**FACHHOCHSCHULE KÖLN,
UNIVERSITY OF APPLIED
SCIENES COLOGNE
ABTEILUNG GUMMERSBACH**

FACHBEREICH INFORMATIK

Diplomarbeit

GEOGRAFISCHE INFORMATIONSSYSTEME UNTER BERÜCKSICHTIGUNG DER ORACLE SPATIAL TECHNIK

von

Reinhard S. Zöllner

Neusser Landstr. 379

50769 Köln

zur Erlangung des akademischen Grades

Diplom - Informatiker (FH)

1. Prüfer: Frau Professor Dr. Heide Faeskorn-Woyke

2. Prüfer: Herr Professor Dr.-Ing. Jackson Roehrig

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

API	Application Programming Interface
EJB	Enterprise Java Beans
GIS	Geografische Informationssysteme
HTML	Hyper Text Markup Language
JDBC	Java Database Connectivity
JMS	Java Message Service
JNDI	Java Naming and Directory Interface
JSP	Java Server Pages
JTA	Java Transaction API
J2EE	Java 2 Enterprise Edition
ODBC	Open Database Connectivity
OGC	Open GIS Consortium
RMI	Remote Method Invocation
SQL	Structured Query Language

K a p i t e l 1

Das Projekt Wassermanagement in Chile

Das Projekt, das hier kurz mit “Wassermanagement in Chile“ überschreiben ist, hat sich zum Ziel gesetzt, ein Monitoring- und Informationssystem für ein effizientes Wassermanagement in Flusseinzugsgebieten zu entwickeln. Dies soll dann erstmalig am Beispiel des Rio Aconcagua in Chile eingesetzt werden. Die Projektleitung liegt in Händen von Herrn Professor Dr.-Ing. J. Roehrig vom Institut für Tropentechnologie der Fachhochschule Köln.
Für dieses Projekt konnten bereits im Vorfeld viele Partner gewonnen werden, die hier kurz aufgeführt werden :

?? Prof. Faeskorn-Woyke, Fachbereich Informatik, FH Köln
?? Prof. Sturm, Fachbereich Ver- und Entsorgungstechnik, FH Köln
?? Prof. Menz, Universität Bonn, Zentrum für Fernerkundung der Landoberfläche
?? Dr. Ole Larsen, Firma Hydrowelten
?? Dipl. Ing. Schneider , Firma UIT
?? Dr.-Ing. G. Kolisch , Wupperverband
?? Prof. Carlos Espinoza, Universidad de Chile
?? Prof. Eduardo Salgado, Universidad Católica de Valparaiso
?? Monica Pardo, Direccion General de Agua, Santiago de Chile
?? Christian Neumann, Direccion General de Agua, V Region
?? Ing. Marco Montt, Confederación del Rio Aconcagua.

Die genaue Problemstellung kann in der Projektbeschreibung von Prof. Dr.-Ing. J. Roehrig nachgelesen werden.
Hier wird sich aus der Sicht der Informatik der Problemstellung genähert. Dies bedeutet im einzelnen, dass erst einmal die Grundlagen von GIS-Systemen vorgestellt werden, da dies ein Gebiet ist, dass im Informatikstudium normalerweise nicht behandelt wird.

Danach wird in einem Kapitel über Datenbanksysteme eine kurze Betrachtung verschiedener Datenbanksysteme folgen. In diesem Kapitel wird auch ein Datenmodell erarbeitet, das für das Projekt eingesetzt werden kann. Des weiteren wird ein Auszug aus diesem Datenmodell umgesetzt, so dass dieses dann beispielhaft angewendet werden kann. Das Kapitel schließt mit Grundlagen zur Oracle Spatial Technologie ab, wobei hier der Bogen von den Grundlagen der GIS-Systeme zur Oracle Spatial Technologie geschlagen wird.

Im vierten Kapitel wird gezeigt, wie mit Java auf Datenbanken zugegriffen werden kann. Das Datenmodell aus Kapitel 3 wird mit einer Standard – Java Benutzeroberfläche versehen.

Im fünften Kapitel wird dann das GIS-System ArcInfo kurz betrachtet und gezeigt wie es möglich ist, mittels ArcSDE auf einer Oracle Datenbank verschiedene Aktionen auszuführen. Weitere Informationen zur Installation sind im Anhang zu finden.

K a p i t e l 2

Geografische Informationssysteme

Geografische Informationssysteme, kurz als GIS-Systeme bezeichnet, werden heute in vielen verschiedenen Zusammenhängen und mit den verschiedensten anderen Technologien eingesetzt. Allen zugrunde liegt aber, dass die Daten in digitaler Form gespeichert werden. Erste Verwendung fanden GIS-Systeme bei Landkarten auf CD. Durch die weite Verbreitung des Internet und die dadurch bedingten neuen Anwendungen wurden hier auch GIS-Systeme eingesetzt. Dabei sind z.B. Stadtpläne zu nennen, aber auch bei der Erstellung von Wetterinformationen für verschiedenste Nutzer werden heutzutage GIS-Systeme eingesetzt. Auch Bauplanungen von größerem Ausmaß, wie z.B. Strassen und Landschaftsbau, sind die klassischen Einsatzgebiete von GIS-Systemen. Durch die Entwicklung von Navigationssystemen für Autos haben selbst in diesem Bereich GIS-Systeme neue Einsatzbereiche gefunden. In diesem Kapitel wird kurz auf die Entwicklung von GIS-Systemen eingegangen, um danach einen Blick auf die verwendeten Datenstrukturen und die verwendeten Algorithmen (eine Grundlage für den Einsatz von GIS-Systemen) zu legen.

2.1 Entstehung der GIS-Systeme

Karten, Pläne und Informationen über seine nähere und weitere Umgebung war und ist für den Menschen von großer Bedeutung. Am Anfang um Wege zu bestimmten Orten zu finden, später um sich auf Kontinenten zurecht zu finden. Heute werden auch wieder die Informationen im kleinen benötigt, wie z.B. wem gehört ein Grundstück, welche Leitungen für Gas, Wasser und Strom sind wo verlegt ? Aber auch Informationen über Städte, Strassen, Flüsse und Länder müssen heute in den verschiedensten Zusammenhängen abrufbar und verwendbar sein. Diese große Bandbreite müssen heutzutage GIS-Systeme abdecken. Um dies zu erreichen entwickelten sich die GIS-Systeme über einen längeren Zeitraum, wobei hier der Zeitraum betrachtet werden soll, in dem sich

die Entwicklung mit dem Einsatz von EDV-Systemen völlig neuen Einsatzgebieten öffnete. Bartelme unterteilt diese Entwicklung in seinem Buch [Bar 95] in 5 Phasen. Diese 5 Phasen sind in Abbildung 2.1 auf einer Zeitleiste dargestellt. Hier sind die Phasen aufgeführt die Barteleme benennt. Dem gegenüber sind einige Zeitpunkte von wichtigen Entwicklungsschritten von Hard- und Software dargestellt, so dass man eine Gefühl für die entsprechenden Entwicklungen erhält.

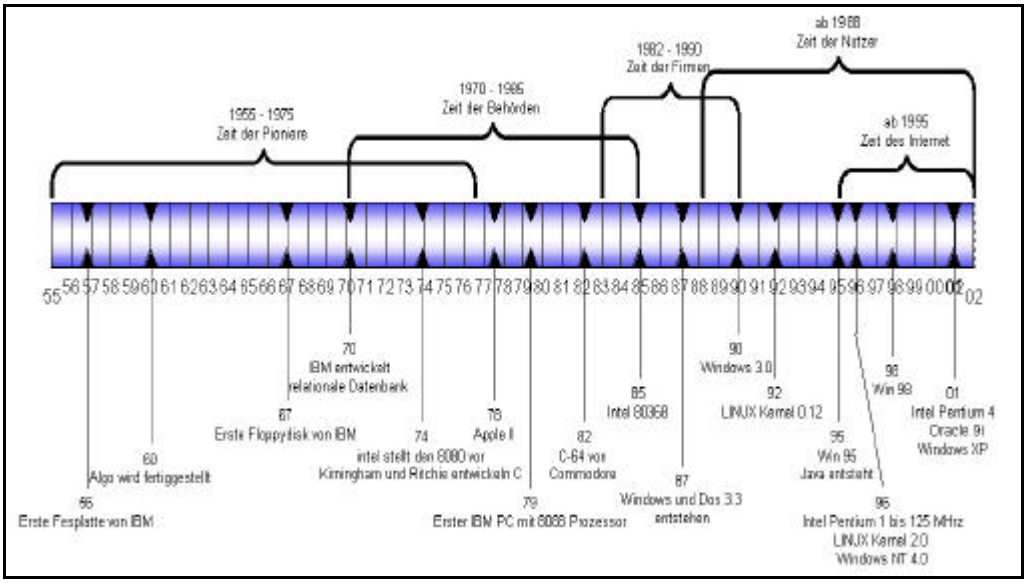


Abbildung 2.1 Zeitleiste

Was charakterisierte die einzelnen Phasen in der Entwicklung der GIS-Systeme ?
Die Zeit der Pioniere 1955 bis 1970 war dadurch gekennzeichnet, dass verschiedenste Pioniere individuelle und voneinander isolierte Wege einschlugen. Daten lagen nicht in digitalisierter Form vor und die Entwicklung der Hardware befand sich noch in den Grundlagen.
In der Zeit von 1970 bis 1985 begannen die ersten Behörden die Verwaltung von Geodaten auf Computer umzustellen. Von der Funktionalität heutiger Geoinformationssysteme waren diese ersten Umsetzungen noch weit entfernt, aber die ersten entscheidenden Grundlagen wurden gelegt.
Die Zeit der Firmen, von 1982 bis 1990, ist gekennzeichnet durch entsprechende Entwicklungen im Software und Hardwarebereich, wie auch aus der Zeitleiste zu ersehen ist. In dieser Zeit wurden grundlegende Entwicklungen getätigt, die bis heute in den GIS-Markt hereinreichen. Firmen wie ESRI entwickelten Arc/Info, Siemens entwickelte Sicad. Die Entwickler auf der Hardwareseite stellten

Workstations zur Verfügung, mit denen es möglich wurde die Datenfülle und die Grafikanwendungen zu bewältigen.
Dann ab 1988 kann man von der Zeit der Nutzer sprechen. Die Software stand zur Verfügung, die Hardware hatte den Stand erreicht, so dass ein sinnvoller Einsatz gewährleistet war. In Netzwerken war es jetzt möglich Daten und Funktionen sinnvoll zu nutzen.
Ab 1995 ist das Zeitalter des Internet angebrochen. Viele neue Anwendungsbereiche entstehen. Daten werden weltweit genutzt. Die Hersteller müssen neue Lösungen entwickeln, denn auch Laien können nun in verschiedensten Anwendungen GIS-Systeme nutzen. Diese neuen Anwendungen sind z.B. Routenplaner, digitale Stadtpläne im Internet oder Navigationssysteme in Autos. Gleichzeitig werden GIS-Systeme aber auch von immer höhere Anzahl verschiedenen Fachrichtungen genutzt, so z.B. von Bauingenieuren zur Straßenplanung oder von Geowissenschaftlern um Daten über Flusssysteme zu analysieren. Aber auch weiterhin setzen Behörden GIS-Systeme ein, um die anfallenden Daten zu speichern.

