten Release ihrer Datenbank 10g mit der objektrelationalen Verwaltung und Verarbeitung von Rasterdaten auf Basis eines speziellen Datentyps Neuland betreten. Die Speicherung und Manipulation von Rasterdaten auf Basis des Objekttyps SDO_GEORASTER erfolgt bislang fast ausschließlich über SQL und entsprechender Methoden und wird noch nicht direkt von den GIS-Herstellern unterstützt. Von Seiten des GIS-Herstellers ESRI wurde dies für die kommenden Releases der ArcSDE angekündigt, Intergraph-Produkte können Dank der Kooperation mit Drittanbietern (Keigan Systems) auf Rasterdaten in Oracle 10g zugreifen (GeoMedia Grid for Oracle).
Auf Basis des neuen Datentyps können georeferenzierte Orthophotos, RGB-Bilder, Multi-Band-Satellitenbilder und GRID-Daten zusammen mit ihren Metadaten gespeichert werden. Der übergeordnete Objekttyp SDO_GEORASTER nutzt dazu unmittelbar die drei untergeordneten Objekttypen

- SDO_RASTER: *Speichert die Pixel-Daten der Blöcke eines Rasterbildes als BLOB*
- SDO_GEOMETRY: *Speichert MBRs des Rasterbildes, seiner Datenblöcke und Pyramiden*
- XMLTYPE: *Speichert sämtliche Metadaten über ein Rasterbild, inkl. seiner Geocodierung (Passpunkte als Koordinaten-Array) als XML-Dokument (vgl. 5.1.4)*

Die Methodenimplementierung zum Objekttyp SDO_GEORASTER umfasst u.a. Funktionen zur:

- *Berechnung von Pyramiden: Speichern von Bildern geringerer Auflösung zu Gunsten einer schnelleren Darstellung bei kleineren Maßstäben*
- *Selektion von Raster-Subsets, die zu einem BLOB mosaikiert werden können*
- *Ausgabe des MBR eines Rasterbildes*
- *Anzeige und Bearbeitung von Metadaten*

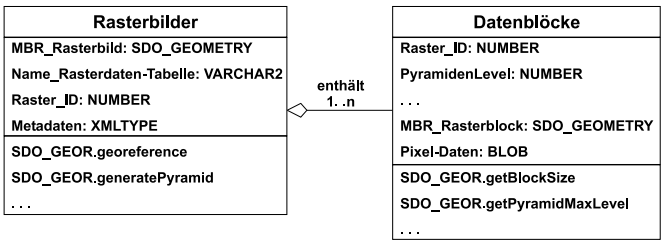

Abb. 5.5: UML-Darstellung des Rasterdatenmodells von Oracle 10g

Zu Gunsten einer effizienteren Speicherverwaltung und zur Beschleunigung des Zugriffs werden Rasterbilder gekachelt (benutzerdefinierte Kachelgröße) und die entstehenden Datenblöcke binär in Rasterdaten-Tabellen abgelegt [ORACLE (5), 2003]. Weitere Konsequenz dieser Kachelung ist, dass für Rasterobjekte ausser dem verfügbaren Plattenspeicher keine Datentyp-spezifische Größenbeschränkung besteht, da ein einzelner Datenblock stets unter der Maximalgröße eines BLOB (4GB) bleibt.

Die einfachste Methode, um Rasterbilder zu speichern, ist die binäre Codierung von schwarzen und weissen Farbpigmenten, die in Form der Zahlenwerte 0 und 1 organisiert sind. Um farbige Rasterbilder zu verwalten kann die Intensität der roten, grünen und blauen Farbanteile von einzelnen Bildpunkten gespeichert werden. Der Aufbau von BLOB-Objekten kann (wie auch die Struktur von Rasterdateien) als sequentielle Byte-Folge (BAND) aufgefasst werden ([HOCHHÄUSLER, 2001]). Im unkomprimierten Zustand lassen sich Rasterobjekte in drei verschiedenen Varianten organisieren: Der BIP-Struktur (Band Interleaved by Pixel), der BIL-Struktur (Band Interleaved by Line) und der BSQ-Struktur (Band SeQuential) ([ORACLE (2), 2004]). Bei der grafischen Darstellung (bzw. Interpretation der Daten) wird die Farbseparation durch die Überlagerung der Farben wieder aufgehoben.

Die Bezeichnung, Metadaten und Geocodierung des gesamten Bildes wird in einer übergeordneten Raster-Tabelle verwaltet. Ein Trigger sorgt für die referentielle Integrität zwischen Raster-Tabellen und Rasterdaten-Tabellen [ORACLE (2), 2004].
Der Import und Export von Rasterdaten erfolgt klassisch via SQL und spezifischer Methoden, zusätzlich werden von Oracle eigene Java-Tools angeboten, die auf diesen Methoden aufsetzen (GeoRasterLoader, GeoRasterExporter). Oracles Strategie zielt hier nach eigenen Aussagen aber verstärkt auf die von Partnern angebotenen Werkzeuge (z.B. JShapeXi+ (GDV-Software), GeoRaster ETL for Oracle (PCI Geomatics), Keigan Grid (Keigan Systems) [FRANCICA, 2004]).

Die einfache Visualisierung von mittels SDO_GEORASTER gespeicherten Rasterdaten und die Anzeige ihrer Metadaten kann über das Oracle Java Tool *GeoRasterViewer* erfolgen. Für die hybride Visualisierung von Vektor- und Rasterobjekten ist mit *MapViewer* ein weiteres Simple Viewing Tool verfügbar, das neben der reinen Darstellung die Generierung einfacher Symbolik für Vektorlayer erlaubt. Die Fülle an durch GIS und multimedialer Technologie verfügbaren Klassifizierungmethoden und kartographischen Darstellungen (vgl. [MENG, 2001]) kann durch diese bislang auf dem Markt befindlichen Werkzeuge selbstredend nicht genutzt werden.

Bislang bestehen in erster Linie noch Defizite bei der Komprimierung der Rasterdaten direkt über das ORDBMS. Hier zeigt sich der enorme Know-How-Vorsprung der GIS-Hersteller, nicht zuletzt aufgrund der langjährigen Kooperation mit den Anbietern der führenden Kompressions-Technologien LZW (Unisys) und MrSID (LizardTech). Jedoch entwickeln auch hier Drittanbieter (z.B. Keagan Systems) auf Basis der von Oracle definierten Datenbankstrukturen geeignete Werkzeuge, die mit der Bereitstellung von Funktionalität für Raster-Vektor-Konvertierung einen weiteren bedeutenden Mosaikstein zur Datenbank-basierten GRID-Analyse liefern.
[FRANCICA, 2004]

5.1.3 Migration von Sachdaten

Für den Fall, dass Geoobjekte wenig komplex sind und sich deren Eigenschaften klassisch über eine Sequenz atomarer Attribute ausreichend genau beschreiben lassen, ist die nachträgliche Abbildung relational modellierter Sachdaten auf objektrelationale Datenbankstrukturen unzweckmäßig. Sie macht nur dann Sinn wenn:

- zu Gunsten der objektorientierten Anwendungsentwicklung mit Objekten anstelle von Tabellen-Tupeln gearbeitet werden soll (Vermeidung des Impedance Mismatch, vgl. Abschnitt 3.3).
- der intuitiveren Darstellung der Daten Vorrang gegenüber der geringeren Komplexität des Datenzugriffs gewährt wird.

Für die Zusammenführung des Informationsgehalts mehrerer relationaler Datenbanken für eine konkrete Anwendung sind Objekt-generierende Objektsichten (vgl. Abschnitt 3.4.5) als natürliche Brücke zwischen dem relationalen und objektrelationalen Paradigma das geeignete Werkzeug. Sie bieten die selbe Funktionalität wie Objekttabellen, erhalten jedoch die flachen, zugrunde liegenden Relationenmodelle für die einfache Fortführung der Datenbasen. Der direkte Zugriff auf die Sachdatentabellen über die GIS-Software bleibt deshalb hier unproblematisch (vgl. Abschnitt 5.2).
Die Tupel aller relationalen Tabellen sind demnach als Zeilenobjekte zugänglich, sofern für diese Tabellen Objektsichten angelegt wurden, wie im Folgenden für die Synchronisation von jährlichen Klimawerten einer Messstation des Nationalparks Bayerischer Wald realisiert:

```
create or replace type WALDHAEUSER_TY as object (
       jahr            NUMBER (4),
       monat           NUMBER (2),
       tag             NUMBER (2),
       t_max           NUMBER (3,1),
       t_min           NUMBER (3,1),
       t_kw            NUMBER (3,1),
       schneebedeckung NUMBER (3)
       );


create or replace view WALDHAEUSER_2000_OV
       of WALDHAEUSER_TY
       with object identifier (jahr, monat, tag) as
       select yahr, monat, tag, t_max, t_min, t_kw, schneebedeckung
       from WALDHAEUSER;
```

Fremdschlüsselbeziehungen zu möglichen Detailtabellen des Relationenmodells können bei Bedarf über weitere Objektsichten mit Referenzen simuliert werden.
Die Aggregation der während des gesamten Analysezeitraums erhobenen Klimadaten aller Stationen in einer Haupttabelle *MAIN* sowie weiterer anwendungsrelevanter Daten, die in beliebigen relationalen Datenbanken gespeichert sein können, erfolgt dann zweckmäßig mittels übergeordneter Objekttypen mit Referenzen, Tabellentypen und Nested Tables.
Die vollständige Zusammenführung von für die Analyse der Totholzausbreitung relevanten Fachdaten ist in [SCHEUGENPFLUG et al., 2004] beschrieben.

```
create or replace type KLIMA_TY as object (
       waldhaeuser_pro_jahr        REF        WALDHAEUSER_TY,
       lusen_pro_jahr              REF        LUSEN_TY,
       taferlruck_pro_jahr         REF        TAFERLRUCK_TY);


create or type KLIMA_WERTE_NT
as table of KLIMA_TY;


create table MAIN (
       jahr        NUMBER(4),
       geometrie   MDSYS.SDO_GEOMETRY,
       klima       KLIMA_WERTE_NT,
       bestand     BESTAND_WERTE_NT,
       boden       BODEN_WERTE_NT,
       info        XMLTYPE)
nested table klima store as KLIMA_WERTE_NT_TAB,
nested table bestand store as BESTAND_WERTE_NT_TAB,
nested table boden store as BODEN_WERTE_NT_TAB;
```

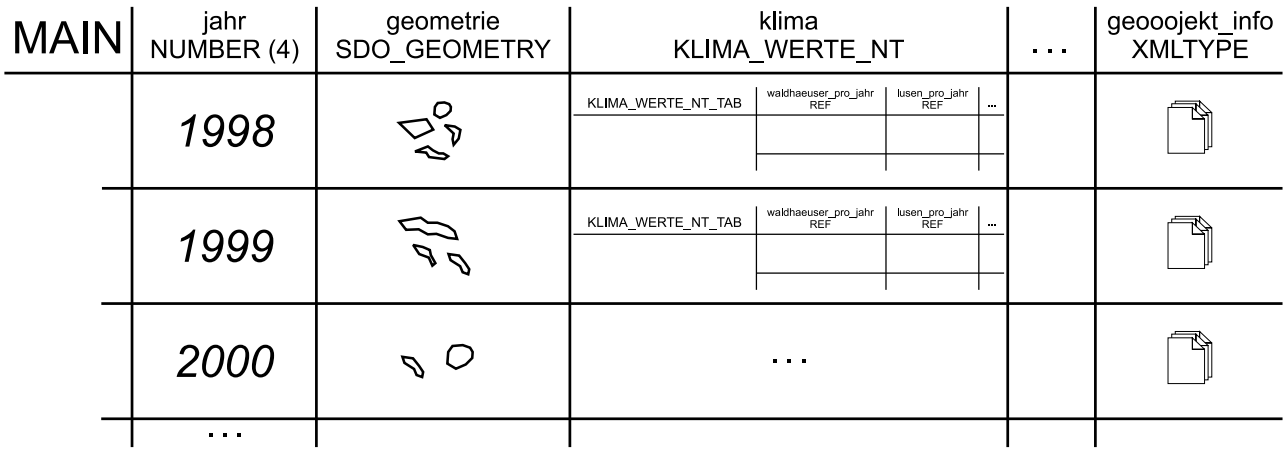

Abb. 5.6: Aggregation von Analyse-relevanter Sachinformation aus verschiedenen Relationenmodellen

5.1.4 Migration von Metadaten

Nach der Etablierung von XML als de-facto Standard für das elektronische Publizieren und den flexiblen Austausch von (komplex strukturierten) Daten, ist der Bedarf XML-Daten in Datenbanken zu speichern beträchtlich gewachsen. Während reine XML-Datenbanken auf XML beschränkt bleiben bieten native XML-Speicherformate (Datentypen), die auf objektrelationaler Technologie von Standarddatenbanken aufsetzen, den Vorteil, dass XML und relationale Daten einfach in einem Datenbankserver integriert werden können.

Aus Sicht der Geoinformatik besteht zunächst die Anforderung, Metadaten nicht nur bzgl. der geforderten Inhalte standardisiert abzulegen (etwa ISO 19115:2003) sondern auch den offenen, standardisierten Zugriff darauf zu gewähren.
Die von Seiten des GIS-Herstellers ESRI längst implementierte XML-basierte Speicherung von Datensatz-begleitender Metainformation, die den Aufbau von Metadaten-Services und damit die Recherche von verfügbaren anwendungsrelevanten Geodaten unmittelbar unterstützt, stellt analog zur integrierten relationalen Abbildung von Geodaten einen klaren proprietären Ansatz dar, da die Metainformation als BLOB-Daten in Systemtabellen von ArcSDE geschrieben werden. Das vollständige Auslesen des Binärstroms und seine Darstellung als XML-Dokument ist via SQL und entsprechenden Methoden zwar möglich (vgl. Realisierung in Abschnitt 4.2.4.2), der gezielte Zugriff auf Elemente in XML-Dokumenten scheitert allerdings.
Dagegen bietet nach [ORACLE (1), 2003] die Speicherung von Metadaten als datenzentrische XML-Dokumente auf Basis der entsprechenden ORDBS-Basisdatentypen (XMLTYPE bei Oracle) die Unterstützung der W3C-Standards

- XPath zur Suche in XML-Dokumenten
- Stylesheet-Transformationen mit XSLT innerhalb des ORDBS
- XML Schema zur Beschreibung der XML-Strukturen
- Validierung der XML-Dokumente gegen ein XML Schema

wodurch die systemunabhängige Verarbeitung und Nutzung der Metadaten gewährleistet ist.

XML wird hier nicht bloß als Abfolge von Zeichen betrachtet, sondern gleichzeitig als geordnete Menge von Objekten, für die über die DOM-Schnittstelle abstrakte Objektklassen bereitgestellt werden, welchen i.d.R. ein bestimmter Knotentyp in der Baumrepräsentation des Objektmodells entspricht. Durch die bekannte objektrelationale Struktur lässt sich der Zugriff auf XML-Dokumente optimieren, indem die Anfragen vom ORDBMS in objektrelationale SQL-Syntax umgeschrieben wird (Query Rewrite) [ORACLE (1), 2003].

Ein im Kontext von Metadaten in der Geoinformatik noch immer akutes Problem ist die Wahrung der Konsistenz zwischen Datensatz-begleitenden Metadaten und zentralen Metadaten infolge redundanter Datenhaltung.
Durch die Integration von XML mit relationaler Datenbanktechnologie, insbesondere der Möglichkeit zur Definition von Constraints und Trigger für XML-Dokumente sowie relationaler Sichten aus XML-Dokumenten sind in ORDBS geeignete Werkzeuge vorhanden, um diese Lücke zu schließen.

Metainformationen zu Geodaten können weiterhin Datensatz-spezifisch und standardisiert abgelegt werden, so wie dies aus Anwendersicht wünschenswert ist. Die skalaren Werte des XML-Metadaten-Dokuments eines Geodatensatzes werden hier in Objekttabellen auf Basis entsprechender Objekttypen gespeichert, deren Struktur aus einem XML-Schema-Dokument abgeleitet wird, das etwa zum Metadatenstandard nach ISO 19115 konform ist.
Die für die gezielte Navigation nach anwendungsrelevanten Geodaten erforderliche zentrale, (relationale) Repräsentation der Metainformation kann dann im Bedarfsfall mittels relationaler Sichten aus dem XML-Content generiert werden.

5.2 Zugriff auf objektrelationale Strukturen

Die Probleme beim Zugriff auf objektrelational modellierte Sachdaten in GIS sind bekannt, aber noch nicht gelöst. So priorisiert [HARTMANN, 2002] im Zuge der Evaluierung und prototypischen Entwicklung zur Modellierung der Liegenschaftsdaten (ALKIS) den Ansatz, nur für die Speicherung der Geometriedaten auf OR-Konzepte aufzusetzen, die Sachdaten jedoch relational zu modellieren.
Da verschachtelte Tabellen, Objekttabellen und nichtatomare Datentypen mit Ausnahme von SDO_GEOMETRY bislang nicht von den GIS-Produkten interpretiert werden können, sind für objektrelational modellierte Sachdaten zunächst aufwendige Umformungen auf flache Strukturen, etwa durch dynamische Sichten[1] erforderlich. Nach der Registrierung dieser Sichten in den Metadaten eines GIS-Herstellers kann lesend darauf zugegriffen werden. Der indirekte Zugriff über Sichten ist allerdings umständlich und mit Einbußen bei der Zugriffsgeschwindigkeit verbunden.

Mittels INSTEAD OF Trigger lassen sich die einer Objekt-Sicht zugeordneten Basistabellen in Oracle auch fortführen [ORACLE (2), 2003].
Jedoch können auch Objekt-Sichten nicht über den gewaltigen Bruch zwischen GIS-Software und ORDBMS hinwegtäuschen, der aufgrund unzureichend implementierter („höherer“) objektrelationaler Datenbankschnittstellen (vgl. Abschnitt 3.1) nachwievor existiert. Über Referenzen modellierte Beziehungen müssen in Fremdschlüsselbeziehungen transformiert und Kollektionstypen auf mehrere Relationen aufgeteilt werden (vgl. Abb. 5.7). Der Vorteil der vollständigen Abbildung semantischer Entwurfsinformation auf objektrelationale Datenbankstrukturen - insbesondere bei Spezialisierungshierarchien (Vererbung) - geht somit entweder ganz verloren oder muss durch mehrere in Beziehung stehende Relationen simuliert werden, was in teueren Verbundoperationen und damit verminderter Performanz resultiert.

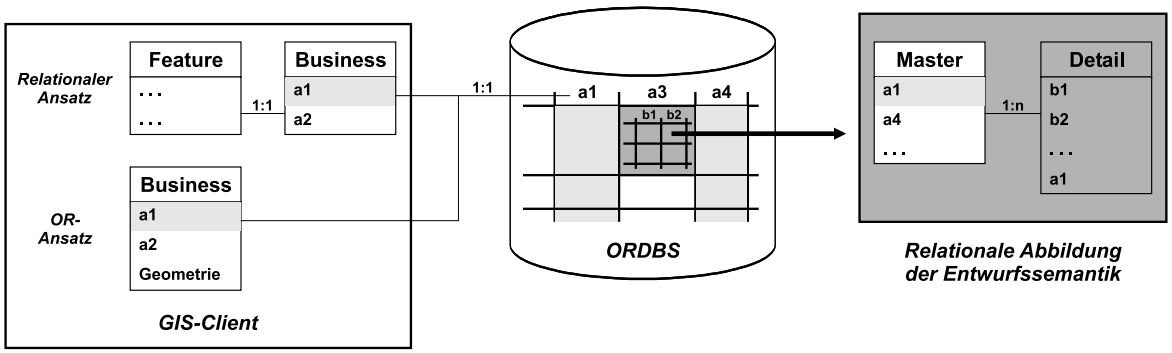

Abb. 5.7: Defizit bei GIS-basierten Zugriffen auf OR-Strukturen am Beispiel kollektionswertiger Tabellen

Zur größtmöglichen Flexibilität bei der Objektmodellierung in GIS bedarf es neben der direkten Unterstützung der Basisdatentypen zur Verwaltung von Raster- und XML-Daten folglich:

- der Erweiterung der bestehenden Datenbankschnittstellen um objektrelationales SQL insbesondere um DQL- und DML-Konstrukte, die auf den objektrelationalen Datenmodellerweiterungen operieren (Interpretation und Fortführung von typisierten bzw. geschachtelten Tabellen)

- GIS-Funktionalität zur Darstellung von 1:m-Beziehungen zwischen Master- und Detail-Objektklassen (vgl. Abb. 5.8)

[1] Eine Sicht ist ein Datenbankobjekt, das das Anlegen angepasster Teile einer Tabelle oder Tabellensammlung ermöglicht. Im Gegensatz zu einer Tabelle enthält eine Sicht keine Daten, sondern nur eine SQL-Anweisung. Die Daten werden über diese Abfrage abgeholt und wie eine Tabelle dargestellt. Wie bei einer Tabelle können bei Sichten mit insert, update und select verschiedene Datenoperationen ausgeführt werden [ABBEY et al., 2000].

Bestand						
Bestandskey	Laufzeitbeginn_FE	Bestandsnummer	...	Bestandsteilflaechen		
				Bstfl_nr	Nutzungsart	Flaechenanteil
5410101018	1996	18	...	1	unbestockt	25%
				2	Altdurchforstung	60%
				3	unbestockt	15%

Masterattribute (Basisdatentypen) Detailattribute (ADT)

Abb. 5.8: Darstellung und Handling von kollektionswertiger Attributinformation in GIS

5.3 Ausgewählte OR-Datenbankanwendungen für den Nationalpark Bayerischer Wald

5.3.1 Datenbankbasierte Analyse der Totholzausbreitung

Kaum ein Thema hat in den letzten Jahren im Nationalpark Bayerischer Wald die Gemüter so bewegt wie die Massenvermehrung des Borkenkäfers und das damit einhergehende, großflächige Absterben der Waldflächen. Bis heute sind seit Beginn der Totholzausbreitung im Jahr 1993 im Rachel-Lusen-Gebiet des Nationalparks ca. 4000 ha Wald abgestorben. Windwürfe gelten zwar allgemein als Auslöser für die Massenvermehrung (Gradation), über die genauen Ursachen der räumlichen Entwicklung der Befallsflächen in Abhängigkeit der Zeit herrscht bislang aber weitgehend Unklarheit. Bislang gelten eine Reihe von Faktoren für die Entwicklung des Buchdruckers, der im Nationalpark heimischen Borkenkäferart, als entscheidend. Neben den klimatischen Verhältnissen in der Schwärmphase des Frühjahrs sind dies Standortfaktoren wie die Bestandesstruktur des Waldes, die Art der Bodentypen und die Durchfeuchtung des Bodens sowie weitere standortspezifischen Parameter, die zur Schwächung der Bäume und damit zu deren Resistenzverlust beitragen. Infolge der vorherrschenden Massenvermehrung kommt der Eigendynamik des Borkenkäfers eine besondere Bedeutung zu, d.h. die Distanz von potentiellen neuen Befallsherden zu noch aktiven Käfernestern ist gegenüber den übrigen Faktoren stärker zu gewichten.

Die Totholzausbreitung stellt ein mehrdimensionales Phänomen dar. Maßgebend hierfür ist die starke Korrelation der Meereshöhe mit den kritischen Einflussfaktoren der Baumartenverteilung und der Temperaturverhältnisse im Nationalpark, ebenso wie die Bedeutung der Entwicklung der Waldschädigung in Abhängigkeit der Zeit.

Die Entwicklung der Käferpopulation sowie die Ausbreitung der Totholzflächen ist als natürlicher Prozess eines Waldökosystems ungemein komplex, nicht zuletzt, da die genannten Faktoren nicht nur singulär einwirken sondern ein Korrelationsgeflecht bilden (vgl. Abb. 5.9). Noch schwieriger scheint die Gewichtung der weitgehend bekannten Faktoren. In Hinblick auf eine vollständige Analyse der Ausbreitungsdynamik ist weiterhin problematisch, dass über entscheidende Einflussgrößen entweder gar keine Messdaten vorliegen oder diese mit unzureichender räumlichen und temporalen Auflösung verfügbar sind.
Positiv ist anzumerken, dass sich das Ausbreitungsszenario mit all seinen einwirkenden Einflussfaktoren durch Geometrie- und Sachdaten vollständig beschreiben lässt, was für den Einsatz von GIS bzw. räumlicher Operatoren und Funktionen eines ORDBMS als Analyseansatz spricht. Für den OR-Ansatz spricht neben den Vorteilen bei der statistischen Auswertung (Kopplung mit Data Mining Konzepten) unter anderem die hohe Flexibilität bei der Verkettung von Funktionen im Bereich des Geoprocessings, was insbesondere bei Prozessen mit Multikorrelation von Bedeutung ist.
Eine detaillierte Beschreibung der Borkenkäferproblematik im Nationalpark Bayerischer Wald und seiner bestimmenden Einflussgrößen gibt [NPV, 2001].

5.3.1.1 Analyseansatz

Ein möglicher Ansatz zur Analyse und Bewertung der Gewichtung einzelner, die Totholzentwicklung beeinflussender Faktoren ist in [SCHEUGENPFLUG, 1999] beschrieben. Das Verfahren basiert auf einer sukzessiven Verschneidung der Einzelfaktoren mit den Totholzflächen der einzelnen Epochen der Zeitreihe. Anschließend erfolgt die Bewertung der Gewichtung der Einflussgrößen auf Grundlage der Anzahl bzw. der Flächensumme aller

Totholzpolygone einer Epoche pro Faktorenklasse. Es wird dabei angenommen, dass der Einfluss eines Faktors direkt proportional zur Anzahl bzw. der Gesamtfläche der Totholzflächen in der entsprechenden Klasse ist. Die einheitliche Bewertung hinsichtlich des Einflusses einzelner Faktoren erfordert zudem die Normalisierung ihrer Klassen (Division durch deren Gesamtfläche).
Um der Eigendynamik der Borkenkäfermassenvermehrung Rechnung zu tragen, erfolgt die gesamte Analyse und Auswertung lediglich mit Totholzflächen, die sich ausserhalb einer 200 Meter-Distanz zu den Käfernestern (Totholzflächen) des Vorjahrs befinden, da davon auszugehen ist, dass in unmittelbarer Nähe zu noch aktiven Käfernestern die Art der Einflussfaktoren unerheblich ist.

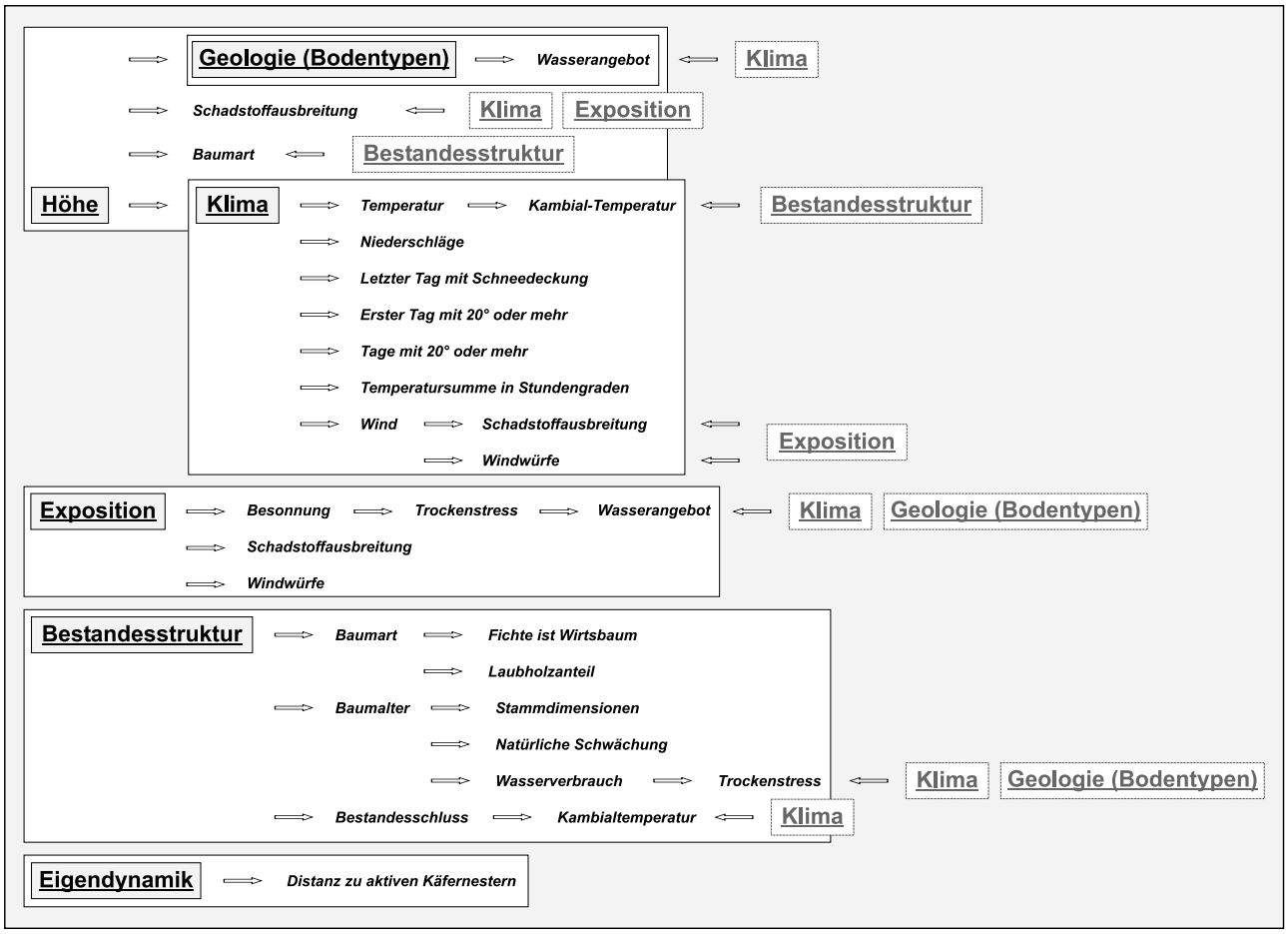

Abb. 5.9: Multikorrelation der Einflussgrößen der Borkenkäferentwicklung / -ausbreitung

Durch den SQL-basierten Analyseansatz im allgemeinen und dem Einsatz (temporärer) Spatial Views im besonderen ist die Verkettung einzelner Geoprocessing-Funktionen zu einem Gesamtworkflow möglich, der für jeden Epochenübergang genutzt werden kann (vgl. Abb. 5.10).

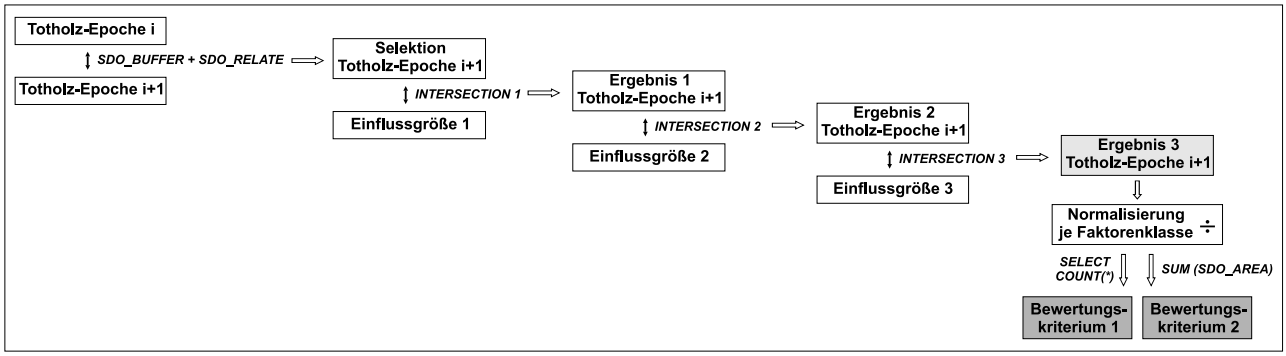

Abb. 5.10: ORDBMS-gestütztes Analyseverfahren zur Gewichtung der Einflussfaktoren der Totholzausbreitung

5.3.1.2 Realisierung

Die Auswertung erfolgt für die Epochenübergänge von 1994 bis 1999, da der Totholzzuwachs in diesem Zeitraum massiv war. Als Einflussgrößen werden betrachtet:

Einflussgröße	Faktorenklassen
Exposition	Nord, Nord-Ost, Ost, Süd-Ost, Süd, Süd-West, West, Nord-West
Bodenarten / Feuchtigkeitsklassen	feucht, frisch, mäßig frisch, Moor, Mosaik
Bestandesstruktur	Bestandsform / Baumartenverteilung, Altersklasse

Tab. 5.1: Einflussgrößen und Faktorenklassen des Analyseprozesses

Die Abhängigkeit der Flächenausbreitung von der Höhenlage lässt sich auf die starke Korrelation der Höhe mit der Baumartenverteilung (98% Fichtenanteil in den Hochlagen, Fichte ist Wirtsbaum des Borkenkäfers) zurückführen. Sie ist dadurch im Analyseansatz bereits berücksichtigt und wird daher nicht als eigenständige Einflussgröße eingeführt. Für die Auswertung besonders interessant können ggf. auch bestimmte Kombinationen aus den in Tab. 5.1 aufgeführten Faktorenklassen sein, wie etwa die Betrachtung der Waldbestände auf (trockenen) Mosaikböden, die zusätzlich nach Süden exponiert sind, und somit einer ganztägigen Sonneneinstrahlung ausgesetzt sind. Dadurch ergibt sich ein doppelt erhöhtes Risiko, dass die Bäume Trockenstress erleiden und somit anfälliger gegenüber Borkenkäferbefall sind.

Die Geoinformation der Einflussgrößen liegt in Form von Vektor-Geometrien (Single-Part-Features) und direkten Attributen vor. Die Beschreibung der Totholzflächen der Epochen durch Single-Part-Features ist zur Vorselektion relevanter Polygone auf Basis eines *Spatial Join* erforderlich, der sämtliche Geometrien eines Layers mit allen Geometrien eines zweiten Layers vergleicht.

Zunächst werden die relevanten Vektordaten mittels des Tools „SHP2SDO“ in Oracle Spatial Tabellen konvertiert und mit einem Primärschlüsselfeld sowie der korrekten Kartenprojektion versehen. Anschließend folgt der Eintrag der entsprechenden Metadaten in das Oracle Spatial Schema.

```
shp2sdo -o totholz1999 totholz1999 -i -s 82032
```

```
drop table TOTHOLZ1999;
createtable TOTHOLZ1999 (
      id            NUMBER(38) not null primary key,
      jahr          NUMBER,
      area          NUMBER,
      perimeter     NUMBER,
```

```
      geom          MDSYS.SDO_GEOMETRY);
delete from USER_SDO_GEOM_METADATA
where table_name = 'TOTHOLZ1999' and column_name = 'GEOM';
insert into USER_SDO_GEOM_METADATA (table_name, column_name, diminfo, srid)
      values ('TOTHOLZ1999', 'GEOM',
      MDSYS.SDO_DIM_ARRAY
      (MDSYS.SDO_DIM_ELEMENT('X', 4598321.000000000, 4615963.500000000, 0.000000050),
      MDSYS.SDO_DIM_ELEMENT('Y', 5416977.500000000, 5429338.122558350, 0.000000050)
      ),
      82032);
commit;
```

Nach der Befüllung der Oracle Spatial Tabellen (Stützpunktkoordinaten der Geometriedaten) mittels „SQL*Loader“

```
sqlldr totholz/***@npora9 TOTHOLZ1999
```

ist zunächst ihre Migration auf die Datenbankversion 9i erforderlich

```
execute sdo_migrate.to_current('TOTHOLZ1999', 'GEOM');
```

bevor die Indizierung der objektrelationalen Geometrien vorgenommen wird. Zur Nutzung des gesamten Funktionsumfanges der räumlichen Operatoren und Funktionen empfiehlt Oracle die Indizierung mit einem R-Tree:

```
create index TOTHOLZ.INDEX_TOTHOLZ1999 on TOTHOLZ.TOTHOLZ1999 (GEOM)
indextype is mdsys.spatial_index parameters ('sdo_indx_dims=2 layer_gtype=collection
sdo_rtr_pctfree=0');
```

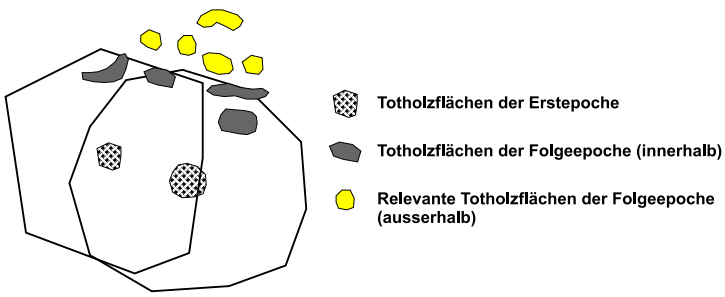

Abb. 5.11: Räumliche Selektion von Analyse-relevanten Totholzflächen

Um der Eigendynamik der Massenvermehrung Rechnung zu tragen, beginnt der Analyseprozess mit der Selektion von Totholzpolygonen, die weiter als 200 Meter zu den Totholzflächen des Vorjahres entfernt liegen (vgl. Abb. 5.11). Die entstehenden Geometrien der 200 m-Buffer werden in eine erste (temporäre) Spatial View übertragen:

```
create view buffer_ergebnis1999 (GEOM) as select SDO_GEOM.SDO_BUFFER
(c.geom, m.diminfo, 200) from totholz1998 c, user_sdo_geom_metadata m
where m.table_name='TOTHOLZ1998' and m.column_name='GEOM';
```

Anschließend wird geprüft, welche Polygone der Folgeepoche nicht innerhalb der Buffer liegen. Die Selektionsmenge wird nach Eliminierung der Duplikate von einer zweiten Spatial View aufgenommen:

```
create view endergebnis_selektion1999 (GEOM, id) as select a.geom, a.id
from totholz1999 a where a.id not in (select distinct a.id from totholz1999 a,
BUFFER_ERGEBNIS1999 b where SDO_RELATE (a.GEOM, b.GEOM, 'mask=ANYINTERACT
querytype=JOIN') = 'TRUE');
```

Im Folgenden werden die Einflussgrößen (vgl. Tab. 5.1) mit den vorselektierten Totholzflächen sequenziell verschnitten, wobei aus den entstehenden Tupel explizit jene selektiert werden müssen, die eine Schnittgeometrie enthalten. Die vollständige Syntax des gesamten Analyseprozesses für einen Epochenübergang ist in Anhang E dargestellt.

```
create view ergebnis_feucht_intersect1999 (id_selek1, id_feucht, text_feucht, GEOM)
as select s.id, d.id, d.text, SDO_GEOM.SDO_INTERSECTION (d.GEOM, m.DIMINFO, s.GEOM,
m.DIMINFO) from FEUCHTIGKEIT d, ENDERGEBNIS_SELEKTION1999 s, user_sdo_geom_metadata m
where m.table_name='FEUCHTIGKEIT' and m.column_name='GEOM';

create view ende_feucht_intersect1999 (id_selek1, id_feucht, text_feucht, GEOM)
as select p.id_selek1, p.id_feucht, p.text_feucht, p.GEOM
from ergebnis_feucht_intersect1999 p where p.GEOM is not null;
```

Das Endresultat sämtlicher Verschneidungen wird aus der letzten View in einen Spatial Table transferiert, auf dessen Basis die Auswertung erfolgt:

```
create table ende1999 as select * from ende_best_intersect1999;

delete from USER_SDO_GEOM_METADATA
where table_name = 'ENDE1999' and column_name = 'GEOM';

insert into USER_SDO_GEOM_METADATA (table_name, column_name, diminfo, srid)
       values ('ENDE1999', 'GEOM',
       MDSYS.SDO_DIM_ARRAY
       (MDSYS.SDO_DIM_ELEMENT('X', 4598321.000000000, 4615963.500000000, 0.000000050),
       MDSYS.SDO_DIM_ELEMENT('Y', 5416977.500000000, 5429338.122558350, 0.000000050)
       ),
       82032);
```

Die Visualisierung mittels GIS erfordert die Registrierung als Spatial Layer im Datenmodell des GIS-Herstellers:

```
sdelayer -o register -l ENDE1999,GEOM -e a -c ID -k DEFAULTS -u totholz -p *** -i esri_sde
-s gis14 -C SDE
```