2.2 Die Elemente eines GIS-Systems

Ein GIS-System setzt sich aus verschiedenen Komponenten zusammen. Betrachtet man dazu erst einmal die Definition zu einem GIS-System. [Bil 99/1]

Definition : Geo-Informationssysteme
„ Ein Geo-Informationssystem ist ein rechnergestütztes System, das aus Hardware, Software, Daten und den Anwendungen besteht. Mit ihm können raumbezogene Daten digital erfasst und redigiert, gespeichert und reorganisiert, modelliert und analysiert sowie alphanumerisch und graphisch präsentiert werden.“

Anhand dieser Definition ist zu erkennen, dass ein Anwender durch Einsatz von Software, Hardware und Daten neue Informationen aus bestehenden Daten gewinnen kann. Diese neu gewonnenen Informationen werden mit Hilfe des gleichen Systems neu aufbereitet und können analysiert werden. Zum besseren Verständnis von beteiligten Personen werden die Daten in verschiedenster Weise

präsentiert. Dieses stellt auch direkt die Funktionalität des Modell – View – Controller Musters dar (siehe auch [BMR 98]). Das Modell repräsentiert die zur Verfügung stehenden Daten. Der Controller der die Informationen neu zusammensetzt und schließlich der View der die Ergebnisse präsentiert (siehe Abbildung 2.2) zeigt, den engen Zusammenhang von Modell – View – Controller Muster, mit dem Aufbau von Geo-Informationssystemen.

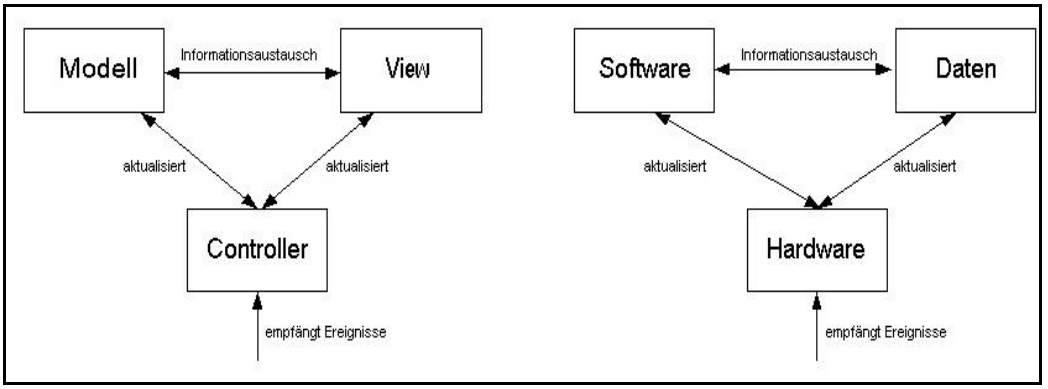


Abbildung 2.2 Aufbau von MVC bzw. GIS-Systemn

Ein besonderes Augenmerk in dieser Definition sollte man auf den Begriff der raumbezogenen Daten legen. Was raumbezogene Daten sind wird im weiteren Verlauf deutlich werden. Zuvor jedoch müssen noch einige Gesichtspunkte des Begriffes der Dimension erläutert werden. Die Abbildung 2.3 Dimensionen von Geometriedaten (entnommen aus [Bil 99/1]) stellt dies bildlich dar.

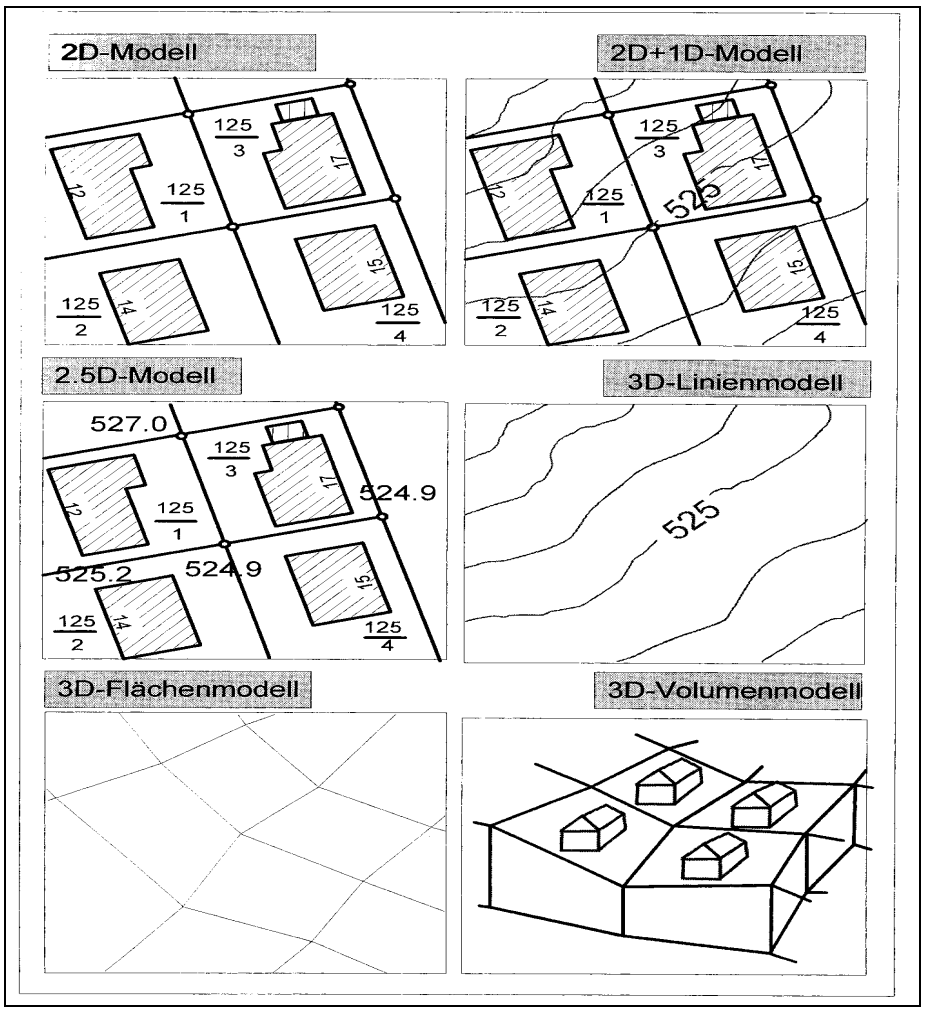


Abbildung 2.3 Dimensionen von Geometriedaten

Zum besseren Verständnis hier noch einmal die Charakteristika in Worte gefasst.

?? zweidimensional (2D), wenn sich die Geometriedaten nur auf die x,y-Koordinaten beziehen und Höhenangaben nicht weiter betrachtet werden.

?? zwei-plus-eindimensional (2D + 1D), wenn die zweidimensionale Darstellung durch eine digitale, durch Höhenlinien, Beschreibung der Geländeoberfläche ergänzt wird.

?? zweieinhalbdimensional (2.5D), wenn zur Lagegeometrie die Höhe z als Attribut gespeichert wird. Dies wird in der Abbildung 2.3 deutlich, da hier keine Höhenlinien gezeichnet werden, sondern einzelne Punkte eine Höhenangabe erhalten.

?? dreidimensional (3D), wenn die x,y,z – Koordinaten mit entsprechender Häufigkeit im gesamten zu bearbeitenden Gebiet vorhanden sind. Wie in der Abbildung zu erkennen ist, wird das 3D Modell nochmals in drei

verschiedene Darstellungen aufgeteilt und zwar in 3D-Linienmodell, 3D-Flächenmodell und 3D-Volumenmodell. Die verschiedenen Modelle sind verschieden aufwendig und es werden verschiedene Verfahren verwendet die Informationen zu speichern.

?? vierdimensional (4D), zu den Raumkoordinaten x,y,z wird der Zeitparameter (t) abgespeichert. Auch bei den anderen Dimensionen kann der Zeitparameter mit in betracht gezogen werden, so dass man z.B. von 2D + t Datenmodellen spricht.

2.3 Darstellung von Daten

In GIS-Systemen müssen verschiedenste Informationen graphisch dargestellt werden. Dazu werden überwiegend Flächen, Punkte und Linien verwendet, die dann die Informationen wiederspiegeln sollen. Aus der graphischen Datenverarbeitung wissen wir, dass es im Endeffekt zwei verschiedene Möglichkeiten der Darstellung gibt. Zum einen die Rasterdarstellung und zum anderen die Vektordarstellung, wobei die Rasterdarstellung auf Pixel aufgebaut ist und die Vektordarstellung auf Punkten und Linien bassiert.
In GIS-Systemen werden Vektordaten und Rasterdaten zur Darstellung und Speicherung der Informationen verwendet. Deshalb sollen hier noch einmal die wichtigsten Kriterien zur Verwendung von Vektordaten und Rasterdaten aufgeführt werden, um dann die Frage zu beantworten, wann am effektivsten Rasterdaten bzw. Vektordaten eingesetzt werden.

2.3.1 Vektordaten und Rasterdaten

Bei Vektordaten wird von einem Ursprung ausgehend ein Punkt beschrieben, an dem ein Linienzug beginnt. Von diesem Punkt wird wiederum durch einen Vektor der nächste Punkt angegeben u.s.w., bis das gesamt darzustellende Objekt beschrieben ist (siehe Abbildung 2.4 b).

Bei Rasterdaten werden einzelne Punkte zur Darstellung des Objektes genutzt, wie in Abbildung 2.4 c zu erkennen ist. In Abbildung 2.4 a ist das reale gezeichnete Objekt zu erkennen.

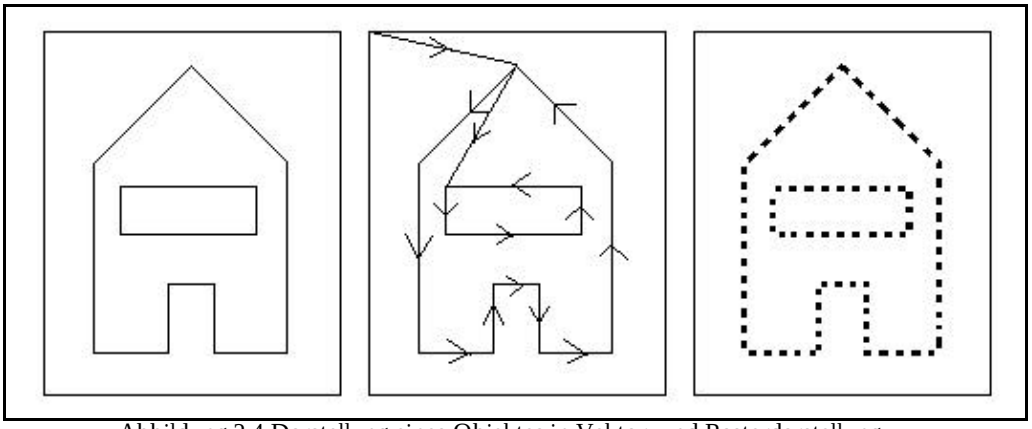


Abbildung 2.4 Darstellung eines Objektes in Vektor- und Rasterdarstellung

Welche Vorteile bzw. Nachteile haben nun Vektordarstellung und Rasterdarstellung ? Welche Datenmengen treten auf ? Wie steht es mit der Datenerfassung und wo findet sich welche Darstellung in GIS-Systemen wieder ? Bei der Datenerfassung werden verschiedenste Geräte eingesetzt. Ein Gerät, das inzwischen überall Einzug gefunden hat, ist der Scanner. Bei einem Scanner werden zeilenweise einzelne Pixel eingelesen, d.h. der Scanner arbeitet mit der Rastertechnik. Weitere Geräte, die zur Datenerfassung die Rastertechnik einsetzen sind z.B. CDD-Kameras. Der Nachteil, den die Rastertechnik mit sich bringt, sind die großen anfallenden Datenmengen, da jedes einzelne Pixel beschrieben werden muss und dann noch ggf. verschiedenste Farbtöne dem Pixel zugeordnet werden müssen. Hier setzt der große Vorteil der Vektordarstellung ein. Für eine Linie muss nur der Anfangspunkt und der Endpunkt bekannt sein - also zwei Punkte - und dieser gesamten Linie kann dann die Farbinformation einmal mitgegeben werden, dies reduziert die Datenmengen erheblich.
Weitere Informationen zur Erfassung, Darstellung und Technik von Vektor- bzw. Rasterdarstellung sind zu finden in [Bil 99/1] bzw. in [ESK 96/1].

Betrachten wir nun noch einmal die einzelnen Eigenschaften, die für Vektordarstellung gelten :

?? Punkte und Linien als graphische Grundstruktur, Flächen als geschlossener Linienzug.

?? einfache Änderung von Schraffur, Strichausdehnung und Strichdicke
?? Daten und Objektlinien geordnet, dadurch linienhafte Betrachtung möglich.
?? logische Datenstrukturierung und Objektbezug möglich
?? geringe Datenmenge
?? kurze Rechenzeiten

Dem gegenüber stellen wir noch einmal die Eigenschaften der Rasterdarstellung :

?? Pixel als graphische Grundstruktur
?? Flächenhafte Betrachtungsweise
?? Ordnung nur nach der Position der Pixel
?? logische Datenstrukturen und Objektbezug sehr eingeschränkt
?? einfache Datenerfassung, kurze Erfassungszeiten
?? große Datenmengen

In GIS-Systemen werden sowohl Vektordarstellungen als auch Rasterdarstellungen verwendet. Dies beginnt bereits bei der Erfassung der Daten und setzt sich dann in der Verarbeitung und der Speicherung dieser fort. Vektordaten werden erfasst in Geländeaufnahmen, Luftbildern und Digitalisierung. Informationsquellen, die Rasterdaten liefern, sind z.B. Satelliten, CCD-Kameras oder Scanner. Diese neuen Informationen werden dann noch mit alphanumerischen Daten oder vorhandenen Sachdaten ergänzt.

Die geometrischen Elemente und deren graphische Repräsentation werden, im Bezug auf GIS-Systeme in Abbildung 2.5 verdeutlicht. Diese Abbildung wurde aus [Bil 99/1] entnommen.

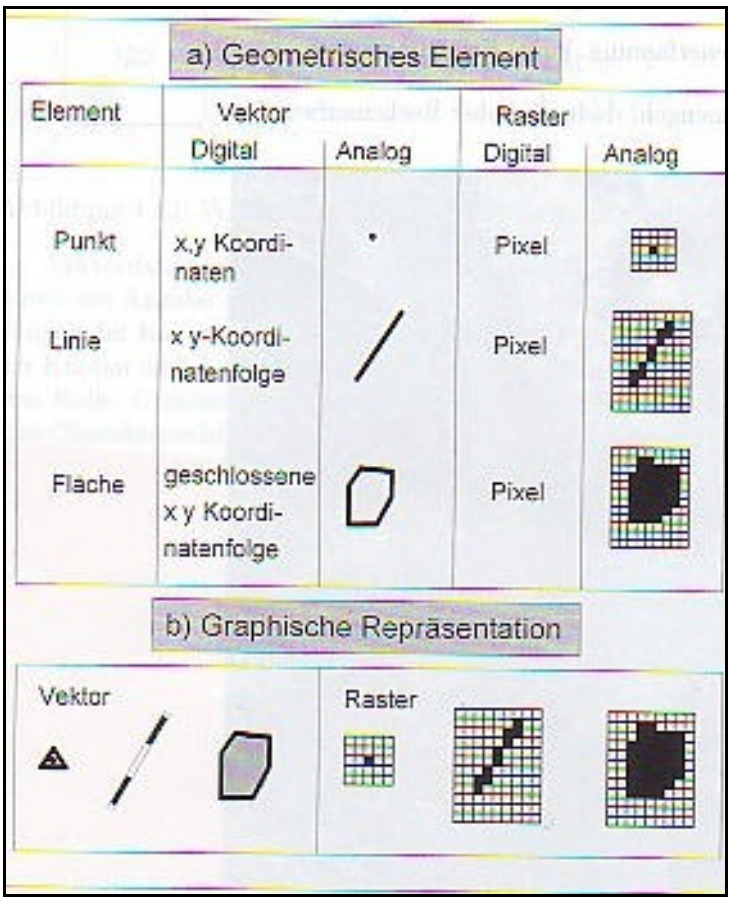


Abbildung 2.5 Geometriedaten und deren graphische Ausgestaltung

Aus Abbildung 2.5 kann entnommen werden, dass für Flächen, denen keine weiteren Informationen zugeordnet werden müssen, Rastergrafiken einfacher einzusetzen sind (z.B. Waldgebiete). Dagegen ist die Darstellung einer Bahnlinie in der Vektordarstellung einfacher zu realisieren , Flächen hingegen sind mit Hilfe der Vektorgrafik etwas komplizierter zu beschreiben.

Nun stellt sich die Frage, wann werden in GIS-Systemen Vektordaten und wann werden Rasterdaten eingesetzt. Dazu wollen wir uns einmal ein GIS-System etwas näher ansehen.

2.3.2 Maßstäbe

Dabei fällt uns auf, dass in einem GIS-System mit verschiedenen Maßstäben gearbeitet wird. Dies ist auch recht einleuchtend, betrachten wir z.B. nur einmal eine Deutschlandkarte. Angenommen wir wollen eine Reise von Köln zum

Bodensee machen, um dann dort die Gegend zu erwandern. Dazu benötigen wir zwei verschiedene Karten. Zum einen eine Autobahnkarte auf der wir den Weg von Köln zum Bodensee finden - hier haben wir einen kleinen Maßstab von 1 : 400000 - da wir ja nur die Verbindungen via Autobahn von Köln nach Konstanz benötigen. In Konstanz angekommen gehen wir nun auf die Wanderschaft und hier benötigen wir dann einen anderen Maßstab, da uns ja hier Fußwege interessieren, so dass wir hier eine Karte mit einem großen Maßstab von vielleicht 1 : 1000 einsetzen.