Vor der Auswertung empfiehlt sich die Indizierung der Spalten der zu analysierenden Faktorenklassen, wobei sich für Kombinationen konkatenierende Indizes anbieten:

```
create index index_attribut_ende1999 on ende1999 (TEXT_FEUCHT, TEXT_EXPO);
```

Die Auswertung der Tabelle ist trivial. Die Anzahl bzw. Gesamtfläche der Totholzpolygone einer Faktorenklasse lassen sich einfach ermitteln durch:

```
select count(*) from ende1999 where text_feucht = 'mosaik' and text_expo = 'Sued';

select sum(sdo_geom.sdo_area(GEOM, 0.005, 'unit=SQ_M')) from ende1999 where text_feucht
= 'mosaik' and text_expo = 'Sued';
```

5.3.1.3 Ergebnisse

Tab. 5.2 zeigt die Ergebnisse der Auswertung für ausgewählte (Kombinationen von) Faktorenklassen für die Zeitreihe zwischen den Jahren 1995 und 1999. Dabei ist

G: die zur Normalisierung benötigte Gesamtfläche einer Faktorenklasse in [m^2]

N: die Anzahl der Totholzpolygone in einer Faktorenklasse

F: die Gesamtfläche dieser Totholzpolygone in [m^2]

$P = \frac{F}{G}$: der Wahrscheinlichkeitsfaktor für die besondere Bedeutung einer Einflussgröße

	1995	**1996**	**1997**	**1998**	**1999**
Mosaik G = 11.653.682	N = 139 F = 43.452 P = 0,0037	N = 278 F = 199.484 P = 0,0171	N = 497 F = 733.773 P = 0,0630	N = 359 F = 336.957 P = 0,0289	N = 738 F = 923.822 P = 0,0793
Moor G = 9.303.441	N = 59 F = 17.114 P = 0,0018	N = 277 F = 104.601 P = 0,0124	N = 513 F = 310.086 P = 0,0333	N = 465 F = 305.280 P = 0,0328	N = 504 F = 391.707 P = 0,0421
Exposition Nord G = 6.268.240	N = 100 F = 39.787 P = 0,0063	N = 251 F = 270.277 P = 0,0431	N = 371 F = 524.319 P = 0,0836	N = 294 F = 291.991 P = 0,0466	N = 453 F = 526.762 P = 0,0840
Exposition Süd G = 24.636.272	N = 150 F = 65.114 P = 0,0026	N = 376 F = 403.746 P = 0,0164	N = 759 F = 1.059.017 P = 0,0430	N = 574 F = 466.294 P = 0,0189	N = 945 F = 916.381 P = 0,0372
Bestandsalter 30-150 Jahre G = 38.533.491	N = 538 F = 190.849 P = 0,0495	N = 1.086 F = 729.150 P = 0,0189	N = 2.223 F = 2.486.589 P = 0,0645	N = 2.094 F = 1.794.930 P = 0,0466	N = 3.387 F = 3.440.417 P = 0,0893
Bestandsalter 150-330 Jahre G = 14.728.702	N = 505 F = 234.959 P = 0,0160	N = 1.497 F = 2.101.804 P = 0,1427	N = 1.727 F = 3.285.622 P = 0,2231	N = 1.075 F = 948.917 P = 0,0644	N = 1.663 F = 1.809.486 P = 0,1226
Moor + Exposition Nord G = 595.678	N = 8 F = 1.873 P = 0,0031	N = 16 F = 4.001 P = 0,0067	N = 56 F = 20.733 P = 0,0348	N = 47 F = 43.404 P = 0,0729	N = 53 F = 123.360 P = 0,2071
Mosaik + Exposition Süd G = 2.422.547	N = 15 F = 4.138 P = 0,0017	N = 54 F = 35.452 P = 0,0146	N = 113 F = 173.554 P = 0,0716	N = 78 F = 93.154 P = 0,0385	N = 131 F = 115.278 P = 0,0476

Tab. 5.2: Bewertung von Einflussgrößen für die Totholzausbreitung von 1995 bis 1999

Aus der Aufstellung ist ersichtlich, dass die Auswertung des Totholzaufkommens in Relation zum Bestandsalter bis auf kleinere Einschränkungen in den Jahren 1998 und 1999 sowohl bei der Anzahl N der gezählten Totholzpolygone als auch beim Wahrscheinlichkeitsfaktor P die höchsten Werte verzeichnet. Der Einfluss der vergleichsweise großen Gesamtflächen dieser beiden Faktorenklassen (G) ist für P durch die durchgeführte Normalisierung bereits eliminiert. Auch nach einer Hochrechnung von N der anderen Faktorenklassen auf die gleiche Gesamtfläche G weisen die beiden auf Basis des Bestandsalters definierten Faktorenklassen im Durchschnitt die größten N-Werte auf.

Ob sich hieraus ein erhöhter Einfluss der Altersstruktur der Bestände im Allgemeinen ableiten lässt, obliegt der Bewertung durch forstwissenschaftliche Institutionen. Im Rahmen dieser Arbeit liegt der Fokus auf der Erarbeitung und Evaluierung eines Analyseverfahrens, das vollständig auf objektrelationalen Datenbankstrukturen aufsetzt.

Die flexible Verkettung von Geoprocessing-Funktionen ist zuletzt verstärkt von den führenden GIS-Herstellern in ihren Produkten implementiert worden. Einzelne Prozessschritte der Geoverarbeitung werden zu einem Funktionsmodell zusammengefasst und können so mit leicht veränderten Parametern mehrfach aufgerufen werden [ESRI (3), 2005].

5.3.2 Real Time Tracking von Wildtieren

Auf der Gesamtfläche des Nationalpark Bayerischer Wald von ca. 24.000 ha leben rund 250 - 300 Stück Rotwild. In den letzten Jahren haben sich die Lebensbedingungen für das Rotwild etwa aufgrund des Abbaus des Grenzzauns zum benachbarten tschechischen Nationalpark Šumava und den dort stattfindenden großen Kahlschlägen stark verändert (vgl. [NPV, 2003]).

Durch den Einsatz von GPS-GSM-Halsbändern in Kombination mit objektrelationalen Erweiterungen zur nativen Verwaltung und Analyse von Vektordaten in ORDBMS sowie hochpräziser Datentypen zur Speicherung von Datums- und Zeitwerten lässt sich die Wanderung der Tiere in Echtzeit verfolgen. Für den Fall, dass ein Tier eine vorher definierte Zone verlässt, reagiert das Monitoring-System durch automatische Benachrichtigung des Personals.

```
create table BEOBACHTUNGSGEBIET (
       gebiets_id    NUMBER (2),
       gebiets_name  VARCHAR2 (50),
       grenze        MDSYS.SDO_GEOMETRY);
```

Zur Verfolgung der Wanderung der Tiere können ihre Positionen in regelmäßigen zeitlichen Abständen in Echtzeit in eine weitere Tabelle eingefügt werden:

```
create table WILD_POSITION (
       tier_id    NUMBER (3),
       zeit       TIMESTAMP (2),
       position   MDSYS.SDO_GEOMETRY);
```

Der SQL-Datentyp *TIMESTAMP* erlaubt die Speicherung von Datums- und Zeitwerten mit Genauigkeiten im Millisekundenbereich [ORACLE (1), 2004]. Beim Update der Position des GPS-Receivers wird der aktuelle Zeitstempel und die Position automatisch gesendet und in die Positions-Tabelle geschrieben. Die Möglichkeit, etliche Datensätze mit präzisen Positions- und Zeitstempel-Werten für beliebig viele Tiere automatisch einzufügen, erlaubt die Entwicklung eines Real-Time-Tracking-Systems für zahlreiche Tierarten im Nationalpark.

```
insert into WILD_POSITION values (3, to_timestamp ('17-JUN-2004 19:17:21.00',
'dd-mon-yyyy hh24:mi:ss.ff'), mdsys.sdo_geometry (2001, 82032,
mdsys.sdo_point_type (4607260.47, 5425484.18, null), null, null));
```

Durch die Kombination von Spatial Functions wie der *Relate Function* mit einem AFTER INSERT Trigger, der nach jedem Insert einer Tierposition ausgeführt wird, lässt sich überprüfen, ob sich ein bestimmtes Tier z.B. noch innerhalb des Nationalparkgebiets befindet. Die Abfrage für den Trigger Body ist demnach:

```
select sdo_geom.relate (p.position, 'ANYINTERACT', b.grenze, 0.5)
from wild_position p, beobachtungsgebiet b
where p.tier_id = 3 and b.gebiets_name like 'nationalpark' and
p.zeit = (select max (zeit) from wild_position where tier_id = 3);
```

Die Unterabfrage liefert den aktuellsten Zeitstempel für das Tier mit der ID = 3. Für den Fall, dass sich das Tier noch innerhalb des Parks befindet, wird *TRUE* zurück gegeben. Hat das Tier den Park verlassen und ist etwa

nach Šumava gewandert, wird *FALSE* zurückgegeben und der Trigger kann z.B. eine HTTP-Verbindung öffnen und eine Nachricht mit der Tier-ID, der Position und dem exakten Zeitpunkt des Überschreitens der Grenze an einen Web-Service ausserhalb der Datenbank senden. Der Web-Service kann dann - falls gewünscht - Emails an die Nationalparkverwaltung schicken oder dynamische Karten auf Basis der aktuellen Tierpositionen erstellen.

Falls sich herausgestellt hat, dass ein Wildtier ein bestimmtes Gebiet einmal pro Jahr durchwandert, können benutzerdefinierte Objekttypen erzeut werden, die genau dieses Ereignis als Raum-Zeit-Zone beschreiben:

```
create or replace type RZ_Zone_TY as object (
       name    VARCHAR2 (50),
       von     DATE,
       bis     DATE,
       zone    MDSYS.SDO_GEOMETRY);
```

Der Objekttyp *RZ_Zone_TY* kann eine Methode enthalten, die überprüft, ob der aktuelle Aufenthaltsort eines Tiers mit dem für diese Jahreszeit erwarteten übereinstimmt, d.h. ob sich das Tier noch innerhalb der vordefinierten Raum-Zeit-Zone befindet.

5.4 Fazit

Mit der Unterstützung der auf Objekttypen basierenden Datentypen zur strukturierten Speicherung von Vektordaten in Datenbanktabellen von ORDBS haben die führenden GIS-Hersteller einen Weg beschritten, der das Bemühen zur Integration von ORDBMS in GIS kennzeichnet und richtungsweisend für die weitere Entwicklung sein sollte.

1. **Vergleich RDBMS / ORDBMS**

   Die durch den integrierten, relationalen Datenhaltungsansatz erzielten Fortschritte für eine realitätsnähere Modellierung von Geoobjekten (vgl. Abschnitt 4.5) bleiben vollständig erhalten und werden um den bedeutsamen Aspekt des offenen Zugriffs erweitert. Dies bringt für die Anwendungspraxis wesentliche Vorteile. Anstelle der proprietär geschriebenen BLOB-Felder, die nur systemabhängig durch die entsprechende GIS-Software interpretiert werden können, treten „transparente“ Objekttypen, die parallel auch den direkten Zugriff via SQL (ohne GIS) zulassen. Allerdings herrschen nachwievor ganz spezifische SQL-Dialekte vor, die von den in SQL/MM-Spatial erarbeiteten Vorgaben abweichen und selbst innerhalb eigener Datenbank-Releases mit unter gravierenden Änderungen unterworfen sind.

2. **Erweiterte Analysefunktionalität**

   Die in ORDBS verfügbaren räumlichen Funktionen eignen sich vor allem zur Analyse bzw. Berechnung vollständiger Layer und weniger zur gezielten Fortführung einzelner Geometrie-Segmente. Da die Zugriffe überwiegend lesend erfolgen, sind für GIS-Anwendungen so bedeutende Aspekte wie Multiuser-Editing und lange Transaktionen in den ORDBMS-Produkten nicht zufriedenstellend umgesetzt worden. Nicht zuletzt aufgrund der eingeschränkten Visualisierungsmöglichkeiten und der im Vergleich zur Benutzerfreundlichkeit eines GIS komplexen SQL-Syntax, werden ORDBS im breiten Einsatz auf absehbare Zeit nicht über die alleinige Rolle der Datenhaltungskomponente hinauskommen.

3. **GIS-Anwendungen ohne eigenes GIS**

   Allerdings lassen sich Web-fähige Monitoring-Lösungen und LBS-Dienste, bei denen weniger die Fortführung komplexer Geometrien, sondern die einfache Verortung von Sachinformationen im Vordergrund steht, hervorragend ohne den Einsatz eines GIS realisieren. Zur Unterstützung derartiger Anwendungen hat sich in den vergangenen Jahren ein deutlicher Trend weg von proprietären Lösungen für Geodaten hin zu offenen Systemen entwickelt, die die nahtlose Integration raumbezogener Daten in die IT-Infrastruktur eines Anwenders erlauben (siehe auch [IDC, 1999]). Die Palette an singulär auf ORDBS zu realisierenden GIS-Anwendungen wurde zuletzt durch die Einführung eigener Topologie- und Netzwerkmodelle, die auf neuen geeigneten Objekttypen aufsetzen, zusätzlich erweitert.

4. **Qualitativ höhere Analysen / statistische Auswertungen**

   Die in dieser Arbeit realisierte Datenbank-basierte Analyse der Totholzentwicklung, bei der einzelne Geoprocessing-Funktionen über räumliche Sichten zu einem Gesamtprozess konkateniert wurden (vgl. Abschnitt 5.3.1.2), bietet gegenüber dem GIS-basierten Ansatz den Vorteil, dass statistische Auswertungen direkt in der Datenbank erfolgen können und die speicherintensiven Datenbestände nicht zwischen verschiedenen Applikationen im Netzwerk transferiert werden müssen. Die Funktionalität zur benutzerdefinierten Verkettung einzelner Analyseschritte zu einem Gesamtmodell wurde zuletzt auch in den Produkten der führenden GIS-Hersteller implementiert, wodurch sich komplexe Prozesse klar strukturieren und dokumentieren lassen.

5. **Interoperabilität auf Basis von OGC-Spezifikationen**

   Der interoperable Zugriff mit GIS unterschiedlicher Hersteller auf Vektordaten, die auf Basis des Oracle Spatial Datentyps SDO_GEOMETRY gemäß den Spezifikationen des OGC (Simple Features Specification for SQL) gehalten werden, ist mit Einschränkungen möglich, was Gegenstand diverser Untersuchungen war (vgl. etwa [BLANKENBACH, 2001] und Abschnitt 6.2, 2).

6. **Interoperabilität auf ORDBMS-Ebene**

   Die interoperable Nutzung von Geodaten, die auf der Simple Features Specification aufsetzt, ist für GIS-Anwendungen allerdings nicht befriedigend. Die Approximation von Kreisbögen und Splines durch eine Vielzahl an Liniensegmenten sollte nicht das Ende der Entwicklung darstellen. Aus der idealistischen Sicht, Interoperabilität auf Datenbankebene zu erreichen, besteht ferner Bedarf, zusätzlich die Standardisierung des Handlings von Betextung voranzutreiben. Die standardisierte Ablage einer Position mit Drehwinkel und deren Zuordnung an Tabellenspalten, die für Annotation herangezogen werden, ist vorzugsweise durch geeignete Objekttypen anzustreben. *Feature linked Annotation*, bei der die Betextung als Teil des Geoobjekts betrachtet und die Fortführung eines Features (Positionsänderung bzw. Eliminierung des Objekts) automatisch auf die Beschriftung übertragen wird, sollte nicht länger proprietär bleiben, sondern muss standardisiert werden. Für die optimale Platzierung von Text-Objekten relativ zu den entsprechenden Geoobjekten bietet sich hier die Nutzung der verfügbaren räumlichen Funktionen wie etwa SDO_CENTROID zur Berechnung des Schwerpunkts von Polygonen und Punktclustern an.

7. **Vorteile von ORDBMS bei der Konsistenzerhaltung**

   Der objektrelationale DB-Ansatz fördert infolge der gemeinsamen Verwaltung von direkten Attributen, Geometrien (und Metadaten) in singulären Datenbanktabellen unmittelbar die Konsistenzerhaltung auch bei direkten Zugriffen via SQL. Die Komplexität von Triggerdefinitionen zur Berücksichtigung von direkten Abhängigkeiten zwischen Attributen und Geometrie ist geringer, was die Transparenz des gesamten Datenbankmodells nach aussen deutlich erhöht.

8. **Empfehlungen**

   Das mit der Unterstützung von Objekttypen zur Speicherung von Vektordaten begonnene Bestreben nach direkter Nutzung von objektrelationaler Funktionalität in GIS kann und muss in naher Zukunft konsequent fortgesetzt werden.

   Von Seiten der ORDBMS-Hersteller sollten in Zusammenarbeit mit den Standardisierungsgremien des OGC und der ISO vordefinierte Basisdatentypen für die Aufnahme weiterer Geoobjekt-Komponenten optimiert werden. Insbesondere besteht hier der Bedarf nach Erarbeitung von XML-Schemata zur Verwaltung von Metadaten (ISO 19115) auf Basis eines standardisierten XML-Basisdatentyps.
   Der Funktionsumfang zur Rasterdatenverwaltung auf Basis von Objekttypen ist zu erweitern, insbesondere um Aspekte der Komprimierung, Mosaikierung und Eliminierung einzelner Grauwerte beim Datenimport, um echte Blattschnittfreiheit beim Aufbau flächendeckender Rasterdatenarchive zu gewährleisten.

   Neben der direkten Unterstützung der objektrelationalen Raster- und XML-Basisdatentypen von Seiten der GIS-Hersteller wäre die Erweiterung der bestehenden Datenbankschnittstellen um objektrelationales SQL insbesondere um DQL- und DML-Konstrukte zur Interpretation und Fortführung von typisierten bzw. geschachtelten Tabellen sowie zum Traversieren über Referenzen der nächste zwingend erforderliche

Schritt zur Integration von ORDBMS in GIS. Andernfalls bleiben die durch die hohe Flexibilität des Datenbankentwurfs erzielbaren Effizienzsteigerungen für die Sachdatenmodellierung in GIS nur eingeschränkt nutzbar. Der Zugriff auf OR-Strukturen über flache Sichten kann dagegen lediglich als Behelfslösung zur sanften Migration betrachtet werden.

Kapitel 6

Vergleiche von RDBS- und ORDBS-basiertem Datenhaltungskonzept

In den Kapiteln 4 und 5 wurden die bei der Migration eines auf Grafikformaten basierenden Geodatenbestandes in ein integriert relationales bzw. objektrelationales Datenhaltungskonzept entstehenden Datenbankstrukturen beschrieben. Der dadurch zu erzielende Mehrwert wurde sowohl theoretisch als auch exemplarisch durch ausgewählte Anwendungsbeispiele im Testgebiet Nationalpark Bayerischer Wald demonstriert.
Im Folgenden werden die in Abschnitt 2.2 erarbeiteten konzeptionellen Anforderungen an die Datenhaltung in der Geoinformatik nochmals explizit aus der Sicht der beiden Modellierungs-Paradigmen bzw. der auf ihnen aufsetzenden Datenhaltungskonzepte beleuchtet und die Zweckmäßigkeit bzw. Unabdingbarkeit eines Ansatzes für die Realisierung von entsprechenden Lösungen bewertet.

6.1 Mehrwert durch integrierte relationale Datenhaltung

Im Folgenden Abschnitt werden neben den explizit durch relationale Modellstrukturen erzielbaren Mehrwertaspekten auch jene Fortschritte diskutiert, die sich allgemein durch die Migration dateibasierter Geodatenbestände in Datenbanken ergeben:

1. **Zentrale Benutzerverwaltung**

   Die Datenbank-basierte Verwaltung aller Geodaten ist Voraussetzung zur feingranularen Vergabe von Zugriffsrechten und somit unabdingbar für die Qualitätssicherung des vorgehaltenen Datenbestandes bei maximaler Nutzung. Die Art des zugrundeliegenden Modellierungsansatzes ist dabei unerheblich. Bei der Migration eines mittels Dateisystem verwalteten Geodatenbestandes ist insbesondere von Interesse, ob sich die hier bereits gewährten Zugriffsrechte mit in das DBMS transferieren lassen.