Ähnlich arbeitet ein GIS-System auch. Zuerst hat man einen kleinen Maßstab, wo dann z.B. ein Land dargestellt wird. Wenn man nun einen Punkt auf dieser Karte immer weiter vergrößert, ändert sich der Maßstab und es ist dann möglich erst Städte, dann einzelne Stadtteile bis hin zu einzelnen Straßen zu erkennen. Wird das GIS-System jetzt z.B. bei einem Energieversorger eingesetzt, kann dies dazu führen, dass einzelne Leitungen für Strom bzw. Gas in der Straße zu erkennen sind.

Hierbei muss dann das GIS-System in der Lage sein, von einer Rasterdarstellung in den kleinen Maßstäben zu einer Vektordarstellung in den großen Maßstäben zu skalieren.

Wann spricht man von großen, mittleren und kleinen Maßstäben ? Wann werden Vektorgrafik und wann wird Rastergrafik eingesetzt ? Was ist Hybride Grafik ? Auf diese Fragen gibt uns Abbildung 2.6 Auskunft.

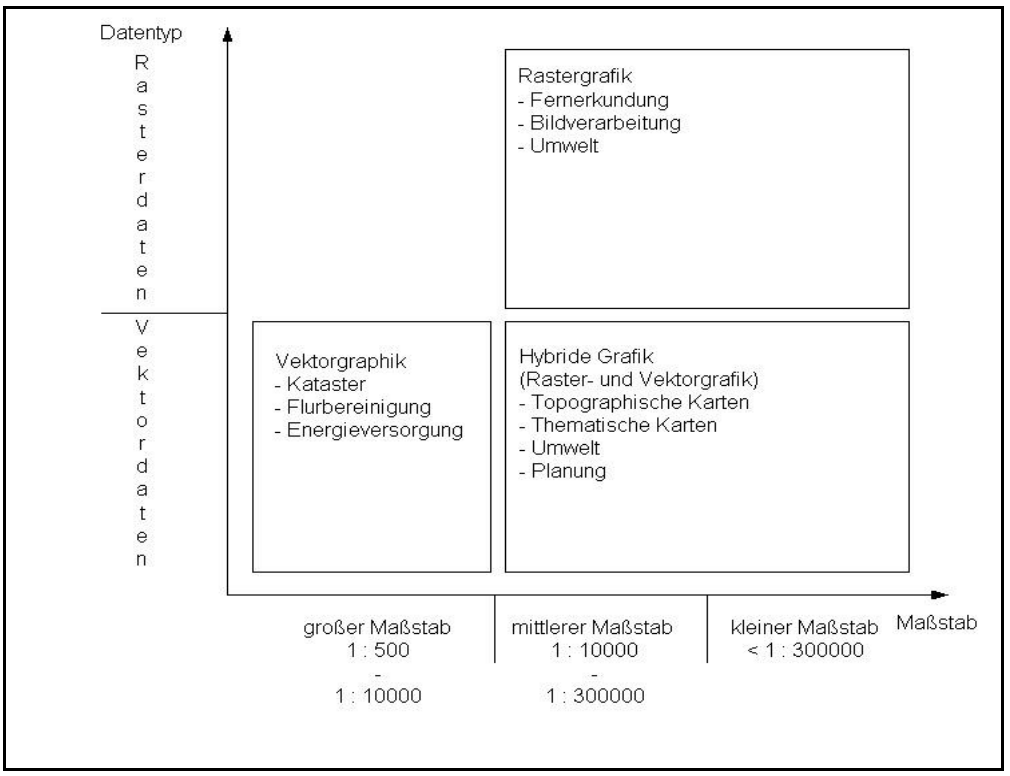


Abbildung 2.6 Maßstäbe

Aus Abbildung 2.6 kann man entnehmen, dass es drei verschiedene Kategorien von Maßstäben gibt. Zum einen der große Maßstab mit einer Auflösung von 1 : 500 bis 1 : 10000 , dann einen mittleren Maßstab von 1 : 10000 – 1 : 300000 und als drittes den kleinen Maßstab größer 1 : 300000.
Der Abbildung ist weiter zu entnehmen, dass die Vektorgrafik bei großen Maßstäben und die Rastergrafik vorzugsweise bei kleinen Maßstäben eingesetzt wird. Die Hybride Grafik, d.h. der Einsatz von Vektorgrafik oder Rastergrafik, bzw. Mischformen dieser finden ihren Einsatz im Bereich der mittleren Maßstäbe bis hin zu den kleinen Maßstäben.

2.4 Datenspeicherung und Verarbeitung

Ein weiterer wichtiger Punkt, der noch erläutert werden muss, sind Strategien zur Datenspeicherung und Verarbeitung. Hier spielt die Datenspeicherung mittels Rasterdaten bzw. Vektordaten ebenfalls eine entscheidende Rolle.
Betracht man z.B. eine Karte in der Brunnen aufgeführt sind, so haben wir dies beispielhaft als Punkte gegeben (siehe Abbildung 2.7).

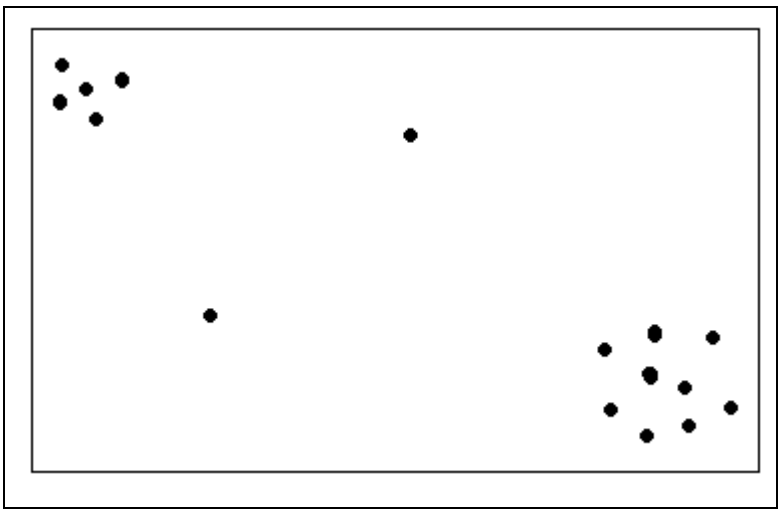


Abbildung 2.7 Brunnenkarte

Wie man nun erkennen kann, sind die Brunnen als Punkte gegeben. Eine weitere Auffälligkeit ist die Häufung der Brunnen. Wie ist es nun möglich diese Daten möglichst einfach und effizient abzuspeichern ?
Wenn man ein Raster über die Karte legt erkennt man, dass in einigen Feldern viele und in anderen Feldern (in diesem Beispiel sogar in der Mehrzahl) keine Brunnen eingetragen sind (siehe Abbildung 2.8).

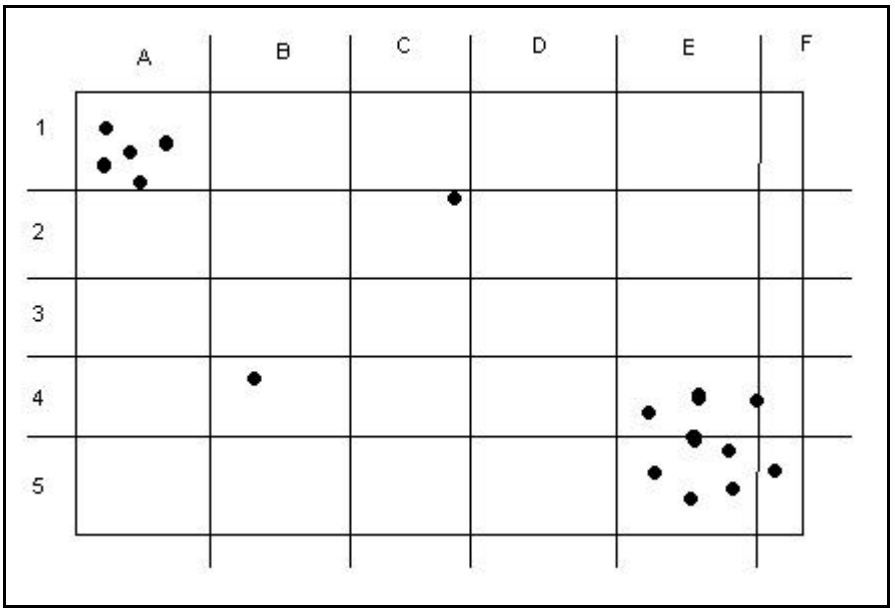


Abbildung 2.8 Brunnenkarte mit Raster

Wie nun zu sehen ist, kann man die Brunnen, die in einem Feld liegen speichern. Man kann also speichern :

Brunnen = { A1, C2, B4, E4, E5, F5 }

Man kennt nun die Felder in denen jeweils Brunnen liegen. Man weiss aber nicht, wie viele Brunnen in dem entsprechenden Feld liegen, bzw. die genaue Position des Brunnens. Diese Situation ist vergleichbar mit einem kleinen Maßstab. Die Anzahl der Brunnen könnte man mit einem zusätzlichen Parameter speichern, so dass man speichern könnte :

Brunnen = {A1(5), C2(1), B4(1), E4(4), E5(4), F5(1) }

Wenn man nun beginnt das Raster zu verfeinern ist es möglich, die Brunnen sehr genau in der Lage zu bestimmen.
Nun steht man vor der Frage, welche effektivere Möglichkeit es gibt, die Brunnen zu speichern. Die Lösung, die angewendet wird, ist der Quadtree. Bei diesem Ansatz wird die Karte zuerst in vier gleichgroße Quadrate aufgeteilt. Diese Quadrate werden dann an den Stellen, wo sich ein Brunnen befindet, weiter verfeinert. Das Ergebnis wird dann mittels eines Baumes gespeichert. Die Aufteilung der Karte in die entsprechenden Quadrate ist in Abbildung 2.9 ersichtlich.