   Durch LDAP-Server (Lightweight Directory Access Protocol) wie etwa Oracle Directory Server (OID) oder Microsoft Active Directory (ADS) lassen sich (unternehmensweite) Verzeichnisdienste realisieren, die die Authentifizierung der Benutzer in einem Repository zentralisieren [ORACLE (3), 2004]. Anstatt der mehrfachen Anmeldung an verschiedenen Datenbanken und Applikationen reicht dann die einmalige Betriebssystem-Authentifizierung (Single Sign-On) aus. Bei der Zentralisierung von Datenbank-Authentifizierungen werden die Rechte der jeweiligen Benutzer über globale Rollen im Verzeichnisdienst bestimmt, denen (herkömmliche) Rollen in der Datenbank zugeordnet sind [VETTER, 2003].

   Trotz der verfügbaren Konzepte zur Synchronisation von Benutzerverwaltungen und Zugriffsrechten lässt sich der Migrationsaufwand dadurch nicht reduzieren, insbesondere da

- ein Benutzer durch die Abbildung von Grafikdateien in Datenbankstrukturen seines Datenbankschemas zum Besitzer der Daten wird und somit automatisch über Vollzugriff verfügt, unabhängig von den Zugriffsrechten, die ihm auf Dateisystem-Ebene zugewiesen waren.
- die Vergabe von Zugriffsrechten auf Datenbank-Tabellen bis auf Attributebene reicht, wohingegen Zugriffe auf Dateien lediglich global (d.h. im Fall von Geodaten auf den gesamten Geodatensatz einschließlich aller direkten Attribute) erteilt werden können.
  Tatsächlich werden hier die erweiterten Möglichkeiten der Datenbank zur Beschränkung des Zugriffs auf ausgewählte Attribute bislang nicht ausreichend von den GIS-Herstellern ausgeschöpft. Entsprechendes gilt für die Beschränkung des Zugriffs auf Untermengen von Geoobjektklassen (Grants auf Sichten zur Beschränkung des Zugriffs auf Selektion bestimmter Tabellentupel).

Generell muss festgehalten werden, dass die automatisierte Gewährung von Zugriffsrechten immer mit einem erhöhten Sicherheitsrisiko verbunden ist.

2. **Validierung auf geometrische Konsistenz**

   Beim Laden von Vektordaten wird der zu migrierende Geodatensatz durch Validierungswerkzeuge auf geometrische Konsistenz überprüft. So werden z.B. nicht-geschlossene Polygone oder zu Null-Längen entartete Linien-Objekte zurückgewiesen und vom Datenimport ausgeschlossen. Bei den herkömmlichen proprietären Grafikformaten fehlen Mechanismen, um die geometrische Konsistenz zu jeder Zeit zu gewährleisten.

3. **Realitätsnahe Modellierung von Geobjekten und deren Beziehungen**

   Die Modellierung existenzieller Abhängigkeiten von Instanzen zweier Geoobjektklassen durch referentielle Integritäten zwischen Detail- und ihren Mastertabellen sowie die Anwendung von Constraints auf direkte Attribute von Geoobjekten erlaubt die realitätsnahe Abbildung von anwendungsrelevanten Ausschnitten der realen Welt. Die Rolle des Datenbanksystems bleibt damit nicht auf die Speicherung und Verfügbarmachung von singulären Geodatensätzen beschränkt - vielmehr können die verfügbaren Modellierungsmittel und Speicherstrukturen auch zur Abbildung und persistenten Verwaltung der Beziehungen zwischen räumlichen und nichträumlichen Objekten genutzt werden.

   Im Vergleich zur herkömmlichen (Datei-basierten) Speicherung von Vektordaten wird der größte Fortschritt erzielt durch die Möglichkeit zur:

   - Definition von Gültigkeitsbereichen (Domains) in Form von „Coded Values“ bzw. „Range Domains“: Sicherung der attributiven Konsistenz bei der Fortführung
   - Definition von Subtypen zur Strukturierung des Datenbestandes ohne Duplizieren der Geodaten (vgl. Abschnitt 4.4.1)

4. **Mehrbenutzerzugriff und Transaktionskonzept für Geometriedaten (lange Transaktionen)**

   Ein kontrollierter Mehrbenutzerbetrieb ist nur auf Basis einer GIS-Systemplattform zu realisieren, die auf einem leistungsfähigen DBMS aufsetzt, so dass bei Änderungstransaktionen auf verlässliche Synchronisationsmechanismen zurückgegriffen werden kann. Die hier etablierten sperrenden und nicht sperrenden (pessimistischen, optimistischen) Ansätze unterscheiden sich nicht nur technisch-methodisch sondern gerade auch unter organisatorisch-strategischen Gesichtspunkten wesentlich voneinander.

   Während sperrende Verfahren davon ausgehen, dass mehrere Benutzer die gleichen Daten zur gleichen Zeit editieren und sich über sog. *Exclusive Locks* vor der Transaktion das Bearbeitungsrecht sichern, prüfen nicht sperrende Konzepte erst am Ende der Transaktion, ob Konflikte zwischen unterschiedlichen Benutzern entstanden sind.
   Ist die technische Realisierbarkeit der Konfliktanalyse und -Lösung hier unproblematisch, verhält es sich mit der Organisation und Abstimmung des Abgleichs von mitunter zahlreichen Versionen, die häufig von Benutzern an unterschiedlichen Stellen bearbeitet werden, ganz anders.

   Den optimistischen Ansätzen liegt i.d.R. ein proprietäres, relationales Datenbankmodell zugrunde, das die verschiedenen Versionen, ihre Besitzer und die von ihnen durchgeführten Änderungen in Form von Add- und Delete-Tabellen persistent verwaltet. Es ähnelt somit dem Update-Modell eines temporalen GIS.

Auf diese Weise ist die Repräsentation der Geodatenbasis für jeden Benutzer zu bestimmten Zeitpunkten möglich, ohne die gesamten Daten duplizieren zu müssen.

Dieses Versionierungskonzept bildet die Grundlage für:

- die bei der Verarbeitung von Geodaten essentiellen langen Transaktionen
- die Steuerung der Projektentwicklung (Modellierung von „What-if Szenarios“)
- Historienführung: Speichern von Snapshots einer Geodatenbank

Die Komplexität des proprietären Datenmodells, bestehend aus diversen Systemtabellen, Constraints, Trigger und Indizes des GIS-Herstellers, bleibt für den Anwender durch den Einsatz des entsprechenden GIS-Frontends weitgehend verborgen.
Die Frage nach dem Sinn, ein Versionierungskonzept für den gesicherten Multiuser-Betrieb Hersteller-unabhängig und unmittelbar in einem ORDBS zu implementieren, muss vor dem Hintergrund relativiert werden, dass das gezielte Editieren von Geometrie-Segmenten ohne den Einsatz eines GIS wenig anwenderfreundlich erscheint. Das Verarbeiten von Third-Party Geometrien in einem GIS erfordert dagegen bislang die Registrierung der entsprechenden Geometrietabellen respektive -Spalten über die Middleware-Komponente des GIS-Herstellers, wodurch die Systemabhängigkeit bestehen bleibt.

5. **Ressourcen-Transparenz bzgl. des vorhandenen Datenangebots für lesende Zugriffe**

   Die Abbildung aller Geodaten, ihrer zentralen und Datensatz-begleitenden Metadaten sowie der Zugriffsberechtigungen in relationalen Datenbanktabellen oder -Sichten erlaubt die gezielte Bündelung der Geoobjekt-Komponenten zu vollständigen, benutzerspezifischen Gesamtsichten. Damit lässt sich die für einen Benutzer aktuell verfügbare Geodatenbasis individuell im Internet präsentieren. Auf Basis der zugrunde liegenden Relationen lassen sich leicht eigene GUI entwickeln, die die Navigation und Selektion nach / von anwendungsrelevanten Geodatensätzen unterstützen (vgl. Abschnitt 4.3.1).

   Eine Alternative zur oben beschriebenen Form der Transparenzsteigerung durch selbst entwickelte Benutzerschnittstellen stellen die proprietären Metadaten-Server-Lösungen der GIS-Hersteller dar, die auf den binären (Datensatz-begleitenden) Metadatenstukturen (XML) aufsetzen. Dabei kann der GIS-Administrator die Metainformationen von ausgewählten Geodatensätzen zu einem zentralen Metadaten-Repository zusammenfassen, das dann den verfügbaren Geodatenbestand für externe Zugriffe (mittels GIS-Clients des Herstellers) dokumentiert.
   Neben der Herstellerabhängigkeit muss hier explizit das ungelöste Problem der Konsistenzsicherung zwischen Datensatz-begleitenden und zentralen Metadaten genannt werden, das insbesondere auf fehlende referentielle Integritäten zwischen Binärdaten (XML) und Relationen zurückzuführen ist.
   Zur Überwindung dieser Einschränkung bietet sich demnach die objektrelationale Speicherung von XML in Kombination mit relationalen Sichten zur Erzeugung zentraler Metadaten-Kataloge an (vgl. Abschnitt 5.1.3).

6. **Flexibler Austausch von Geodaten und Metadaten über das Internet**

   Klassische Relationen als einziges logisches Konstrukt zur synchronen Verwaltung von Geo- und Metadaten inkl. aller gewährten Zugriffsrechte bilden die Basis für kontrollierte lesende und schreibende Transaktionen auf die Geodatenbank.
   Darauf lassen sich leicht dezentrale Fortführungskonzepte aufsetzen, die - ggf. unterstützt durch geeignete individuell angepasste Benutzerschnittstellen - die dezentrale Fortschreibung von Geodatenbasen gestatten. Dadurch wird einerseits der zentrale Fortführungsaufwand minimiert und andererseits die Beschreibung der Geodaten auf die hierfür einzig kompetenten Fachanwender übertragen (vgl. Abschnitt 4.3.2).
   Dieser Fortführungsansatz ist mit dem moderner GDI vergleichbar, mit Ausnahme, dass die Datenhaltung nicht verteilt sondern zentral, jedoch mit unmittelbarem Web-Zugriff der Akteure von aussen erfolgt. Die Rollen von Datennutzer und Datenanbieter verschwimmen dadurch zusehens.

   Auf Grundlage der einheitlich relationalen Modellierungsebene lassen sich demnach Automatismen zur kontrollierten Fortschreibung von zentral verwalteten Geodatenbeständen schaffen, in die von der GIS-Software abgeleitete und synchronisierte implizite Metadaten sehr einfach eingebunden werden können.

Die Interpretation der proprietären, binären Tabellenspalten für direkte Zugriffe auf Vektor-, Raster- und Metadaten sowie für offene Zugriffe über OGC-Schnittstellen eines OWS erfordert allerdings zwingend den Einsatz der Middleware-Komponente des GIS-Herstellers bzw. der entsprechenden Objektbibliotheken.

7. **Skalierbarkeit der Systemplattform**

   Die wachsende Bedeutung der Skalierbarkeit von DBS für Anwendungen der Geoinformatik ist primär zurückzuführen auf

   - gewaltige Datenvolumina infolge ausgereifterer Technologien zur Datenbank-basierten Verwaltung von Geodaten (insbesondere Rasterdaten) auf Basis von Objekttypen und binären Datentypen
   - steigende Benutzerzugriffe infolge der Web-Fähigkeit zentraler Geodatenbanken
   - zunehmende Vernetzung von GIS-Anwendungen und serverseitige Geodatenverarbeitung

   Die Möglichkeiten zur Anpassung eines DBMS an sich ändernde Benutzer- und Transaktionslasten basieren auf der Erweiterbarkeit des Netzwerkes, der Serversysteme, der Datenbanken und Anwendungen durch Hinzufügen zusätzlicher Hardware. Während Desktop-DBS wie etwa Microsoft Access aufgrund des Systemaufbaus nicht skaliert werden können und sich daher nur für den Einsatz auf Einzelplatzrechnern oder in kleinen Netzwerken mit limitierter Datenbankgröße und einer begrenzten Anzahl von Benutzern eignen, können die leistungsstarken Client-Server-Datenbanksysteme durch entsprechende Skalierung nahezu bis an die theoretischen Grenzen (z.B. mehrere TB max. DB-Größe) ausgedehnt werden.

   Bei aktuellen Technologien zur maximalen Skalierbarkeit (High-Level-Scalability) und Hochverfügbarkeit durch die Vernetzung weltweit verteilter Rechner und ihrer Datenbestände stehen neue verteilte Systemarchitekturen im Fokus, die unter den Schlagworten *Grid-Computing*, *Clustering* und *Peer-to-Peer-Systeme* populär geworden sind.

8. **Verfügbarkeit von zentralen, Backup- und Recovery-Mechanismen**

   Die aus wirtschaftlichen Überlegungen sinnvolle Integration von Geodaten und Unternehmensdaten führt zum Aufbau speicherintensiver Datenbankumgebungen. Im Vergleich zum gesamten Datenvolumen ist der Speicherumfang, der der Fortschreibung des Geodatenbestandes zuzuordnen ist, in der Regel gering. Intelligente Backupkonzepte fortschrittlicher DBMS wie inkrementelle Backups sind somit umso mehr von Bedeutung.

   Mit wachsender Größe der Datenbank wird das Erstellen von Backups bzw. des Recovery zu einem immer kritischeren Zeitfaktor. Die Erstellung der Datensicherung nimmt immer mehr Zeit in Anspruch, während der Datenbankadministration immer kleinere Zeitfenster zugestanden werden. Im Falle eines Recovery muss die Datensicherung wieder eingespielt werden, was wiederum einiges an Zeit benötigt und für einen längeren Systemausfall sorgt.
   Die Hot Backup-Fähigkeit von DBS, die die Datensicherung im laufenden Betrieb ermöglichen, ist vor diesem Hintergrund unverzichtbar. Weiterführende Konzepte stellen z.B. Inkrementelle Backups dar, die sich auf die Sicherung der Änderungen seit der letzten Datensicherung beschränken. Dadurch muss der einzelne Sicherungsvorgang weniger Daten kopieren und ist damit schneller. Die Zusammenführung eines Basisbackups mit einer Reihe von inkrementellen Backups durch einen ausserhalb der Datenbank ablaufenden Prozess beschleunigt auch das Recovery, da nur ein Gesamtbackup eingespielt werden muss ([DURBEN, 2004]).

   Neueste Entwicklungen im Bereich des Datenbank-Recovery sehen zusätzlich Funktionalität für datierte Reverse-Rollbacks vor, d.h. die anwenderfreundliche und schnelle Wiederherstellung eines konsistenten Datenbank-Zustands beginnend vom aktuellen Zeitpunkt anstelle des Einspielens eines Vollbackups inkl. mehrerer Roll Forwards.

6.2 Mehrwert durch integrierte objektrelationale Datenhaltung

Um den im Folgenden aufgeführten konzeptionellen Anforderungen zu begegnen, ist das logische Datenbankmodell des objektrelationalen Ansatzes zwingend erforderlich oder der Einsatz führt gegenüber dem relationalen Modell zu zusätzlicher Effizienzsteigerung.
Da die räumlichen Basisdatentypen der ORDBS nativ von den führenden GIS als weiteres Speicherformat unterstützt werden, ist die Entscheidung pro objektrelationaler Ansatz mit keinerlei Einschränkungen in Bezug auf die unter Abschnitt 6.1 diskutierten Anforderungen verbunden.

1. **Zusammenführung und Analyse von Geodaten**

   Zusätzlich zu den Overlay-Funktionen der GIS-Software gestattet der objektrelationale Ansatz die Nutzung der in ORDBS verfügbaren räumlichen Operatoren und Funktionen und damit die einfache Kombination der raumbezogenen Analyse mit Data Mining Konzepten und statistischen Auswertungsverfahren.
   Die Ableitung veränderlicher Information von gespeicherter statischer Attributinformation der Geoobjekte lässt sich beim objektrelationalen Ansatz sehr einfach über geeignete Methoden realisieren. Die Funktions-Logik liegt somit zentral bei den Daten in der Datenbank und kann von jedermann, der über entsprechende Zugriffsrechte verfügt, unmittelbar im Netzwerk genutzt werden.

2. **Offener Zugriff auf die Geodatenbasis**

   Der systemunabhängige, direkte, SQL-basierte Zugriff auf Geometrietabellen (ohne SQL-API der Middleware) ermöglicht die Entwicklung von GIS-Anwendungen und die Analyse raumbezogener Daten ohne auf ein GIS zurückgreifen zu müssen. Die Anwendungspalette reicht vom einfachen dezentralen Einpflegen von POI für touristische Anwendungen (z.B. Hotelstandorte) bis hin zur komplexen Analyse raumzeitlicher Prozesse, wie die in dieser Arbeit realisierten Anwendungen der Echtzeit-Verfolgung von Wildtieren und der Totholzausbreitung (vgl. Abschnitte 5.3.1 und 5.3.2).

   Eine kurze Diskussion der Realisierung von Interoperabilität erscheint an dieser Stelle angebracht:

   Die Interoperabilität zwischen GIS verschiedener Hersteller stellt zweifelsohne ein erstrebenswertes Ziel dar, die Umsetzung auf Basis eines Industriestandards wie Oracle Spatial ist aus idealistischer Sicht jedoch skeptisch zu bewerten. Das geometrische Objektmodell der OGC Simple Features Specification for SQL ist rein konzeptionell und verzichtet auf die genaue Vorgabe der Implementierung. Neben der Variabilität der Speicherstrukturen lässt die Definition von gültigen Geometrieobjekten ausserdem gewisse Freiheiten bei der Implementierung durch die GIS- bzw. ORDBMS-Hersteller zu. So definiert ESRI die äussere Begrenzung eines Polygons mit Löchern im Uhrzeigersinn, die innere Begrenzung entgegen des Uhrzeigersinns. Oracle geht genau entgegengesetzt vor. Die Probleme zeigen sich - trotz zweier OGC-konformer Produkte - dann bei der Validierung der Shapes mit den Validierungs-Funktionen von Oracle Spatial.

   Das eingeschränkte Geometrie-Modell der Simple Features mit der fehlenden expliziten Deklaration von topologischen Beziehungen, der Beschränkung auf den zweidimensionalen Raum sowie auf lineare Geometrien lässt erahnen, dass die derzeitige Version wenig geeignet ist, um interoperable GIS-Anwendungen zu realisieren. Künftig werden hier ISO 19107 (Spatial Schema) und ISO 19136 (GML) zunehmend an Bedeutung gewinnen, die über ein deutlich erweitertes Geometrie-Objekt-Modell verfügen. GML als standardisiertes (neutrales) Speicherformat bietet sich an, um in Bezug auf die Verwaltung von Geometriedaten transparente Speicherstrukturen zu schaffen, die sowohl von den GIS-Systemen der Hersteller als auch direkt über SQL interpretiert werden können.

3. **Flexible Modellierungsmittel zur vollständigen Abbildung von raumbezogenen Daten in 2D und 3D inklusive Zeitbezug**

   Der objektrelationale Ansatz bietet gegenüber dem relationalen wesentliche Vorteile beim Aufbau von drei- bzw. vierdimensionalen Topologien, da Maschen-, Kanten-, Knoten-Hierarchien sehr viel zweckmäßiger in Kollektionen anstelle von vielen - über Fremdschlüssel referenzierten - singulären Tabellen verwaltet werden. Damit werden teuere Verbundoperationen für die Aggregation der Objektsegmente a priori ausgeschlossen. Der R-Tree ist in der Lage, multidimensional zu indizieren, was für die performante Verfolgung der Historie von Geoobjekt-Knoten im Raum-Zeit-Kontinuum genutzt werden kann. Die CAD-orientierten Ansätze bei

der Entwicklung von echten 3D- bzw. 4D-GIS, die die in GIS verfügbare Geoprocessing-Funktionalität vollständig auf mehrdimensionale Räume übertragen, profitieren somit wesentlich von den flexiblen objektrelationalen Datenbankstrukturen.

Für den zweidimensionalen Raum ergeben sich weitreichende Möglichkeiten zum Aufbau einer Versionsverwaltung, die gegenüber den auf dem relationalen Modell aufsetzenden Lösungen der GIS-Hersteller wesentlich kompakter und damit für den Anwender transparenter ist. Objektrelationale Geometrietabellen können sämtliche Versionen von Geoobjekten inkl. aller direkten Attribute und ggf. einer Beschreibung der fachlichen Bedeutung aufnehmen. Als Alternative zur Vergabe des Zeitstempels über direkte Attribute bietet sich die Verknüpfung von Geoobjekten mit diskreten oder kontinuierlichen Zeitwerten mittels Referenzen an. Das Traversieren über Referenzen mittels GIS erfordert jedoch zunächst die Erweiterung der bestehenden Datenbankschnittstellen um objektrelationales SQL.

4. **Wahrung der Konsistenz zwischen Geometrie-, Sach- und Metadaten**

   Aus Anwendersicht macht es wenig Sinn, die attributive und geometrische Charakteristik von Geoobjekten auf unterschiedliche Relationen aufzuteilen, wie dies beim relationalen Datenhaltungskonzept der Fall ist. Dieser aus Performanzgründen motivierte Ansatz macht das Datenbankmodell komplexer und erhöht die Systemabhängigkeit, da zwingend das jeweilige GIS eingesetzt werden muss, um Inkonsistenzen zwischen Geometrie und „direkten“ Attributen zu vermeiden. Entsprechend verhält es sich mit den zugehörigen Metadaten, die ebenfalls binär kodiert in referenzierten Systemtabellen liegen.
   Der in Abschnitt 2.1 vorgestellte 3. Ansatz, nach dem sämtliche Konsistenz-sichernden DDL-Strukturen transparent an den Anwender weitergegeben werden, ist eher theoretischer Natur und bleibt in der Praxis nahezu unberücksichtigt.

   Objektrelationale Geometrie-Typen ermöglichen die Bündelung von Geoobjekt-Komponenten in einer Tabelle und reduzieren damit das Risiko des Konsistenzverlusts deutlich. Der Bündelungsprozess kann durch die strukturierte Beschreibung der fachlichen Thematik von Geoobjekten auf Basis weiterer, geeigneter Basisdatentypen (XMLTYPE) konsequent fortgesetzt werden. Dies setzt allerdings voraus, dass diese Datentypen ebenfalls nativ von GIS unterstützt werden.

   Metadaten beschreiben Geodaten und nicht die den Geodaten zugrunde liegenden Geoobjekte, daher sollten sie auch in eigenen, externen Tabellen verwaltet werden, jedoch entgegen des relationalen Ansatzes offen interpretierbar bleiben. Insofern bietet sich erneut der Einsatz des objektrelationalen Basisdatentyps XMLTYPE an.

5. **Standardisiertes und konsistentes Handling von Annotation**

   Die Speicherung von Betextung in der Datenbank zur Beschriftung von Geoobjekten ist bislang von den GIS-Herstellern zwar implementiert, jedoch ebenfalls proprietär realisiert. Das Speicherschema für Annotation basiert wie das der Geodaten beim relationalen Datenhaltungsansatz entweder auf Binärstrukturen oder aber auf klassischen Textfeldern bzw. Objekttypen.

   Die Standardisierung von Annotation auf Basis eines geeigneten, einheitlichen Objekttyps, der neben dem Text die Platzierungsinformation in Form einer Koordinate und eines Drehwinkels aufnimmt, scheint hier zweckmäßig. Die Realisierung von *Feature-Linked Annotation* - also die Änderung des Platzierungspunktes des Labels bei räumlicher Änderung des zugehörigen Geoobjekts - könnte hier durch die Berechnung des Feature-Schwerpunktes über die *Controid-Function* erfolgen.

6. **Modellierbarkeit bekannter komplexer Objekte und deren Beziehungen**

   Die Fülle der in Kapitel 3 vorgestellten objektrelationalen Datenbankstrukturen erlaubt eine wesentlich kompaktere und konsistentere Modellierung von komplexen Objekten und ihren Beziehungen, sofern diese Komplexität vor dem Datenbankentwurf bekannt ist. Anders verhält es sich mit der Komplexität von Objekten und dynamischen Prozessen, die auf das Zusammenwirken vieler, z.T. unbekannter Einflussgrößen zurückzuführen ist, wie etwa bei Phänomenen der Natur.

Ein Ausschnitt der realen Welt muss bzgl. seiner Geoobjekte und Beziehungen vollständig verstanden und bekannt sein, um ihn möglichst realitätsnah in der Datenbank abbilden zu können, andernfalls ist das stärkste Modellierungswerkzeug wertlos.

Für die Analyse von komplexen Prozessen müssen darüber hinaus sämtliche, auf die Entwicklung Einflussnehmende Objektwerte in der erforderlichen räumlichen und temporalen Auflösung verfügbar sein.

Der Komplexität von Geoobjekten begegnet man in der Regel durch die Definition entsprechender (verschachtelter) Objekttypen, die dann zur Attributierung der Geometrie-Tabellen herangezogen werden. Die Komplexität von Beziehungen tritt durch Kollektionen, (mehrfach)verschachtelte Tabellen und / oder Referenzen in Erscheinung. Die low-level Datenbankschnittstelle zwischen aktuellen GIS und ORDBS verhindert jedoch die Nutzbarmachung des Modellierungspotentials für die Geoinformatik.

6.3 Bewertung für die Anwendungspraxis

6.3.1 Getrennte Datenhaltung

Im Folgenden wird der aktuell erzielbare Nutzen und die Grenzen des getrennten, des integriert relationalen sowie objektrelationalen Datenhaltungsansatzes für die Anwendungspraxis bewertet. Die Zweckmäßigkeit eines Ansatzes ergibt sich dabei aus typischen Rahmenbedingungen, die durch die qualitativen und funktionalen Anforderungen sowie die Anwendung selbst beeinflusst werden.

Der klassische getrennte Datenhaltungsansatz hat durch die neuen aber inzwischen mehr und mehr ausgereiften Konzepte zur integrierten Verwaltung von Geodaten nicht an Bedeutung verloren und wird für die Masse der GIS-Anwender auch künftig der erste Weg der Datenorganisation bzw. Modellierung bleiben, wenn die Anwendung das Prinzip der Datenintegration zur Nutzung heterogener Datenquellen erfordert. Das Konzept der getrennten Datenhaltung weist gegenüber den integrierten Ansätzen die geringste Einstiegshürde für die lokale Nutzung von Geodaten auf, da jedermann mit Dateien und deren Organisation in Dateisystemen vertraut ist. Die Zweckmäßigkeit des Ansatzes ist insbesondere gegeben

- bei GIS-Projekten geringer Laufzeit, bei welchen nach Abschluss des Analyse- und Visualisierungsprozesses die Geodaten gelöscht oder zumindest nicht mehr fortgeführt werden.
- wenn die breite, externe Nutzbarmachung der Geodaten sekundär ist.
- wenn die rasche Nutzung und Visualisierung der Geodaten im Vordergrund steht.
- wenn kein Mehrbenutzerbetrieb erforderlich ist.
- wenn keine Versionierung zum Zwecke der Projektorganisation notwendig ist.
- wenn ein Höchstmaß an Flexibilität zur einfachen Strukturierung heterogener Geodaten und Dokumente gewünscht wird (Dateisystem).
- bei zu geringem Know-How zur Administration und Konfiguration einer Client-Server-Datenbank.
- bei begrenztem Budget.

Zur Sicherung der attributiven Konsistenz von Vektordaten sowie zur Modellierung von Objektbeziehungen durch Integritätsbedingungen bieten manche GIS-Hersteller das Geodatenmanagement auf Grundlage eines Desktop-Datenbanksystems an, was entsprechend mit enormen Einschränkungen u.a. in Bezug auf die maximale Anzahl von Objekten und Speicherkapazität verbunden ist. Ein Mehrbenutzerbetrieb auf dieser Basis ist ausgeschlossen.

6.3.2 Integrierte Relationale Datenhaltung

Der Wechsel zu einem der beiden integrierten Datenhaltungsansätze kommt einem Quantensprung gleich - insbesondere hinsichtlich der Sicherung der Qualität der vorgehaltenen Geodaten -, da sämtliche seit Jahrzehnten bewährte und erprobte Datenbankkonzepte erstmals auch direkt für Geodaten genutzt werden können. Dabei ist es unerheblich, ob der relationale oder objektrelationale Ansatz gewählt wird.

Der durch das Setup und die Konfiguration eines leistungsfähigen Geodatenbanksystems sowie durch die Migration sämtlicher Geodaten bedingte Mehraufwand ist gerechtfertigt bzw. erforderlich wenn

- der Aufbau eines Geoinformationssystems mit langfristiger Nutzung im Vordergrund steht.
- die breite Nutzung der vorgehaltenen Datenbestände gewünscht ist, somit externe Zugriffe über das Intranet / Internet auf die Geodatenbasis anfallen.
- die Zusammenführung von Geodaten mit Unternehmensdaten zur Nutzung des Raumbezugs in Wirtschaftsprozessen vorrangiges Ziel ist.

- von mehreren Benutzern schreibend auf die Geodaten zugegriffen werden muss, somit der gesicherte Mehrbenutzerzugriff essentiell ist.
- die Entwicklung von realitätsnahen, Anwendungs-spezifischen Geodatenmodellen angestrebt wird.
- auf Versionierungskonzepte zur Historienführung oder zur Organisation von Projekten zurückgegriffen werden muss.
- anspruchsvolle, Konsistenz-erhaltende Datensicherungs-Mechanismen für Geodaten genutzt werden sollen.
- individuelle Benutzerschnittstellen zum Zugriff auf die Geodatenbasis entwickelt werden sollen.

Die Entscheidung für oder gegen den Einsatz des getrennten bzw. eines integrierten Datenhaltungskonzepts wird primär durch die konkreten qualitativen und funktionalen Anforderungen des Anwenders an das Datenmanagement beeinflusst, eine untergeordnete Rolle spielt dagegen die eigentliche Anwendung selbst. Der Migrationsaufwand ist dank flexibler und mittlerweile auch zuverlässiger, im Batchbetrieb arbeitender Migrations-Tools überschaubar, so dass der finanzielle Spielraum und das vorhandene Know-How zur Administration der Geodatenbank die KO-Kriterien für einen möglichen Architekturwechsel darstellen. Abb. 6.1 skizziert den Abwägungsprozess zwischen Aufwand und Nutzen.

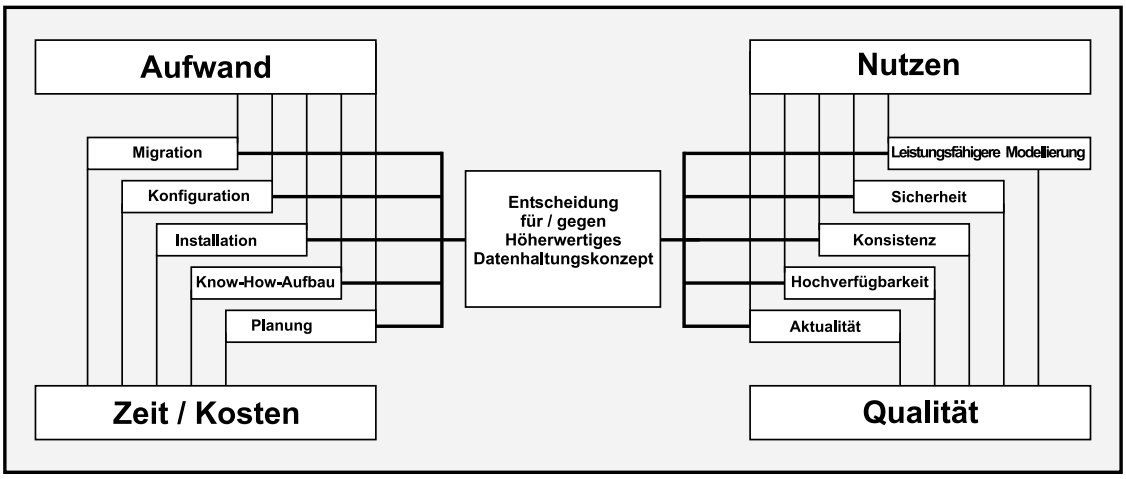

Abb. 6.1: Architekturwechsel: Abwägung zwischen Aufwand und Nutzen

Für den Fall, dass der integriert relationale Ansatz die einzige Alternative für leistungsfähigeres Geodatenmanagement darstellt, insbesondere wenn das DBMS keine räumlichen Objekttypen anbietet, wird die Migration durch eine noch höhere Systemabhängigkeit erkauft. Die bei der Abbildung von GIS-Dateiformaten in das proprietäre Relationenmodell eines GIS-Herstellers entstehenden (binären) Datenbankstrukturen lassen sich nur von den GIS-Produkten dieses Herstellers ausreichend interpretieren. Für den Fall, dass Geodaten zwischen zwei Herstellersystemen physisch ausgetauscht werden sollen, müssen die Binärdaten der Geometrietabellen zunächst wieder in das Ausgangs-Grafikformat konvertiert werden, ehe das Zielsystem über Konverter bedient werden kann. Vielversprechend ist hier die Alternative des Modellbasierten Datenaustauschs, sofern Regeln für die automatische Herleitung des Transferformats sowie Prozessoren zum Export und Import der Geodaten in das bzw. aus dem Transferformat existieren (vgl. Abschnitt 4.2.1).

6.3.3 Integrierte Objektrelationale Datenhaltung

Ein auf räumlichen Objektypen eines ORDBMS basierendes Datenmanagement ist aus Sicht der Anwenderpraxis dem relationalen Ansatz vorzuziehen bzw. erforderlich, wenn darüber hinaus

- GIS-Anwendungen systemunabhängig auf Basis der räumlichen Erweiterung des Datenbanksystems entwickelt werden oder parallel Zugriffe via SQL und das GIS-Frontend anfallen (Sicherung der Konsistenz zwischen Geometrie und direkten Attributen).

- Massendaten direkt über Erweiterungen der Datenbank analysiert werden (z.B. Spatial Data Mining).
- Location-Based Services oder raumbezogene Echtzeit-Anwendungen, insbesondere für das Monitoring oder Auskunftsysteme entwickelt werden (vgl. Real Time Tracking Anwendung 5.3.2).
- für die objektorientierte Anwendungsentwicklung Objekte persistent in der Datenbank gespeichert werden sollen.

Das wesentlich kompaktere Datenmodell des objektrelationalen Ansatzes lässt darüber hinaus die systemunabhängige Entwicklung historischer Auskunftssysteme zu, bei denen die Verknüpfung von Geoobjekten mit flexiblen diskreten und kontinuierlichen Zeittypen im Vordergrund steht.

Das besondere Potential von OR-Strukturen zur Entwicklung von echten 3D- und 4D-GIS als weiterer in Abschnitt 6.2 diskutierter Mehrwertaspekt ist Gegenstand der aktuellen Forschung. Die interoperable Nutzung von Geodaten auf der Grundlage eines Datenbank-basierten Industriestandards sowie die Homogenisierung der proprietären Implementierungen von Annotation in GIS ist dagegen eher idealistischer Natur und steht - insbesondere nachdem der Ansatz des offenen Zugriffs über OGC-Schnittstellen den Nachweis seiner Leistungsfähigkeit längst erbracht hat - nicht mehr so sehr im Fokus der Forschung.

Tab. 6.1 gibt eine Übersicht über die Vorzüge und Grenzen der diskutierten Datenhaltungsansätze:

- unzureichend (unterstützt) / Nachteile
\+ ausreichend (unterstützt) / Vorteile
(+) bedingt (unterstützt)

Kriterium	**Referenz-GIS Desktop Getrennt**	**Referenz-GIS + High-End RDBMS Integriert Relational**	**Referenz-GIS ++ High-End ORDBMS Integriert Objektrelational**
Strukturierbarkeit des Gesamtdatenbestandes	+	-	-
Konsistenz (geometrisch)	-	+	+
Konsistenz (attributiv)	-	+	+
Konsistenz zwischen Geometrie und Attributen	-	(+)	+
Realitätsnahe Modellierung von Geoobjekten	-	+	+
Benutzerverwaltung / Vergabe von Zugriffsrechten	-	+	+
Erweiterbarkeit bzgl. Benutzerschnittstellen	-	+	+
Flexibler Zugriff über SQL	-	(+) über SQL-API	+
Hochverfügbarkeit	-	+	+
Skalierbarkeit	-	+	+
Datensicherungskonzepte	-	+	+
Display-Performance	+	+	(+)
Finanz. Aufwand (Akquisitions- & Wartungskosten)	+	-	-
Erforderliches Know-How	+	-	-
Objektorientierte Anwendungsentwicklung	-	-	+
Dynamische Prozesse	-	(+)	+

Tab. 6.1: Bewertung der Datenhaltung nach qualitativen und funktionalen Gesichtspunkten

6.4 Bewertung der realisierten Anwendungen

Analyse der Totholzausbreitung mittels ORDBS:

Durch die Aggregation von einzelnen Geoprocessing-Funktionen zu einem Gesamtworkflow, der sich mit veränderten Startwerten jederzeit wiederholen lässt, kann das Analysepotential von GIS ideal für bekannte dynamische Prozesse, speziell zur Simulation von Szenarien bei Planungsprozessen ausgeschöpft werden. Zuletzt wurde diese Funktionalität zunehmend von den führenden GIS-Herstellern in ihren Produkten implementiert.

Der im Rahmen dieser Arbeit gewählte Ansatz der vollständig Datenbank-basierten Analyse mit Verkettung von Spatial Functions über Spatial Views demonstriert die Leistungsfähigkeit von räumlichen Objekttypen bzw. darauf definierter Methoden eindrucksvoll. In Bezug auf die Benutzerfreundlichkeit im direkten Vergleich mit einem GIS kann die an den proprietären Dialekt des ORDBMS-Herstellers gebundene SQL-Syntax selbstverständlich nicht punkten. Primär stellt jedoch hier die verfügbare Funktionalität an sich und nicht die leichte Bedienbarkeit das Kriterium dar, zumal sich die Benutzerfreundlichkeit über selbstentwickelte GUIs leicht erzielen lässt.

Echte Vorteile bietet der ORDBS-basierte Ansatz bei der statistischen Auswertung von Massendaten bzw. der Aufdeckung von Datenmustern mittels Data Mining Konzepten der Datenbank, da der Datentransfer in Fremdsysteme über das Netzwerk entfällt.

Real Time Tracking mittels ORDBS:

Die Anwendung zeigt das Potential, das die Verknüpfung von Raum und Zeit in hoher Auflösung mit räumlichen Analysefunktionen in WEB-fähigen Datenbanken bietet. Der Nutzen wird nicht durch die präzise Abbildung komplexer Geo(metrie)-Objekte sondern durch die Masse an Datenbankeinträgen pro Zeiteinheit und die Definition einer Transaktions-Chronologie (Inserts $\Rightarrow$ räumliche Selects) über die Mittel der aktiven Datenbank (Trigger) generiert. Die Anwendung erfordert hochauflösende temporale Datentypen eines DBMS, die die Abbildung der Zeit in Sekundenbruchteilen erlauben. Die konkreten Anforderungen dieser Anwendung führen dazu, dass ORDBS hier den einzigen Lösungsansatz darstellen.

Die Klassifizierung und konsistente Fortführung des Wegenetzes sowie die Modellierung existentieller Abhängigkeiten - demonstriert am Anwendungsbeispiel des Verkehrsnetzes des Igelbussystems - durch Nutzung des GIS oder geeigneter CASE-Tools (Computer Aided Software Engineering) als Modellierungswerkzeug basiert ausschließlich auf dem Relationenmodell. So wird die Ableitung von Subtypen aus übergeordneten Objektklassen etwa durch triviale Fremdschlüsselbeziehungen, existentielle Abhängigkeiten zwischen verschiedenen Objektklassen mittels referentieller Integritäten modelliert. Die Anwendungen lassen sich somit unabhängig vom zugrunde liegenden Datenbanksystem entwickeln - die Nutzung eines ORDBS führt hier zu keinerlei Mehrwert.

Bzgl. der Darstellungs-Performance von Vektordaten bei der Gestaltung kartographischer Visualisierungen verzeichnet die proprietäre, binäre Speicherung des relationalen Ansatzes in der Regel ein kleines Plus gegenüber der Speicherung auf Basis von Objekttypen. Die Unterschiede treten insbesondere dann in Erscheinung, wenn sämtliche Daten (also alle Stützpunkte) für ein Gebiet großer Ausdehnung aus den Datenbanktabellen ausgelesen und dargestellt werden müssen. Bei gezielten Anfragen auf konkrete Geoobjekte ist hier kein Unterschied zwischen den beiden Ansätzen zu verzeichnen. Die enorme Display-Performance der für die schnelle graphische Darstellung optimierten Vektorformate wird von keinem der beiden integrierten Datenhaltungskonzepte ganz erreicht.

6.5 Fazit

Der Leistungsgewinn durch die Migration von Datei-basierten Geodatenbeständen in ein integriertes Datenhaltungskonzept ist enorm und bedeutet Mehrwert insbesondere in Bezug auf die Modellierungsfunktionalität und die breite Nutzbarmachung und Sicherung der Qualität von Geodaten.