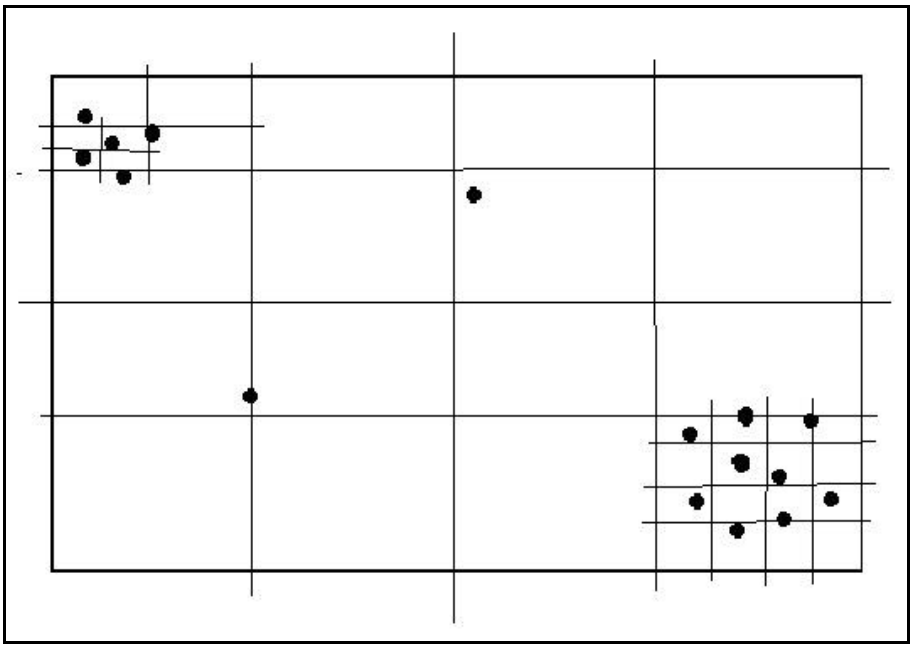


Abbildung 2.9 Quadtree-Prinzip

In dieser Abbildung ist zu erkennen, dass die genaue Speicherung der Position der Brunnen einen geringeren Aufteilungsaufwand voraussetzt, als die gleiche

Genauigkeit bei der Aufteilung in gleichgroße Felder, die dann in einer Liste gespeichert werden.
Des weiteren ist es möglich, sich auch nur für bestimmte Positionen auf der Karte eine genaue Aufschlüsselung zu erstellen.
Nach dieser Betrachtung zur effektiven Speicherung von Punkten, wollen wir nun einmal die Vorgehensweise bei Strecken betrachten. Dazu stellen wir uns vor, wir müssen das Straßennetz eines Ortes daraufhin untersuchen, ob es möglich ist von Punkt A nach Punkt B zu gelangen. Hierzu haben wir wieder eine Karte und die beiden Punkte gegeben (siehe Abbildung 2.10).

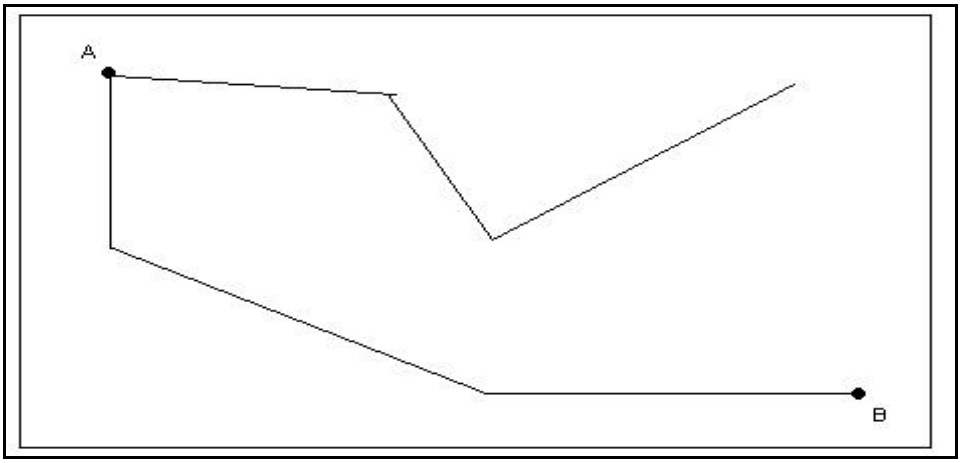


Abbildung 2.10 Verbindungskarte

In der Abbildung sind die Punkte A und B sowie die Wege zu erkennen.
Als nächstes legen wir ein Raster über die Karte und speichern die Strecken ab, die von einem Feld des Rasters in ein benachbartes Feld übergehen (siehe Abbildung 2.11).

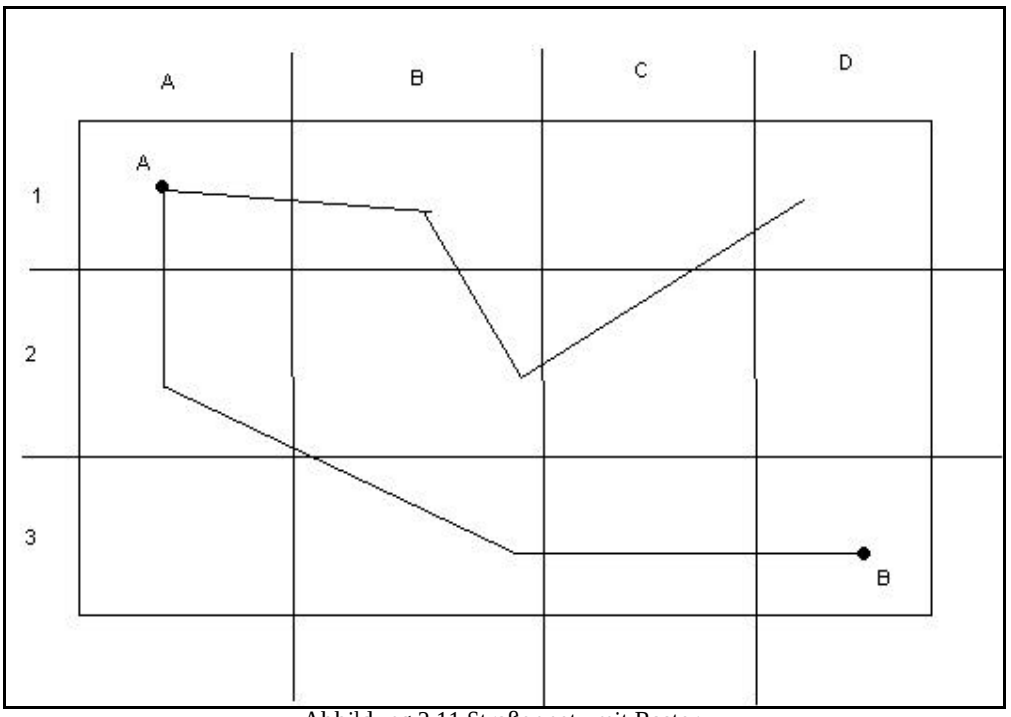


Abbildung 2.11 Straßennetz mit Raster

Hier wird nun gespeichert welche Verbindungen bestehen. Beginnend mit A1-B1 wird dies entsprechend mit allen Verbindungen durchgeführt.

Verbindungen = { A1-B1, B1- B2, B2-C2, C2-C1, C1-D1, A1-A2, A2-B3, B3-C3, C3-D3 }

Mit Hilfe dieser Verbindungen ist es jetzt möglich festzustellen, ob ein Weg von A nach B existiert. Dazu muss festgestellt werden wo der Punkt A liegt.

Punkt A liegt in Feld A1.

Dann muss festgestellt werden wo Punkt B liegt.

Punkt B liegt in Feld D3.

Jetzt muss mit Hilfe eines geeigneten Graphalgorithmus die Liste der Verbindungen durchsucht werden, um festzustellen, ob es eine Verbindung zwischen den Feldern A1 und D3 gibt. Hierzu kann beispielsweise der Algorithmus Topologisches Sortieren oder auch der Algorithmus zum feststellen des kürzesten Weges in Graphen genutzt werden.

Weiter Informationen zu Graphalgorithmen finden man in [Sed 99] oder [ALG 00].

K a p i t e l 3

Datenbanksysteme

In diesem Kapitel wird kurz die Funktionsweise von Datenbanksystemen angerissen und an den Beispielen Oracle und MySQL vertieft. Des weiteren wird erläutert, warum sich im Projekt “Wassermanagement in Chile“ für die Oracle Datenbank entschieden wurde und was dies dennoch an Problemen mit sich brachte.
Außerdem wird in diesem Kapitel ein Datenmodell entworfen, das für das Projekt einsetzbar ist. Hierzu werden dann die Entscheidungskriterien diskutiert und zum Abschluss des Kapitels wird noch ein Auszug aus dem Datenmodell modelliert, der dann für das Kapitel 4 (Technologien mittels Java auf Datenbanken zuzugreifen) konkret genutzt wird.

3.1 Datenbanksysteme

Heute findet man große Datenbankensysteme, die von verschiedenen Herstellern angeboten werden. Es finden sich aber auch inzwischen leistungsstarke Alternativen aus dem Bereich der Open-Source-Software. An dieser Stelle soll nur Oracle für die kommerzielle Seite genannt werden bzw. MySQL für die Seite der Open-Source-Software, da für das Projekt “Wassermanagement in Chile“ auch zwischen diesen beiden Produkten eine Wahl getroffen werden musste.
Verdeutlichen wir uns aber zuvor noch einmal kurz, was eine Datenbank ist und was diese leisten muss.
Wir finden in [DBS 00] eine entsprechende Definition, die hier kurz wiederholt werden soll :

> „ Eine Datenbank ist eine Ansammlung von Daten, die allen Benutzern bzw. Anwendungen zur Verfügung steht und in der die Daten nach einheitlichen Regeln abgespeichert werden. Eine Datenbank besteht aus einer Datenbasis und einem Datenbasisverwaltungssystem (DBMS).“

Von dieser Definition ausgehend müssen noch die Begriffe Datenbasis und DBMS geklärt werden. Auch hierzu findet man in [DBS 00] entsprechende Definitionen die hier der Vollständigkeit halber noch einmal aufgeführt sind.

„ Unter der Datenbasis versteht man die eigentlichen Daten der Datenbank im Dateisystem, also eine gewisse Anzahl von physikalischen Dateien, in denen die Anwendungsdaten und das Data-Dictionary gespeichert sind.
Das Data-Dictionary (oder auch Metadaten), enthält Daten, die die Datenbasis z.B. Tabellenstrukturen, definieren. Es beinhaltet außerdem Daten über Verwendung und Bedeutung des Datenmodells, ihre Beziehungen untereinander und Integritätsbedingungen.
Das Data Base Managment System (DBMS) ist ein Softwaresystem zur Verwaltung der Datenbasis. Alle Prozesse der Datenbank werden gesteuert und verschiedenen Dienstleistungen zur Verfügung gestellt. Durch eigene Parametrisierung wird das System an spezielle Bedürfnisse angepasst. Dazu gehören z.B. Funktionen zum Speichern der Daten, zur Verwaltung der Ressourcen und Maßnahmen zur Sicherung der Datenkonsistenz. Ein DBMS liegt in der Funktionalität zwischen Anwendungsprogramm und Betriebssystem.“

Anhand dieser Beschreibung kann man sich schon ein recht genaues Bild darüber machen, wie ein Datenbanksystem aufgebaut ist. Tiefer soll an dieser Stelle jedoch der Aufbau eines Datenbanksystems nicht betrachtet werden.
Als nächstes sollten die Begriffe “Relationale Datenbank“ und “Objektrelationale Datenbank“ in ihrer Bedeutung klar werden. Dazu findet man in [DBS 00] folgendes :

„ Relationale Datenbank
Die Daten und ihre Beziehungen werden in Tabellen (Relationen) abgebildet. Eine Tabelle ist horizontal in Zeilen und vertikal in Spalten aufgeteilt. Jeder Datensatz wird als Zeile dargestellt. Die Zeilen heißen Tupel und die Spaltenüberschriften Attribute.“

Anhand dieser Beschreibungen kann man sich nun vorstellen, was eine Datenbank ist und was sie leisten muss. Weiter werden wir an dieser Stelle nicht

auf Datenbanken eingehen, da diese Grundlagen bereits seit der Datenbankvorlesung bekannt sein sollten.

Die beiden Produkte Oracle und MySQL erfüllen diese Voraussetzungen und bieten dem User jeweils eine komplett funktionierendes Datenbanksystem. Was sind nun die entscheidenen Unterschiede von Oracle und MySQL ?

MySQL ist ein Open-Source-Produkt, das im Rahmen der beigefügten Lizenzbedingungen genutzt werden kann. MySQL wurde von der schwedischen Consultingfirma TcX entwickelt. Diese benötigten ein flexibles und schnelles Datenbanksystem, das zu dieser Zeit nicht auf dem Markt vorhanden war und so entwickelten sie ihr eigenes System MySQL. Aufgrund der Lizenzbedingungen ist es möglich, MySQL umsonst einzusetzen, was dafür sorgte, dass Universitäten, gemeinnützige Organisationen und Internet Service Provider davon regen Gebrauch machten. Inzwischen wird MySQL auch in der Geschäftswelt eingesetzt. MySQL lässt sich mit Java sowie als auch mit PHP nutzen. Für beides findet man im Internet viele hervorragende Beispiele, um sich der jeweiligen Problematik zu nähern. Im Anhang sind mehrere Internetseiten aufgeführt, die sich mit MySQL, Java und PHP beschäftigen. Hier sind immer wieder aktuelle Informationen zu finden. Weiterhin ist es möglich, sich bei entsprechenden Foren und Newsgroups, Tipps und Informationen zu beschaffen, die bei der Problemlösung helfen können. Eine Einführung in MySQL ist findet man unter anderem in [Mas 01].

Als weiteres Datenbanksystem soll hier stellvertretend für alle anderen Datenbanksysteme kurz Oracle erwähnt werden. Oracle ist ein Datenbanksystem, das inzwischen als Version 9i zu erhalten ist, wobei jedoch die Version 8i im kommerziellen Einsatz die verbreiterte ist, da die Version 9i erst seit Anfang 2002 nutzbar ist. Oracle gehört mit zu den führenden Datenbanksystemen und ist in der Geschäftswelt sehr verbreitet. Für viele Anwendungen findet man spezielle Treiber, mit denen es möglich ist auf Oracle zuzugreifen. Selbst große Anwendungen wie z.B. SAP arbeiten direkt mit Oracle zusammen, bzw. werden so entwickelt, das sie als Datenbank Oracle nutzt.

Eine Einführung in Oracle 8i findet man in [ACA 00] oder in [Loc 97], wobei das letztgenannte auch inzwischen für Oracle 9i erschienen ist.

3.2 Oracle vs. MySQL

Für den Projekteinsatz “Wassermanagement in Chile“ wurde überlegt, welches Datenbanksystem einsetzt werden sollte. Für beide Systeme sprachen verschiedene Punkte, die hier kurz aufgeführt werden sollen.

Für MySQL sprachen :

- ?? geringe Kosten durch das Lizenzmodell
- ?? weite Verbreitung da Open-Source
- ?? schnell und für verschiedenste Möglichkeiten konfigurierbar
- ?? klein und flexibel für den ersten Testeinsatz

Gegen MySQL sprachen :

- ?? keine Unterstützung der Spatial Technologie
- ?? keine Möglichkeit eine Anbindung an ArcSDE durchzuführen

Diese beiden Punkte sorgten schon dafür, das MySQL nicht ausgewählt werden konnte.

Für Oracle sprachen :

- ?? weltweit bekanntes Produkt
- ?? Umsetzung von Spatial Technologie
- ?? Anbindung an ArcSDE mittels Treiber möglich
- ?? Oracle in Campuslizenz zu Testzwecken vorhanden

Gegen Oracle sprachen :

- ?? hohe Anschaffungskosten als Vollprodukt

Da jedoch ArcSDE eingesetzt werden sollte, wurde Oracle als Datenbanksystem ausgewählt. Im späteren Einsatz des gesamten Systems muss eine Lösung für die

Anschaffung des Datenbanksystems Oracle gefunden werden. Dies spielt besonders im Hinblick auf Entwicklungsländer eine entscheidende Rolle, da hier die Kostenseite nicht außer acht gelassen werden darf.

3.3 Ein Datenmodell

In diesem Kapitel wird ein Datenmodell entwickelt, welches für den Einsatz im Projekt "Wassermanagement in Chile" nutzbar wäre. Die Modellierung erfolgt mit dem Tool ErWin, mit welchem dann auch gleich das Skript zur Erstellung der Datenbank erzeugt wird. Dieses Skript ist vollständig im Anhang C zu finden. Hier soll zusammenfassend berichtet werden, warum welche Entscheidungen getroffen wurden. Um festzustellen, was modelliert werden sollte, wurden mehrere Gespräche geführt, die dafür sorgten, dass das Datenmodell immer wieder erweitert wurde. In den ersten Gesprächen wurde ein Pflichtenheft erstellt, dieses ist in Anhang E wiedergegeben.

Die wichtigste Entität des Datenmodells ist die Entität Fluss. Dieser Entität sollten verschiedenste Attribute zugeordnet werden. So ist es von Bedeutung, dass der Fluss einen Namen hat. Zur besseren Verwaltung wurde jedem Fluss eine ID zugeordnet, so dass es möglich wird das Modell für mehrere Flüsse, bzw. für Flusssysteme zu nutzen. Bei der Analyse der Entität Fluss zeigte sich, dass jeder Fluss irgendwo entspringt. Dies kann verschiedene Ursachen haben, so z.B. als Quelle, aber auch als Abfluss eines Sees. Diese Möglichkeiten sollten im Datenmodell berücksichtigt werden und werden unter dem Attribut Quellgebiet abgelegt. Des weiteren ist von Bedeutung, wohin der Fluss fließt, auch hier gibt es wieder verschiedene Möglichkeiten. Der Fluss kann in einen See oder in das Meer fließen, es ist aber auch möglich, das der Fluss in einen anderen Fluss mündet, dies wird im Attribut Mündungsgewässer abgelegt. Ein weiteres Attribut der Entität Fluss ist die Länge des Flusses in km. Somit haben wir schon einige Attribute für die Entität Fluss gesammelt und können uns dies in Abbildung 3.1 visuell vor Augen führen.

In der Abbildung ist ein weiteres Attribut zu erkennen, das als 'natürlich' bezeichnet ist, hier kann festgehalten werden, ob es sich um einen Kanal handelt. Wenn dies der Fall ist, würde in natürlich ein 'false' stehen, um zu zeigen, dass es

sich um ein künstliches oder besser ein von Menschenhand geschaffenes Gewässer handelt.
Weiter sind in der Abbildung 3.1 sofort die Datentypen der Attribute ersichtlich.

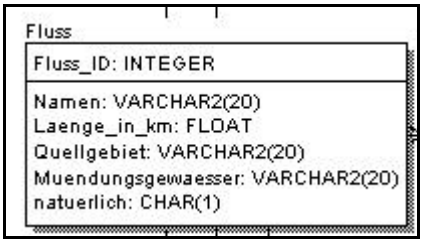


Abbildung 3.1 Entität Fluss

Die nächste Entität, die in diesem Zusammenhang betrachtet werden muss, ist die Entität Gewässer. Wie bei der Entität Fluss wird hier auch wieder eine ID genutzt, sie wird als Gewässer ID bezeichnet und dient unter anderem dazu, die Flüsse den Gewässern zuzuordnen, was in der entsprechenden Entität 'Gewässer-Fluss' dann vollzogen wird. In der Entität Gewässer findet man die Attribute Namen, für das Gewässer, Fläche in Quadratkilometer sowie Tiefe in Metern, diese Attribute sind bereits durch ihre Namen erklärt.
Dieser Teil des Modells ist vollständig in Abbildung 3.2 wiedergegeben.