Auf Basis des Relationenmodells können bereits wesentliche konzeptionelle Anforderungen, die der GIS-Anwender an das Geodatenmanagement stellt, erfüllt werden. Aufgrund der bislang wenig weit fortgeschrittenen Integration von objektrelationalen Datenbankstrukturen in GIS verspricht in funktionaler Hinsicht das objektrelationale Datenhaltungskonzept gegenüber dem relationalen Ansatz für die Anwenderpraxis keine Verbesserungen, sofern der Zugriff ausschließlich über den GIS-Client erfolgt.

Unter qualitativen Gesichtspunkten ist jedoch das objektrelationale Schema aufgrund des kompakteren Datenbankmodells beim Einsatz einer objektrelationalen Datenbank vorzuziehen, da Transaktionen zur Änderung direkter Attribute von Vektordaten alternativ auch über SQL angestoßen werden können ohne die Konsistenz zwischen Geometrie und Sachdaten zu gefährden. Sonstige direkte Änderungen, insbesondere in den Systemtabellen des GIS-Herstellers sind fatal und führen zu Inkonsistenzen. Die Folge sind etwa fehlerhaft im Repository des GIS-Herstellers registrierte Layer, die nicht mehr über die GIS-Software interpretiert werden können.

Der objektrelationale Ansatz lässt die rein Datenbank-basierte Entwicklung zahlreicher GIS-Anwendungen zu, wovon einige bislang ausschließlich direkt über ein ORDBS realisierbar sind. Darüber hinaus bieten objektrelationale Strukturen ausreichend Potential, um einerseits die voherrschenden proprietären Implementierungen der GIS-Hersteller (etwa bei der Speicherung von Datensatz-begleitenden Metadaten) offener zu gestalten und andererseits die Modellierung von Geoobjekten um weitere Komponenten zu ergänzen.

Kapitel 7

Zusammenfassung und Ausblick

Lange Zeit standen Objektrelationale Datenbanken im Fokus der Öffentlichkeit. ORDBMS galten als ideale Lösung, um neben einfach strukturierbaren Massendaten auch komplexe Objekte zu modellieren - entkoppelt von Anwendungsprogrammen - persistent zu speichern und über SQL standardisiert zugänglich zu machen. Die Hoffnungen wurden nicht enttäuscht, ORDBMS bieten dank ihrer Objekttypen die größtmögliche Flexibilität und Offenheit beim Speichern und Zugriff auf komplexe Daten. Sie stellen folgerichtig auch für die Verwaltung von Geodaten und ihrer Metainformation mehr als nur eine Alternative zu relationalen Datenbankmanagementsystemen dar. Die Kombination von räumlichen Operatoren und Funktionen mit Datentypen zur Aufnahme von hochaufgelösten Zeitwerten erlaubt die Analyse von dynamischen Prozessen selbst ohne den Einsatz eines GIS.

Dennoch stagniert der weiterführende Integrationsprozess von ORDBMS in GIS seit geraumer Zeit. Es ist erkennbar, dass die Aufmerksamkeit, die man ORDBS heute entgegen bringt, allgemein merklich nachgelassen hat. Für diese Entwicklung sind zwei Gründe wesentlich:

1. Was bereits entwickelt und verfügbar ist, weckt weniger Interesse.
2. Mit XML steht mittlerweile ein Standard zur Verfügung, der sich hervorragend eignet, um komplexe Objekte strukturiert abzubilden und Objektdaten flexibel über das Internet auszutauschen.

Bezeichnend ist, dass es gerade dank objektrelationaler Technologie nun möglich ist, XML-Daten intelligent in der Datenbank zu verwalten.

Es stellt sich die Frage, welche Faktoren die weiterführende Nutzung objektrelationaler Technologie in der Geoinformatik konkret behindern und welche Empfehlungen für die weitere konstruktive Entwicklung von Geoinformationssystemen und ORDBMS ausgesprochen werden können.

Die Akzeptanz bzw. Nachfrage objektrelationaler Strukturen im Allgemeinen ist aufgrund mangelnder Transparenz der komplexeren Syntax einerseits und des erzielbaren Nutzens andererseits bei den meisten Datenbank-Anwendern noch immer unzureichend. Die mangelnde Aufgeschlossenheit gegenüber OR-Konzepten der meisten Anwender lässt sich auch darauf zurückführen, dass der relationale Datenbankentwurf für die Realisierung der Masse an Datenbank-Anwendungen ausreicht bzw. häufig historisch gewachsene relationale Datenbankmodelle vorliegen, auf die sämtliche Arbeitsprozesse abgestimmt sind.

Mit dem Bedarf an einer stärkeren Nutzbarmachung objektrelationaler Technologie in Geoinformationssystemen im Speziellen verhält es sich ähnlich. Aus Sicht vieler GIS-Anwender ist die Modellierung von Geoobjekten über direkte und indirekte Attribute mit atomaren Werten zur Beschreibung der Kerneigenschaften sowie der Geometrie zur Darstellung der absoluten Position und geometrischen Ausprägung im räumlichen Bezugssystem ausreichend. Der breite Bedarf, nicht aber die speziellen Anforderungen einiger weniger Anwendungen, ist maßgebend für die weitere Entwicklung. Insofern ist die bisherige Zurückhaltung der GIS-Hersteller bei der Integration von OR-Konzepten in GIS nachvollziehbar. Der GIS-Anwender wird nur dann bereit sein, in eine um zusätzliche OR-Konzepte erweiterte, nächste Generation von GIS-Produkten zu investieren, wenn ihm die dadurch zu erzielenden Effizienzsteigerungen vorab deutlich aufgezeigt werden.

Es mangelt somit an Pilotanwendungen, mit denen der Nutzen von objektrelationaler Technologie im Allgemeinen und für raumbezogene Anwendungen im Speziellen besser als bislang demonstriert werden kann. Hier sind insbesondere die **Hochschulen** aufgerufen, durch Grundlagenforschung und Referenzprojekte in enger Kooperation mit den GIS- und ORDBMS-Herstellern das noch nicht ausgeschöpfte Potential zu demonstrieren. Die vorliegende Arbeit leistet hierzu einen Beitrag.

Das Ziel der Eroberung von Kundenstämmen bzw. Marktanteilen durch die breite Ausrichtung der Middleware, die üblicherweise DBMS mehrerer Hersteller unterstützt, ist dem Integrationsprozess ebenfalls nicht dienlich. Nicht alle DBMS bieten objektrelationale Erweiterungen an und die ORDBMS, die unterstützt werden, unterscheiden sich wiederum massiv in ihren SQL-Dialekten, was Mehraufwand bedeutet, wenn neue Features integriert werden sollen. Die Tatsache, dass Software-Produkte bereits als konform gepriesen werden, wenn lediglich ein Subset eines Standards erfüllt wird, ist bedauerlich. Die Abweichungen der SQL-Dialekte von Oracle, IBM DB2, IBM Informix und Postgres von Standard SQL:1999 sind gewaltig. Für die direkte Nutzung von XML bzw. GML für GIS-Anwendungen ist die Harmonisierung der SQL-Dialekte von Seiten der **ORDBMS-Hersteller** jedoch essentiell.

Die Erfahrungen der letzten Jahre haben eindeutig gezeigt, dass objektrelationale Technologie dann auf breite Akzeptanz stößt und genutzt wird, wenn sie in Form neuer Basisdatentypen gebündelt bzw. gekapselt wird (SDO_GEOMETRY, XMLTYPE). Die Verfolgung der im Rahmen dieser Arbeit erarbeiteten Konzepte zur künftigen Nutzung eines - wenn möglich standardisierten - Objekttyps für die datenzentrische Verwaltung von XML-Daten bedürfen keiner großen Anstrengungen von Seiten der GIS-Hersteller. Durch die native Unterstützung eines solchen Basisdatentyps in den Produkten der **GIS-Hersteller** ließe sich ein enormer Nutzen erzielen, zum einen

- durch die Öffnung der proprietären Implementierung der Datensatz-begleitenden Metadaten.

  zum anderen

- durch die Erweiterung der Geoobjekt-Modellierung um weitere Komponenten, insbesondere zur strukturierten Abbildung der fachlichen Bedeutung (Thematik, Funktion).

Die Steigerung der breiten Akzeptanz, XML als Speicherformat zur erweiterten, strukturierten Beschreibung von Geoobjekten in GIS einzusetzen, erfordert die Implementierung von leistungsstarken XML-Editoren (vgl. XMLSpy), z.B. zur benutzerfreundlichen Definition von XML-Schemata und Dokumenten für spezielle Anwendungen.

Im Hinblick auf die Verbesserung des offenen Zugriffs auf Vektordaten bietet sich GML als neutrales, standardisiertes Speicherformat an, da damit Transparenz sowohl für die GIS-Produkte der Hersteller als auch für die Datenbank-Hersteller zu erzielen ist. Gleichzeitig eröffnet sich dadurch die Chance, Interoperabilität auf Basis eines offiziellen Standards anstelle von Industriestandards zu gewährleisten. Weiterer Forschungsbedarf besteht demnach noch in der Optimierung von lesenden und schreibenden Zugriffen auf GML-Daten mittels GIS bzw. SQL (XQuery und SQL-Funktionen).

Literaturverzeichnis

[ABBEY et al., 2000] **Abbey, M., Corey, H., Abramson, I.**: *Oracle 8i für Einsteiger*, Carl Hanser Verlag, München Wien.

[ANDEREGG, 2002] **Anderegg, F.**: *Raumbezogene Datentypen in SQL/MM Spatial und verwandten Standards*, Informatik-Seminar, Hochschule für Technik Rapperswil.

[ANSI/ISO/IEC, 1999] **ANSI, ISO, IEC**: *ISO International Standard: Database Language SQL - Multimedia and Application Packages - Part 3: Spatial*, ANSI/ISO/IEC 13249-3, Dezember 1999.

[AMBLER, 2000] **Ambler, S. W.**: *Mapping objects to relational databases - what you need to know and why*, Paper unter http://www-4.ibm.com/software/developer/library/mapping_to_rdb/index.html.

[ATKINSON et al.,1989] **Atkinson, M., Bancilhon, F., DeWitt, D., Dittrich, K.R., Maier, D., Zdonik, S.**: *The Object-Oriented Database System Manifesto*, In Won Kim, J.-M. Nicolas and S. Nishio, Hrsg., Proc. of the 1st Int'l Conf. on Deductive and Object-Oriented Databases (DOOD), Seiten 40-57, Kyoto, Japan, Dezember 1989. Elsevier Science Publishers B.V..

[AUTODESK et al., 2003] **Autodesk, Intergraph, Laser-Scan, MapInfo**: *Open Interoperability With Oracle Spatial Technology*, White Paper, September 2003.

[BARTELME, 2000] **Bartelme, N.**: *Geoinformatik - Modelle, Strukturen, Funktionen*, Springer Verlag.

[BAUER, 2002] **Bauer, M.**: *Walddynamik nach Borkenkäferbefall in den Hochlagen des Bayerischen Waldes*, Dissertation, Lehrstuhl für Waldbau und Forsteinrichtung der TU München.

[BAYERISCHE STAATSREGIERUNG (1), 1999] **Bayerische Staatsregierung**: *High-Tech-Offensive Zukunft Bayern - Arbeits- und Lebensperspektiven für das 21. Jahrhundert*, Broschüre unter http://www.bayern.de.

[BAYERISCHE STAATSREGIERUNG (2), 1999] **Bayerische Staatsregierung**: *High-Tech-Offensive Zukunft Bayern - Stark in die Zukunft - Perspektiven für das 21. Jahrhundert*, Broschüre unter http://www.bayern.de.

[BISCHOF et al., 2002] **Bischof, I., Plabst, S.**: *Neue Medien in der Lehre*, Unveröffentlichtes Manuskript zum Treffen der PU-AG in Münster, 23.04.2002.

[BLANKENBACH, 2001] **Blankenbach, J.**: *Umsetzung und Grenzen der Interoperabilität zwischen vier ausgewählten GI-Systemen auf der Basis von Oracle 8i Spatial*, Diplomarbeit, Geodätisches Institut der TU Darmstadt.

[BOHN, 2003] **Bohn, C.**: *Ein Vergleich zwischen PostgreSQL und Oracle Database*, Seminararbeit am Lehrstuhl für Wirtschaftsinformatik III der Universität Mannheim.

[BOLLMANN et al., 2001] **Bollmann, J., Koch, W.G.**: *Lexikon der Kartographie und Geomatik, Band 1 und 2*, Spektrum Akademischer Verlag Heidelberg, Berlin.

[BFN, 2003] **Bundesamt für Naturschutz**: *Ausgewählte Arbeitsschwerpunkte des BfN*, http://www.bfn.de.

[BOURRET, 2003] **Bourret, R.**: *XML and Databases*, Paper unter http://www.rpbourret.com/xml/XMLAndDatabases.htm.

[BUHMANN et al., 2003] **Buhmann, E., Wiesel, J.**: *GIS-Report, Software Daten Firmen*, Bernhard Harzer Verlag Karlsruhe 2003.

[CATTELL, 1994] **Cattell, R. (Ed.)**: *The Object Database Standard: ODMG-93*, Morgan Kaufmann Publishers, San Mateo CA 1994, ISBN 1-55860-302-6.

[CHEN, 1976] **Chen, P.P.**: *The Entity-Relationship-Model - Towards a Unified View of Data*, ACM Transactions on Database Systems, Vol. 1, Nr. 1, P. 9-36.

[CODD, 1970] **Codd, E.F.**: *A relational model for large shared data banks*, Communications of the ACM, Vol. 13, P. 377-387.

[CODD, 1982] **Codd, E.F.**: *Relational Database: A practical foundation for productivity*, Communications of the ACM, Vol. 25, Nr. 2, P. 109-117.

[CODD, 1990] **Codd, E.F.**: *The relational model for database management: Version 2*, Addison-Wesley Publishing Company, Inc..

[CORREIA, 2003] **Correia, E.J.**: *ISO to SQL's Big Three: Get Together on XML - SQL 2003 to standardize XML interfaces, improve interoperability*, Paper unter http://www.sdtimes.com/news/088/story2.htm.

[CZAJA, 2003] **Czaja, J.**: *Abschlussbericht zum HTO-Teilprojekt 33-2: Sensorgestütztes mobiles GIS*, unveröffentlichter Bericht, Lehrstuhl für Geodäsie der TU München.

[DADAM, 2000] **Dadam, P.**: *Das Netzwerk-Datenmodell*, Vorlesungsmanuskript, Lehrstuhl für Informatik, Uni Ulm.

[DARWEN et al., 1995] **Darwen, H., Date, C.J.**: *The Third Manifesto (1995)*, ACM SIGMOD RECORD, Band 24, Nr. 1, S. 39-49, März 1995.

[DE LANGE, 2002] **de Lange, N.**: *Geoinformatik in Theorie und Praxis*, Springer Verlag, Januar 2002, ISBN 3-540-43286-8.

[DILLON (1), 2005] **Dillon, S.**: *Share you data as XML*, In: Oracle Magazine, January/February 2005, Januar 2005.

[DILLON (2), 2005] **Dillon, S.**: *Which Storage XML? - When to use CLOBs or objects to store your XML*, In: Oracle Magazine, March/April 2005, März 2005.

[DONAUBAUER, 2004] **Donaubauer, A.**: *Interoperable Nutzung verteilter Geodatenbanken mittels standardisierter Geo Web Services*, Dissertation, Fachgebiet Geoinformationssysteme der TU München.

[DURBEN, 2004] **Durben, R.**: *Inkrementelle Backups*, In: DOAG News Q1/2004, Januar 2004, Hrsg.: Deutsche ORACLE-Anwendergruppe e.V..

[EICHINGER, 2003] **Eichinger, A.**: *Entwicklung einer web-fähigen Geodatenbank für geowissenschaftliche Zwecke*, Diplomarbeit, Fachgebiet Geoinformationssysteme der TU München.

[ESRI, 1998] **Environmental Systems Research Institute, Inc.**: *ESRI Shapefile Technical Description*, ESRI White Paper, Juli 1998.

[ESRI, 2001] **Environmental Systems Research Institute, Inc.**: *ArcSDE Administration Guide*, ArcGIS Digital Books.

[ESRI (1), 2002] **Environmental Systems Research Institute, Inc.**: *Understanding ArcSDE*, ArcGIS Digital Books.

[ESRI (2), 2002] **Environmental Systems Research Institute, Inc.**: *Raster Data Management with ArcSDE*, In: ArcUser - The Magazine for ESRI Software Users, July - September 2002, http://www.esri.com/news/arcuser.

[ESRI (3), 2002] **Lyszkiewicz, M.**: *Expanding Data Sharing with Metadata Services*, In: ArcUser - The Magazine for ESRI Software Users, April - June 2002, http://www.esri.com/news/arcuser.

[ESRI (1), 2003] **Environmental Systems Research Institute, Inc.**: *ArcSDE Configuration and Tuning Guide for Oracle*, ArcGIS Digital Books.

[ESRI (2), 2003] **Environmental Systems Research Institute, Inc.**: *Building a Geodatabase*, ArcGIS Digital Books.

[ESRI (3), 2003] **Environmental Systems Research Institute, Inc.**: *Managing ArcSDE Services*, ArcGIS Digital Books.

[ESRI, 2004] **Environmental Systems Research Institute, Inc.**: *ArcGIS for Developers - A Complete System for Developing ArcObjects Anywhere*, http://www.esri.com.

[ESRI (1), 2005] **Environmental Systems Research Institute, Inc.**: *Produktspezifikationen*, http://www.esri.com.

[ESRI (2), 2005] **Environmental Systems Research Institute, Inc.**: *ESRI User Forum*, http://www.esri.com.

[ESRI (3), 2005] **Environmental Systems Research Institute, Inc.**: *Neuerungen in ArcView 9.0*, ArcGIS Digital Books.

[FOWLER et al., 2000] **Fowler, M., Scott K.**: *UML konzentriert - Eine strukturelle Einführung in die Standard-Objektmodellierungssprache*, Addison-Wesley-Verlag, 2000.

[FRANCICA, 2003] **Francica, J.**: *The Direction of Oracle's Spatial Strategy*, Artikel unter http://www.directionsmag.com.

[FRANCICA, 2004] **Francica, J.**: *Raster Data - Poor Second Cousin to Vector No More*, Artikel unter http://www.directionsmag.com.

[FRANZ, 1971] **Franz, F.**: *Volumenberechnung*, In: Kennel, E. 1973: Bayerische Waldinventur 1970/71, Inventurabschnitt I: Großrauminventur Aufnahme- und Auswertungsverfahren. Forstlicher Forschungsbericht München, Nr. 11, S. 98-100.

[GNÄGI et al., 2004] **Gnägi, H.R., Plabst S.**: *Modellbasierter Datenaustausch und Geowebservices im Internet - Zwei Wege zur Geodateninfrastruktur*, In: Tagungsband zum 9. Münchener Fortbildungsseminar Geoinformationssysteme, März 2004, TU München, Hrsg.: Schilcher, M..

[GRILL, 1990] **Grill, E.**: *Relationale Datenbanken, vom logischen Konzept zur physischen Realisierung*, AngewandteInformationsTechnik Verlags GmbH.

[HARTMANN, 2002] **Hartmann, J.**: *Umsetzung und prototypische Entwicklungen zur zukünftigen Führung des Liegenschaftskatasters (ALKIS$^{(R)}$)*, Dissertation, Geodätisches Institut der TU Darmstadt.

[HEUER, 1997] **Heuer, A.**: *Objektorientierte Datenbanken - Konzepte und Sprachen*, Addison-Wesley-Longman, 1997.

[HEURICH, 2004] **Heurich, M.**: *Zwischenbericht zum HTO-Teilprojekt 33-8: Entwicklung von innovativen Methoden zur Erfassung von Waldstrukturen im Nationalpark Bayerischer Wald*, unveröffentlichter Bericht, Nationalparkverwaltung Bayerischer Wald.

[HOCHHÄUSLER, 2001] **Hochhäusler, T.**: *Theoretische und praktische Ansätze zum Aufbau geographischer Datenbanken*, Diplomarbeit, Fachgebiet Geoinformationssysteme der TU München.

[HOHENSTEIN et al., 2002] **Hohenstein, U., Pleßer, V.**: *Oracle 9i*, dpunkt.verlag, Heidelberg 2002.