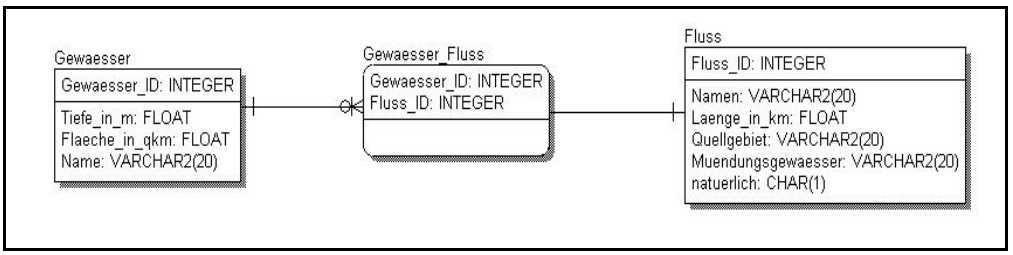


Abbildung 3.2 Modellierung Gewässer - Fluss

Zu beachten ist, dass ein Fluss in einen anderen Fluss münden kann, dies müsste dann mit einem Trigger modeliert werden.
Als nächstes müssen in das Datenmodell Messpunkte aufgenommen werden, da im Projekt "Wassermanagement in Chile" an verschiedenen Messpunkten verschiedene physikalische Größen gemessen werden sollen und diese Werte dann in einer Datenbank abgelegt werden sollen. Bei der Analyse dieser Zusammenhänge stellt man fest, dass es möglich wird, mit zwei Hauptentitäten zu arbeiten, zum einen mit der Entität Messpunkt zum andern mit der Entität Messwert.

Betrachten wir hier zuerst den Messpunkt und verdeutlichen uns erst einmal was ein Messpunkt überhaupt ist. Ein Messpunkt in unserem Fall ist ein konkreter Ort, der irgendwo auf dem Planeten Erde liegt. Punkte auf der Erde können genau mittels Längen- und Breitengrad angegeben werden. Des weiteren ist es wichtig zu wissen, ab wann es einen Messpunkt gegeben hat. Deshalb sollte das Einrichtungsdatum mit in das Datenmodell aufgenommen werden. Wie man der Abbildung 3.3 entnehmen kann, findet man die aufgeführten Attribute im Datenmodell wieder, zusätzlich ist noch eine ID für den Messpunkt aufgeführt und die Fluss ID um eine Beziehung zwischen dem Messpunkt und dem Fluss zu modellieren.

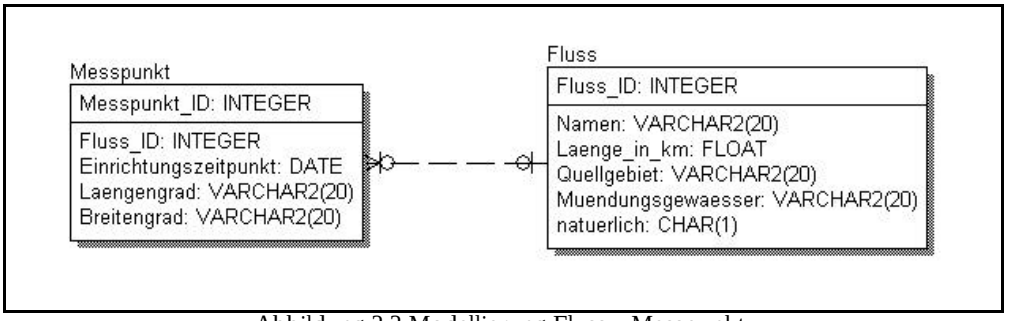


Abbildung 3.3 Modellierung Fluss – Messpunkt

Anmerkung zur Modellierung : Bei der Modellierung wurde in diesem Fall für den Messpunkt die Attribute Längengrad und Breitengrad mit varchar2(20) als Datentyp definiert. In wieweit sich Längengrad und Breitengrad darstellen lassen, müsste in einer konkreten Implementierung etwas genauer betrachtet werden, oder ob mittels der Spatial Technik eine andere Darstellung gewählt werden kann. Die Spatial Technik wird noch in einem speziellen Kapitel näher betrachtet.
Betrachten wir nun, wie in diesem Datenmodell die Messwerte modelliert wurden und wie es möglich, ist andere Messwerte hinzuzufügen. In Abbildung 3.4 sind die beiden Entitäten Messpunkt und Messwert dargestellt.

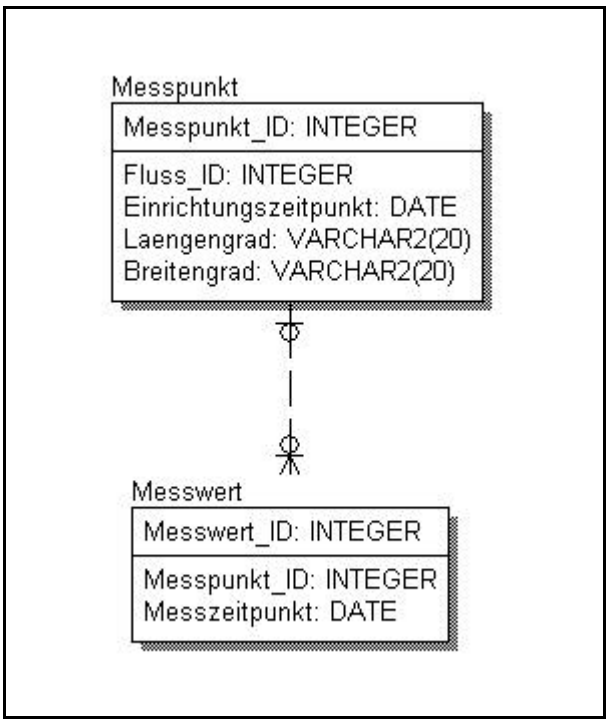


Abbildung 3.4 Messpunkt – Messwert

Es ist zu erkennen, dass die Entität Messpunkt mit der Entität Messwert erweitert wird. Dabei werden die Attribute Messwert ID zur besseren Verwaltung und der Messzeitpunkt hinzugefügt. Jetzt fehlen nur noch die Messwerte, die in der Datenbank abgespeichert werden sollen. Um dies zu erreichen wird für jede Art von Messwert eine Entität erstellt, die dann den Messwert aufnimmt (siehe Abbildung 3.5) .

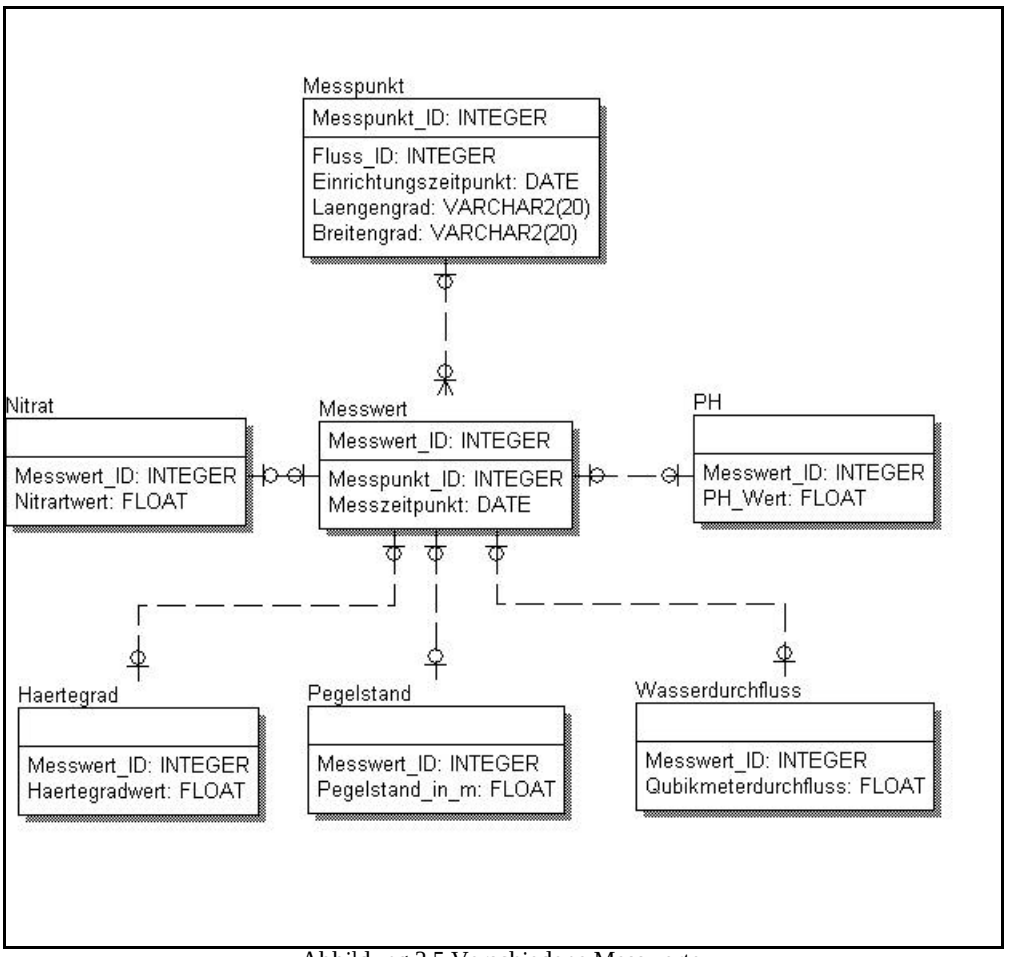


Abbildung 3.5 Verschiedene Messwerte

Hier sind nun beispielhaft die Entitäten für Nitrat, PH-Wert, Härtegrad, Pegelstand und Wasserdurchfluss dargestellt. Wenn man nun eine weitere physikalische Größe in der Datenbank speichern möchte, muss dafür nur die Entität erstellt werden und diese in Zusammenhang mit dem Messwert gebracht werden und schon ist es möglich, diese neue Größe ebenfalls zu speichern.
Betrachten wir hier noch als Beispiel wie der Nitritwert aufgenommen werden könnte. Dazu muss eine Entität Nitrit erstellt werden. Dieser Entität wird das Attribut Nitritmesswert zugeordnet, die Messwert ID wird von der Messwert-Entität übernommen. Abbildung 3.6 zeigt das erweiterte Datenmodell.