[HÖCK et al., 2001] **Höck, M., Manegold, J.**: *ArcMap, Programmierung mit VBA*, Eigenverlag: http://arcobjectsbuch.de.

[HÖRNER, 2001] **Hörner, C.**: *Datenbanken und Datenmodellierung: Semantische Datenbankmodelle*, Vorlesungsskript, Institut für Informatik der TU Clausthal.

[HUBER (1), 2002] **Huber, U.**: *Wissenschaftlich-Technische Dokumentation zum Referenz-GIS „Nationalpark Bayerischer Wald“*, Dokumentation zum Referenz-GIS „Nationalpark Bayerischer Wald“, 1324 Seiten, ISBN 3-935049-92-7.

[HUBER (2), 2002] **Huber, U.**: *Das Referenz-GIS „Nationalpark Bayerischer Wald“, eine fachübergreifende Forschungsplattform für die Geoinformatik*, Dissertation, Fachgebiet Geoinformationssysteme der TU München.

[IDC, 1999] **International Data Corporation**: *Worldwide Spatial Information Management Markets and Trends*, Report unter http://www.idc.com.

[INTERLIS, 2004] **Interlis**: *Produktspezifikationen*, http://www.interlis.ch.

[KIENER et al., 2002] **Kiener, H.**: *Standardbogen zu Natura 2000 Bayern - Gebietsbezogene Erhaltungsziele*, unveröffentlicht, Regierung von Niederbayern, Nationalparkverwaltung Bayerischer Wald, Bayerisches Landesamt für Umweltschutz.

[KLINE et al., 2001] **Kline, K., Kline, D.**: *SQL in a Nutshell*, Köln.

[KLÖCKING, 2004] **Klöcking, B.**: *Zwischenbericht zum HTO-Teilprojekt 33-7: Wasser- und Stoffhaushalt einer sich wandelnden Naturlandschaft im Nationalpark Bayerischer Wald*, unveröffentlichter Bericht, Bayerische Landesanstalt für Wald und Forstwirtschaft.

[KÖNIG-RIES, 2004] **König-Ries, B.**: *Datenbanksysteme*, Vorlesungsskript unveröffentlicht.

[KÜNG, 1994] **Küng, P.**: *O_2 und OBJECTSTORE vor OBJECTIVITY und ONTOS: Zehn objektorientierte DBMS im Vergleich*, Zeitschrift OUTPUT, Goldach im Juni 1994, S. 60-63.

[KVITKA, 2005] **Kvitka, C.**: *XQuery: A new way to search*, In: Oracle Magazine, January/February 2005, Januar 2005.

[LI et al., 2004] **Li, Y., Li, L.**: *Research on spatial database design and tuning based on Oracle and ArcSDE*, In: Proceedings zum 20. ISPRS-Kongress, Juli 2004, Istanbul.

[LONEY et al., 2003] **Loney, K., Koch, G.**: *Oracle 9i Die umfassende Referenz*, Carl Hanser Verlag, München Wien.

[LOOMIS, 1987] **Loomis, M.E.S.**: *The Database Book*, Macmillan Publishing Company, New York.

[LOPEZ, 2004] **Lopez, X.R.**: *Oracle Database 10G - A Spatial VLDB Case Study*, Paper unter http://otn.oracle.com.

[LOTHER, 2003] **Lother, G.**: *Konzeptionelle Aspekte eines landesweiten Fachgeoinformationssystems für die Bestandsdokumentation forstlicher Geodaten*, Dissertation, Fachgebiet Geoinformationssysteme der TU München.

[MAHR, 1999] **Mahr, B.**: *Probleme der Migration*, Unveröffentlichtes Manuskript, TU Berlin, 7 Seiten.

[MATTOS, 2000] **Mattos, N.M.**: *SQL99, SQL/MM and SQLJ: An Overview of the SQL Standards*, Tutorial, IBM Database Common Technology, 2000.

[MEIER et al., 2000] **Meier, A., Wüst, T.**: *Objektorientierte und objektrelationale Datenbanken*, dpunkt.verlag, Heidelberg.

[MELTON, 2002] **Melton, J.**: *Understanding Object-Relational and Other Advanced Features*, Morgan Kaufmann Publishers, San Francisco.

[MENG, 2001] **Meng, L.**: *Die Bandbreite kartographischer Darstellung*, In: Festschrift 50 Jahre Kartographie in München, FH München.

[MÜLLER, 2004] **Müller, T.**: *SQL-basierte Datenbankzugriffe und XML: Klassifizierung von Anwendungsprogrammen*, In: Tagungsband zum 16. Workshop „Grundlagen von Datenbanken“, Institut für Informatik, Heinrich-Heine-Universität Düsseldorf, Seiten 88-92, Monheim am Rhein, Juni 2004.

[NPV, 2001] **Nationalparkverwaltung Bayerischer Wald**: *Waldentwicklung im Bergwald nach Windwurf und Borkenkäferbefall*, Nationalpark Bayerischer Wald.

[NPV, 2003] **Nationalparkverwaltung Bayerischer Wald**: *Jahresbericht 2003*, Nationalpark Bayerischer Wald.

[NEUMANN, 1996] **Neumann, K.**: *Datenbanktechnik für Anwender*, Carl Hanser Verlag, München Wien.

[NEUMEIER, 2004] **Neumeier, S.**: *Abschlussbericht zum HTO-Teilprojekt 33-5: Konzeption und Entwicklung eines webbasierten touristischen Geoinformationssystems für die Nationalparkregion*, unveröffentlichter Bericht, Lehrstuhl für Bodenordnung und Landentwicklung der TU München.

[OLLE, 1978] **Olle, T.W.**: *The CODASYL Approach to Database Management*, J. Wiley & Sons, Chichester.

[OLLE, 1981] **Olle, T.W.**: *Das CODASYL-Datenbankmodell*, Springer Verlag.

[Open Geospatial Consortium, Inc., 1999] **Open Geospatial Consortium, Inc.**: *Open GIS Simple Features Specification for SQL, Revision 1.1 Mai 1999*, OpenGIS Project Document 99-049.

[ORACLE, 2001] **Tsai, J.**: *Oracle9i Database Product Family*, Oracle Technical White Paper, Oktober 2001.

[ORACLE (1), 2002] **Sharma, J.**: *Oracle Spatial, User's Guide and Reference, Release 9.2*, Oracle Documentation, März 2002.

[ORACLE (2), 2002] **Sharma, J.**: *Oracle Spatial*, Oracle Technical White Paper, Mai 2002.

[ORACLE (1), 2003] **Drake, M.**: *Oracle XMLDB*, Oracle White Paper, Januar 2003.

[ORACLE (2), 2003] **Jue, T.**: *Using OCCI: An Introduction to Objects*, Oracle White Paper, Mai 2003.

[ORACLE (3), 2003] **Lee, G.**: *Simple Strategies for Complex Data: Oracle9i Object-Relational Technology*, Oracle Technical White Paper, Oktober 2003.

[ORACLE (4), 2003] **Lee, G.**: *SQL 2003 Standard Support in Oracle Database 10g*, Oracle Technical White Paper, November 2003.

[ORACLE (5), 2003] **Farley, J.**: *Oracle Database 10g - Managing Geographic Raster Data Using GeoRaster*, Oracle Technical White Paper, Dezember 2003.

[ORACLE (6), 2003] **Oracle Inc.**: *Oracle Migration Workbench*, Oracle Technical White Paper, November 2003.

[ORACLE (7), 2003] **Geringer, D.**: *Oracle Spatial Best Practises*, Oracle Technical White Paper, Dezember 2003.

[ORACLE (1), 2004] **Qian, L.**: *Developing Spatial Applications Using Oracle Spatial and Map Viewer*, Oracle Technical White Paper, Februar 2004.

[ORACLE (2), 2004] **Godfrind, A.**: *Oracle Spatial 10g Update Seminar*, unveröffentlichter Foliensatz zum Oracle Spatial Seminar in München vom 5. April 2004.

[ORACLE (3), 2004] **Keh, A.**: *Using Oracle with Microsoft Active Directory*, Oracle Technical White Paper, September 2004.

[PISTOR, 2003] **Pistor, P.**: *Neuerungen in SQL2003*, Fachvortrag, Fachbereich Informatik der FH Heidelberg.

[PLABST, 2001] **Plabst, S.**: *Entwicklung eines objektrelationalen Datenmodells für ein kulturhistorisches Geoinformationssystem*, Diplomarbeit, Fachgebiet Geoinformationssysteme der TU München.

[PRIEBE, 2003] **Priebe, D.**: *Anwendung objektrelationaler DBMS - SQL:1999*, Fachvortrag, Lehrstuhl Datenbank- und Informationssysteme der Uni Rostock.

[QUEST, 2004] **Quest Software**: *Produktspezifikationen*, http://www.quest.com.

[SANDMANN, 2004] **Sandmann, M.**: *Objektrelationale Funktionalität der Oracle 9i Rel.2 DB*, Konzeptpapier, unveröffentlicht.

[SAAKE et al., 1997] **Saake, G., Türker, C., Schmitt I.**: *Objektdatenbanken*, Wiener Verlag, Himberg.

[SAUER, 1998] **Sauer, H.**: *Relationale Datenbanken - Theorie und Praxis*, Addison-Wesley-Longman, Bonn.

[SCHEUCH et al., 2004] **Scheuch, R., Lankes U.**: *Interview mit Andrew Mendelsohn, Oracle Corporation*, In: DOAG News Q1/2004, Januar 2004, Hrsg.: Deutsche ORACLE-Anwendergruppe e.V..

[SCHEUGENPFLUG, 1999] **Scheugenpflug, S.**: *Raum-Zeit-Analysen in Geoinformationssystemen am Beispiel des Referenz-GIS „Nationalpark Bayerischer Wald“*, Diplomarbeit, Fachgebiet Geoinformationssysteme der TU München.

[SCHEUGENPFLUG, 2002] **Scheugenpflug, S.**: *Forschung über Waldökosysteme*, In: Tagungsband zum 7. Münchener Fortbildungsseminar Geoinformationssysteme, März 2002, TU München, Hrsg.: Schilcher, M..

[SCHEUGENPFLUG, 2003] **Scheugenpflug, S.**: *GIS-Forschung über Waldökosysteme im Nationalpark Bayerischer Wald*, In: Tagungsband zur 23. Wissenschaftlich-Technischen Jahrestagung der DGPF, September 2003, Bochum, Hrsg.: Seyfert, E., ISSN 0942-2870.

[SCHEUGENPFLUG, 2004] **Scheugenpflug, S.**: *Abschlussbericht zum HTO-Teilprojekt 33-1 „Geodatenserver für forstwirtschaftliche und touristische Anwendungen“*, unveröffentlichter Bericht, Fachgebiet Geoinformationssysteme der TU München.

[SCHEUGENPFLUG et al., 2004] **Scheugenpflug, S.,Schilcher, M.**: *Object-relational features for modeling and analysis of spatio-temporal data*, In: Proceedings of 20th Congress of International Society for Photogrametry and Remote Sensing (ISPRS), Juli 2004, Istanbul.

[SCHILCHER et al., 1996] **Schilcher, M.**: *Geoinformationssysteme - Zwischenbilanz einer stürmischen Entwicklung*, In: Zeitschrift für Vermessungswesen (ZfV), 121. Jahrgang, Heft 8, S. 361-377.

[SCHILCHER et al., 2001] **Schilcher, M., Aumann, G.**: *Fortschritt in der GIS-Entwicklung durch mehr interdisziplinäre Zusammenarbeit*, In: 200 Jahre Vermessungsverwaltung - Es ist ein Maß in allen Dingen, Hrsg.: Bayerisches Staatsministerium der Finanzen, Abteilung Vermessung, Informations- und Kommunikationstechnik, München: Selbstverlag, 2001, ISBN 3-935612-01-X, S. 320-333.

[SCHILCHER, 2002] **Schilcher, M.**: *Geoinformatik I - Modellierung und Datenmodelle*, Vorlesungsskript unveröffentlicht.

[SCHILCHER, 2003] **Schilcher, M.**: *Geoinformatik I - Einführung in Geodatenbanken*, Vorlesungsskript, unveröffentlicht.

[SCHMIDHAUSER, 2003] **Schmidhauser, A.**: *SQL/XML*, Paper unter http://www.hta-be.bfh.ch/ schmd/db.

[SCHWINN et al., 2002] **Schwinn, U., Czarski, C.**: *Relational and beyond - Oracle9i, die native XML-Datenbank*, Paper unter http://www.doag.de/pub/docs/sig/database/2002-11/sigxml.PDF.

[SCHWINN, 2003] **Schwinn, U.**: *XML in der Oracle Datenbank - relational and beyond*, Paper unter http://www.btw2003.de/proceedings/paper/IP6.pdf.

[SEEMANN, 2003] **Seemann, M.**: *Native XML-Datenbanken im Praxiseinsatz*, Software & Support Verlag, 1. Auflage, 2003.

[SEIFERT, 2004] **Seifert, S.**: *Zwischenbericht zum HTO-Teilprojekt 33-3: Dynamisches Informationssystem für Naturwälder*, unveröffentlichter Bericht, Lehrstuhl für Waldwachstumskunde der TU München.

[SPIELVOGEL, 2004] **Spielvogel, S.**: *Zwischenbericht zum HTO-Teilprojekt 33-6: Umsetzung und Speicherung von Kohlenstoff in Waldböden des Nationalpark Bayerischer Wald*, unveröffentlichter Bericht, Lehrstuhl für Bodenkunde der TU München.

[STAKEN, 2001] **Staken, K.**: *Introduction to Native XML Databases*, Paper unter http://www.xml.com/pub/a/2001/10/31/nativexmldb.html.

[STEINBRECHER, 2004] **Steinbrecher, R.**: *Zwischenbericht zum HTO-Teilprojekt 33-4: Entwicklung eines mobilen Sensors zur punktgenauen Früherkennung von Borkenkäferbefall (Künstliche Nase)*, unveröffentlichter Bericht, Institut für Atmosphärische Umweltforschung.

[STONEBRAKER et al., 1990] **Stonebraker, M., Rowe, L., Lindsay, B., Gray, J., Carey, M., Brodie, M., Berstein, P., Beech, D.**: *Third-Generation Database System Manifesto*, SIGMOD Record, Vol.19, No.3, September 1990, P. 31-44.

[STONEBRAKER et al., 1999] **Stonebraker, M., Moore, D.**: *Die nächste große Welle*, Carl Hanser Verlag, München Wien.

[TÜRKER, 2003] **Türker, C.**: *SQL:1999 & SQL:2003*, dpunkt.verlag, Heidelberg, 1. Auflage, ISBN 3-89864-219-4.

[TU Berlin, 2004] **TU Berlin**: *Relationale Anfragesprachen*, Skriptum zur Lehre, Computergestützte InformationsSysteme (CIS), TU Berlin unter http://cis.cs.tu-berlin.de/Lehre/WS-0304/Sonstiges/db-pages/Inhalt/rel_db_sprachen/Folien/index.html.

[VETTER, 2003] **Vetter, S.**: *Zentrale Benutzerverwaltung von Oracle*, Unveröffentlichtes Manuskript zur Schulung und Beratung der Fa. Trivadis, 2003.

[WERNER, 2003] **Werner, C.-D.**: *ArcGIS - GIS-Plattform mit Perspektive*, 8. Internationales Anwenderforum für Geoinformationssysteme, Februar 2003, Duisburg.

[ZEILER, 1999] **Zeiler, M.**: *Modeling Our World*, ESRI Press Series.

[ZIEGLER, 2001] **Ziegler, M.**: *Untersuchung geographischer Anfragesprachen auf der Basis relationaler und objektrelationaler Datenbankmanagementsysteme*, Dissertation, Fachgebiet Geoinformationssysteme der TU München.

Abkürzungsverzeichnis

ADT	Abstract Data Type
ANSI	American National Standards Institute
API	Application Programming Interface
BayStMLF	Bayerisches StaatsMinisterium für Landwirtschaft und Forsten
BCNF	Boyce-Codd-Normalform
BLOB	Binary Large Object
CASE	Computer Aided Software Engineering
CODASYL	Conference on Data Systems Languages
DB	DatenBank
DBMS	DatenBankManagementSystem
DBS	DatenBankSystem
DCL	Data Control Language
DDL	Data Definition Language
DFK	Digitale FlurKarte
DFN	Deutsches ForschungsNetz
DML	Data Manipulation Language
DOM	Document Object Model
DQL	Data Query Language
DTD	Data-Type-Definition
ESRI	Environmental Systems Research Institute
FFH	Fauna Flora Habitat
GDI	GeoDatenInfrastruktur
GDS	GeoDatenServer
GeoDBMS	GeoDatenBankManagementSystem
GUI	Graphical User Interface
IMS	Internet Map Server
ISO	International Standards Organization
LBS	Location Based Services
LDAP	Lightweight Directory Access Protocol
LRZ	Leibnitz RechenZentrum
MBR	Minimum Bounding Rectangle
NF^2	Non First Normal Form
NPV	NationalParkVerwaltung Bayerischer Wald
ODBC	Open DataBase Connectivity
ODMG	Object Database Management Group

OGC	Open Geospatial Consortium
ODBMS	ObjektDatenBankManagementSystem
OID	Object Identifier
OLAP	OnLine Analytical Processing
OO	ObjektOrientiert
OODBMS	ObjektOrientiertes DatenBankManagementSystem
OR	ObjektRelational
ORDBMS	ObjektRelationales DatenBankManagementSystem
ORDBS	ObjektRelationales DatenBankSystem
OQL	Object Query Language
OWS	OGC Web-Service
PNF	Partitioned Normal Form
POI	Point Of Interest
RDBMS	Relationales DatenBankManagementSystem
RDBS	Relationales DatenBankSystem
SERM	Strukturiertes Entity Realtionship Modell
SFSQL	Simple Features Specification for SQL
SQL	Structured Query Language
STMLF	Staatsministerium für Landwirtschaft und Forsten
UDT	User Defined Type
UML	Unified Modelling Language
W3C	World Wide Web Consortium
WFS	WEB Feature Specification des Open Geospatial Consortium
WMS	WEB Mapping Specification des Open Geospatial Consortium
XML	eXtensible Markup Language
1NF	Erste Normalform
2NF	Zweite Normalform
3NF	Dritte Normalform

Abbildungsverzeichnis

Tabellenverzeichnis

Anhang A

Glossar

Abstract Data Type (ADT)	ADT erlauben neben der Neubenennung oder der Kombination von Basisdatentypen auch die Definition von Methoden und die Kapselung der internen Darstellung der Werte im Sinne der Objektorientierung. Der Zugriff auf die Attribute eines so erzeugten Objekts erfolgt dann ausschließlich über Methoden.
Binary Large Object (BLOB)	Datentyp zur Speicherung von bis zu 4GB großen Binärdaten in der Datenbank. Ein BLOB hat keine Struktur, die vom DBMS (via SQL) interpretiert werden kann. Oracle speichert LOBs physisch nicht in der Tabelle, in der sie als Datentyp definiert werden, sondern ausserhalb in einem separaten Bereich, den die Datenbank speziell für LOB-Daten angelegt hat. Der Verweis erfolgt mittels Locator-Werte, die auf den Standort der Daten zeigen. Durch die Auslagerung der Daten vermeidet die Datenbank, dass bei jedem Lesen mehrerer Zeilen in der Datenbank auch die LOB-Daten gelesen werden müssen. Stattdessen werden die LOB-Daten nur bei Bedarf gelesen. Für den Bereich, in dem die LOB-Daten hinterlegt werden, können geeignete Speicherparameter (Tablespace und Größe) definiert werden (Tuning). Grundsätzlich verweist Binär auf das binäre Zahlensystem, in dem man nur mit Nullen und Einsen rechnet. In der Praxis bedeutet binär, dass eine Datei aus einer Folge von Nullen und Einsen zusammengesetzt ist und diese Datei ausschließlich von bestimmten Applikationen interpretiert werden kann.
Client / Server Architektur	Als Client / Server Architekturen werden Soft- und Hardwarekonfigurationen bezeichnet, in denen Dienste auf einem Rechner (Server) von Prozessen auf einem anderen Rechner (Client) in Anspruch genommen werden. Um eine Client / Server Architektur realisieren zu können, müssen Diensteschnittstellen vorhanden sein, die den Aufruf eines Dienstes auf einem anderen Rechner erlauben. Diese Schnittstellen können proprietär oder standardisiert sein. Um Systeme unterschiedlicher Hersteller verbinden zu können, bieten neuere Systeme meist eine oder mehrere Standardschnittstellen an.
Data Mining	Unter Data Mining versteht man das systematische (in der Regel automatisierte oder halbautomatische) Entdecken und Extrahieren unbekannter Informationen aus großen Mengen von Daten.

GeoDBMS	Ein GeoDBMS (Geo-Datenbank-Management-System) ist ein vollständiges DBMS mit zusätzlichen Möglichkeiten zur Repräsentation, Abfrage und Manipulation von Objekten mit Raumbezug auf der Erde. Das GeoDBMS hat Aufgaben eines DBMS zu erfüllen. Darüber hinaus unterstützt es räumliche Datentypen und Operationen auf diesen Datentypen.
Lightweight Directory Access Protocoll (LDAP)	Protokollstandard zur Realisierung von Verzeichnisdiensten zur Zentralisierung redundant vorhandener Daten, was in einer Vereinfachung der Verwaltung der IT-Infrastruktur resultiert. Meist werden im Verzeichnisdienst Daten rund um den Benutzerzugang abgelegt.
LZW	Das im Bereich der Computergraphik am weitesten verbreitete Kompressionverfahren ist die sogenannte „Lempel-Ziv-Welch“ (LZW)-Kompression. Es ist dies eine verlustfreie Datenkompressionsmethode, die in den verschiedensten Grafikformaten, wie GIF und TIFF Verwendung findet.
OGC	Open Geospatial Consortium, Standardisierungsgremium mit Sitz in den USA. Entwickelt de facto Standards für die Interoperabilität von Geoinformationssystemen. Zusammenarbeit mit ISO TC211 und weiteren Normungs- und Standardisierungsgremien.
OLAP	Online Analytical Processing (OLAP) bezeichnet ein Datenhaltungskonzept, welches komplexe Geschäftsanalysen ermöglicht, die vom Endanwender in einer mehrdimensionalen Umgebung durch Werkzeug- und IT-Unterstützung vorgenommen werden können. Hierbei werden Daten - relationale oder „flache“ eindimensionale - in einem für die Analyse optimierten Cube (Würfel) gespeichert, indem sie entlang von Dimensionen oder Achsen für die unternehmensrelevante Größen gespeichert werden.
Pixeltiefe	Wird auch als Farbtiefe bezeichnet. Die Pixeltiefe legt die Anzahl der Bits an Farbinformation pro Bildpunkt fest, d.h. die Pixeltiefe definiert die Maximalanzahl von Farben, die digitale Geräte darstellen können. Eine Farbtiefe von 24 Bit („True Color“-Qualität) entspricht z.B. 16,7 Mio. möglichen Farben.
Quad-Tree	Der Quad-Tree gehört zu den bekanntesten geometrischen Zugriffsstrukturen. Bei Quad-Trees wird von einem quadratischen Untersuchungsraum ausgegangen, der in vier gleiche Teile unterteilt wird. Jeder der vier Quadranten wird in einem iterativen Prozess in weitere Sub-Quadranten aufgeteilt. Je höher die Informationsdichte bzw. je mehr räumliche Objekte in einem bestimmten Quadranten vorzufinden sind, desto öfter wird der betreffende Bereich unterteilt. Die Anzahl der Zellteilungen ist somit eine Funktion der Datendichte. Die Zuordnung von Objekten zu Quadranten lässt auf die grobe Objektposition schließen. Diese Einteilung kann allerdings nur bei punktförmigen Objekten überschneidungsfrei bzw. eindeutig erfolgen. Linien- oder flächenhafte Elemente müssen bei einem statischen Raster dieser Art in der Regel mehreren Quadranten zugeordnet werden.
Transitive Abhängigkeit	Seien X, Y und Z Attribute. Ist Y von X funktional abhängig und Z von Y, so ist Z von X funktional abhängig. Diese Abhängigkeit ist transitiv.

WFS	Ein OGC Web Feature Service ist ein Geodienst, der es einer Client-Software erlaubt, Features in einer Geodatenbank abhängig von ihren Eigenschaften zu selektieren und zum Client zu übertragen. Die Kommunikation zwischen Feature Service und der Client-Software basiert auf dem HTTP-Protokoll.
WMS	Ein OGC Web Map Service ist ein Geodienst, der es einer Client-Software erlaubt, mittels vom OGC spezifizierter Operationen folgende Informationen abzurufen: dienstinstanzbezogene Metadaten (Capabilities), digitale Karten und optional Informationen zu Features, die auf der Karte dargestellt sind. Die Kommunikation zwischen Map Service und der Client-Software basiert auf dem HTTP-Protokoll.
Worldfile	Ein Worldfile, häufig auch als Steuerdatei bezeichnet, wird von GIS genutzt, um den Raumbezug von Rasterdaten herzustellen. Sie wird vom GIS-System zum entsprechenden Image interpretiert. Beim TIFF-Dateiformat gibt es die Option, den Koordinatenbezug direkt in der Imagedatei abzuspeichern (GeoTIFF), somit entfällt die zusätzliche Steuerdatei.

Anhang B

Vollständige Abbildung des GIS-Systemkonzepts

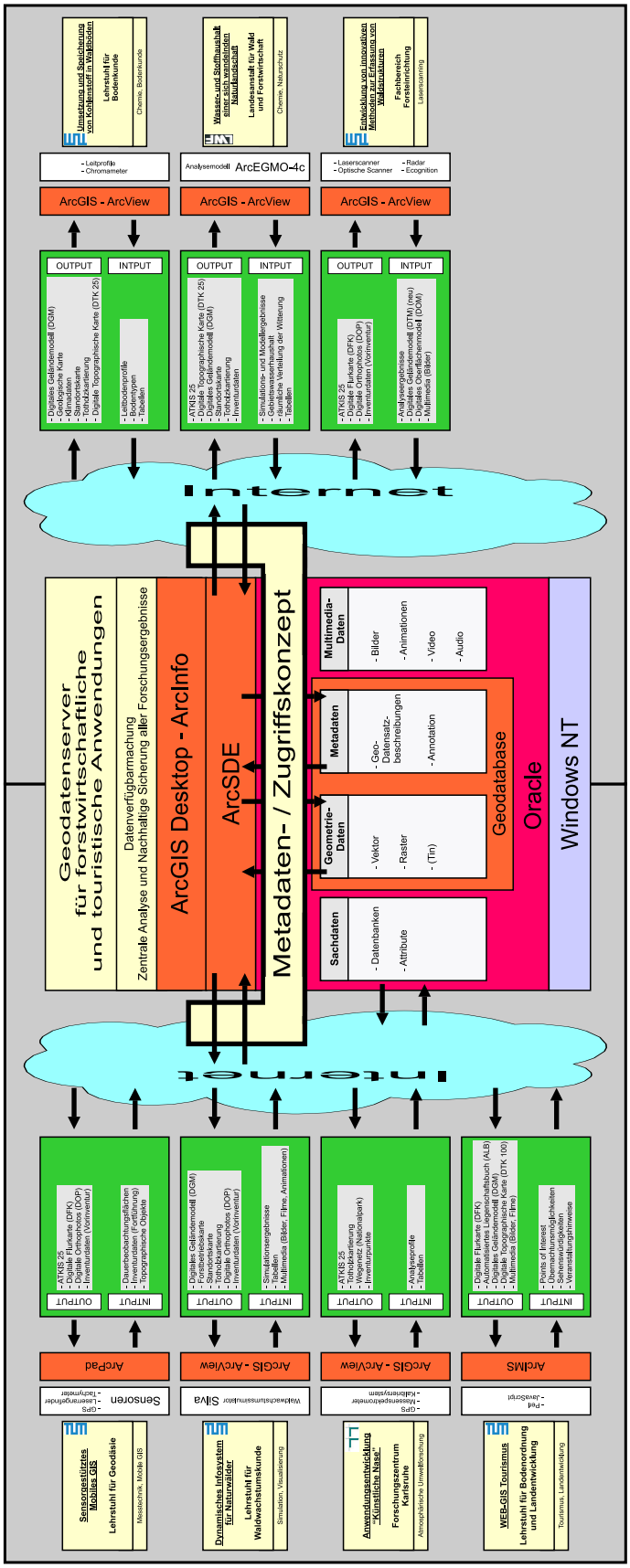

Abb. B.1: GIS-Systemkonzept zur Verknüpfung interdisziplinärer Anwendungen

Anhang C

Forstliches Sachdatenmodell nach dem Redesign

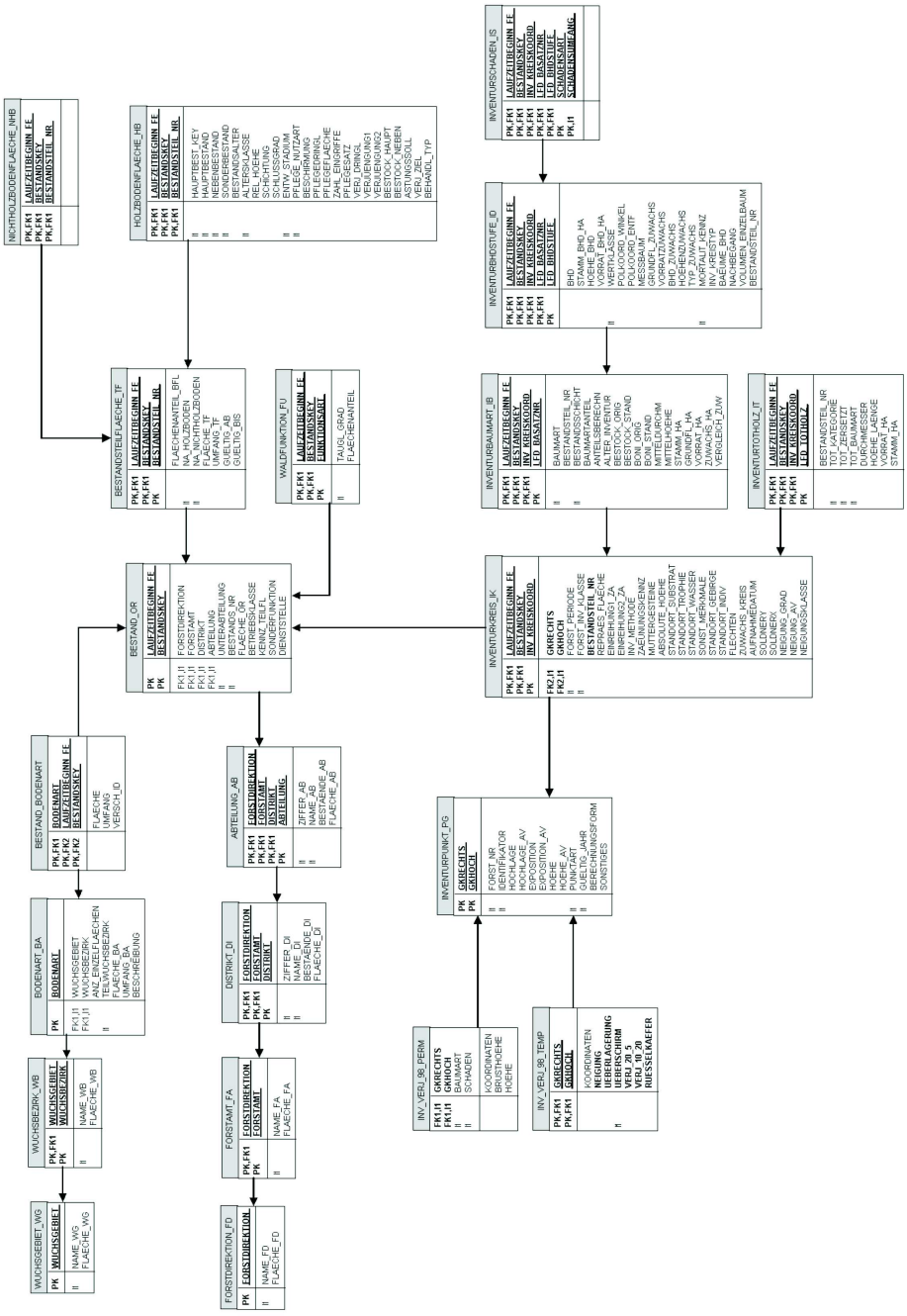

Abb. C.1: Logisches Datenbankmodell der forstlichen Sachdaten

Anhang D

UML-Klassendiagramm des logischen Metadatenmodells

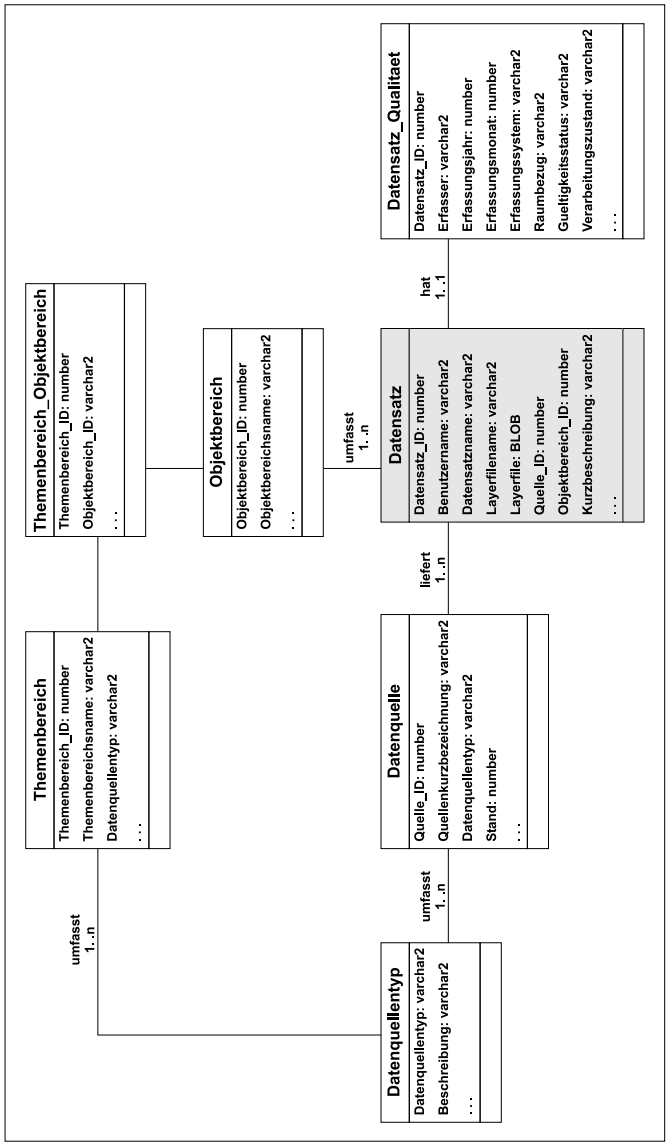

Abb. D.1: UML-Klassendiagramm des für externe Zugriffe relevanten Teils des logischen Metadatenmodells

Anhang E

SQL-Syntax zur Analyse der Totholzentwicklung

– Vorselektion durch Definition eines 200 m-Buffers

```
create view buffer_ergebnis1999 (GEOM) as select SDO_GEOM.SDO_BUFFER
(c.geom, m.diminfo, 200) from totholz1998 c, user_sdo_geom_metadata m
where m.table_name='TOTHOLZ1998' and m.column_name='GEOM';
```

– Räumliche Selektion und Eliminierung von Duplikaten

```
create view endergebnis_selektion1999 (GEOM, id) as select a.geom, a.id
from totholz1999 a where a.id not in (select distinct a.id from totholz1999 a,
BUFFER_ERGEBNIS1999 b where SDO_RELATE (a.GEOM, b.GEOM, 'mask=ANYINTERACT
querytype=JOIN') = 'TRUE');
```

– Sequentielle Verschneidung mit Einflussfaktoren mit jeweils Beschränkung auf Tupel mit Schnittgeometrien
– Verschneidung mit Feuchtigkeitsklassen

```
create view ergebnis_feucht_intersect1999 (id_selek1, id_feucht, text_feucht, GEOM)
as select s.id, d.id, d.text, SDO_GEOM.SDO_INTERSECTION (d.GEOM, m.DIMINFO, s.GEOM,
m.DIMINFO) from FEUCHTIGKEIT d, ENDERGEBNIS_SELEKTION1999 s, user_sdo_geom_metadata m
where m.table_name='FEUCHTIGKEIT' and m.column_name='GEOM';

create view ende_feucht_intersect1999 (id_selek1, id_feucht, text_feucht, GEOM)
as select p.id_selek1, p.id_feucht, p.text_feucht, p.GEOM
from ergebnis_feucht_intersect1999 p where p.GEOM is not null;
```

– Verschneidung mit Exposition

```
create view ergebnis_expo_intersect1999 (id_selek1, id_feucht, text_feucht, id_expo,
text_expo, GEOM) as select t.id_selek1, t.id_feucht, t.text_feucht, h.id, h.text,
SDO_GEOM.SDO_INTERSECTION (h.GEOM, m.DIMINFO, t.GEOM, m.DIMINFO) from EXPOSITION h,
ENDE_FEUCHT_INTERSECT1999 t, user_sdo_geom_metadata m where m.table_name='EXPOSITION'
and m.column_name='GEOM';

create view ende_expo_intersect1999 (id_selek1, id_feucht, text_feucht, id_expo, text_expo,
```

```
GEOM) as select z.id_selek1, z.id_feucht, z.text_feucht, z.id_expo, z.text_expo, z.GEOM
from ergebnis_expo_intersect1999 z where z.GEOM is not null;
```

– Verschneidung mit Bestandsinformation

```
create view ergebnis_best_intersect1999 (id_best, id_selek1, id_feucht, text_feucht,
id_expo, text_expo, bestkey, nam, habestkey, habestform, bestandsal, GEOM) as select
k.id, f.id_selek1, f.id_feucht, f.text_feucht, f.id_expo, f.text_expo, k.bestkey, k.nam,
k.habestkey, k.habestform, k.bestandsal, SDO_GEOM.SDO_INTERSECTION (k.GEOM, m.DIMINFO,
f.GEOM, m.DIMINFO) from BESTAENDE k, ENDE_EXPO_INTERSECT1999 f, user_sdo_geom_metadata
m
where m.table_name='BESTAENDE' and m.column_name='GEOM';

create view ende_best_intersect1999 (id_best, id_selek1, id_feucht, text_feucht, id_expo,
text_expo, bestkey, nam, habestkey, habestform, bestandsal, GEOM) as select y.id_best,
y.id_selek1, y.id_feucht, y.text_feucht, y.id_expo, y.text_expo, y.bestkey, y.nam,
y.habestkey, y.habestform, y.bestandsal, y.GEOM from ergebnis_best_intersect1999 y
where y.GEOM is not null;
```

– Übertragung in Result Table und Registrierung:

```
create table ende1999 as select * from ende_best_intersect1999;

delete from USER_SDO_GEOM_METADATA
where table_name = 'ENDE1999' and column_name = 'GEOM';

insert into USER_SDO_GEOM_METADATA (table_name, column_name, diminfo, srid)
       values ('ENDE1999', 'GEOM',
       MDSYS.SDO_DIM_ARRAY
       (MDSYS.SDO_DIM_ELEMENT('X', 4598321.000000000, 4615963.500000000, 0.000000050),
       MDSYS.SDO_DIM_ELEMENT('Y', 5416977.500000000, 5429338.122558350, 0.000000050)
       ),
       82032);
```

– Transaktion abschließen:

```
commit;
```

Anhang F

Lebenslauf

Persönliche Daten

Name:	Stefan Johannes Scheugenpflug
Geburtsdatum:	02.04.1973
Geburtsort:	Regensburg
Familienstand:	ledig
Staatsangehörigkeit:	deutsch

Ausbildung und berufliche Tätigkeit

1999 – 2005	Wissenschaftlicher Angestellter am Fachgebiet Geoinformationssysteme der Technischen Universität München
1998	Diplomarbeit am Fachgebiet Geoinformationssysteme der Technischen Universität München
	Raum-Zeit-Analysen in Geoinformationssystemen am Beispiel des Referenz-GIS „Nationalpark Bayerischer Wald“
1993 – 1999	Studium des Vermessungswesens an der Technischen Universität München
1992 – 1993	Grundwehrdienst
1983 – 1992	Albrecht-Altdorfer-Gymnasium, Regensburg
1979 – 1983	Grundschule Viehhausen, Landkreis Regensburg