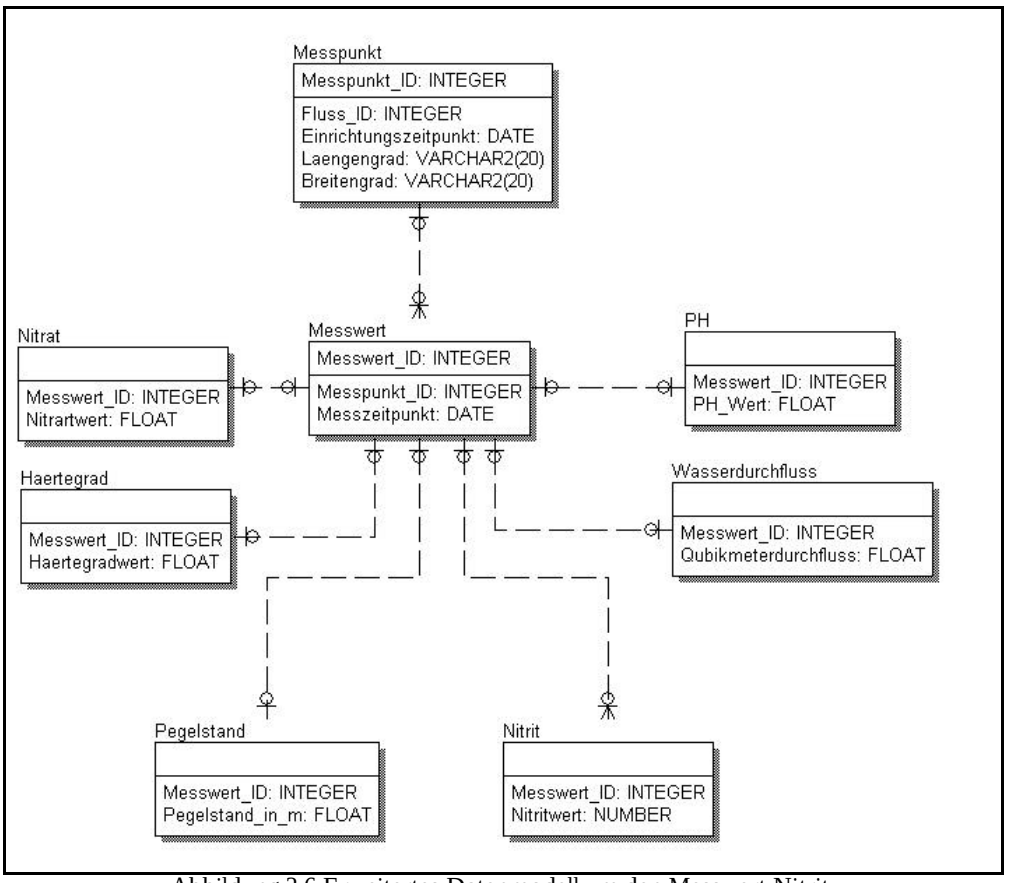


Abbildung 3.6 Erweitertes Datenmodell um den Messwert Nitrit

Nachdem nun die Möglichkeiten gegeben sind, Messpunkte einzurichten und Messwerte zu speichern , muss man sich Gedanken darüber machen, wie die Umgebung des Flusses in Betracht gezogen werden muss. Die Region in der der Fluss liegt kann Einflüsse auf die Messwerte nehmen. So ist es z.B. möglich, das durch intensive Landwirtschaft ein hoher Nitratwert auftreten kann. Aus diesem Grund muss in das Datenmodell die umgebende Region mit aufgenommen werden. Auch ist es vielleicht von Nöten die Regionen oberhalb des Messpunktes zu betrachten, was wieder ein Grund dafür ist, diese Informationen mit in die Datenbank aufzunehmen. Dies wird in ähnlicher Weise erreicht, wie die Modellierung der Messwerte. Zuerst wird eine Entität Region geschaffen und über eine weitere Entität wird dann die Verbindung Fluss und Region modelliert (siehe Abbildung 3.7).

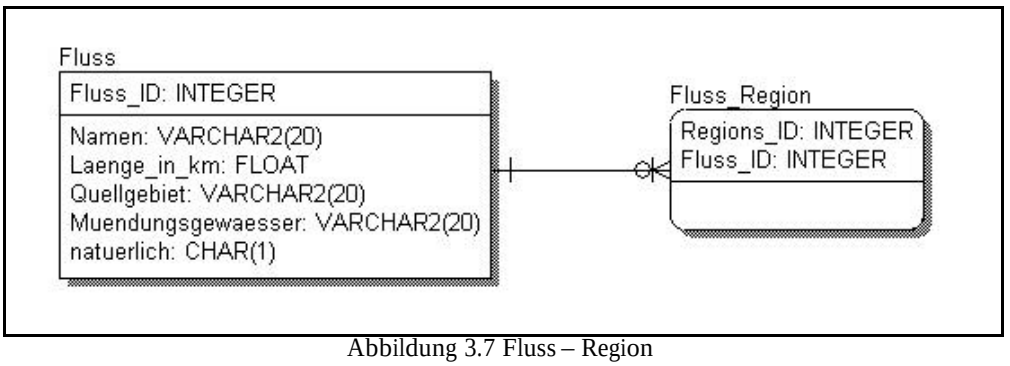


Abbildung 3.7 Fluss – Region

Nun wird die Entität Region mit weiteren Entitäten erweitert, so dass verschiedene Besonderheiten der einzelnen Regionen abgebildet werden können. In diesem Datenmodell wurden ersteinmal nur vier verschiedene Regionsarten unterschieden. Die Unterscheidung wurde aufgeschlüsselt in : Stadt, Industrie, Agrarwirtschaft und Viehwirtschaft. Wenn nun eine andere Differenzierung gefordert würde, müssten entsprechende Entitäten erstellt werden, dies würde genauso erfolgen wie in dem oben bereits gezeigten Beispiel für die Messwerte. In Abbildung 3.8 ist die derzeitige Aufschlüsselung wiedergegeben.

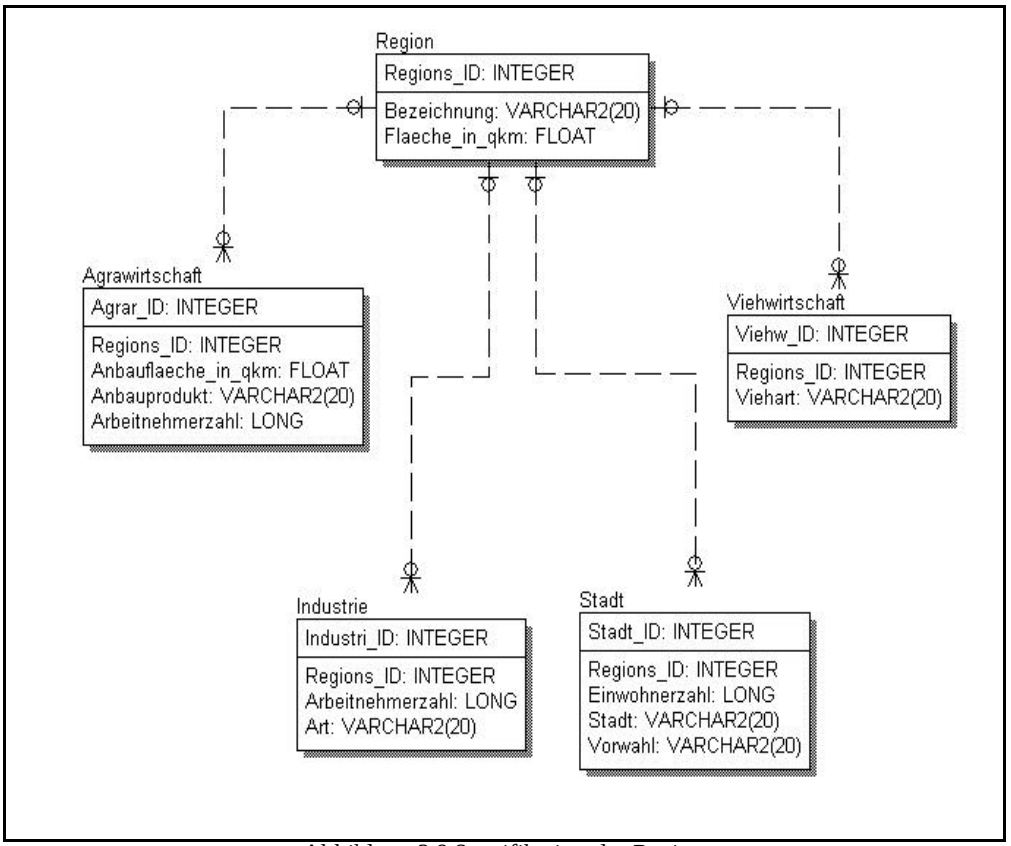


Abbildung 3.8 Spezifikation der Region

In Abbildung 3.8 sind neben der Entität Region, die weiteren Entitäten Agrarwirtschaft, Industrie, Stadt und Viehwirtschaft zu erkennen.
Die Entität Region besitzt die Attribute Regions ID, als Schlüssel, sowie die Attribute Bezeichnung und Fläche in Quadratkilometer. Diese Entität wird dann durch die anderen vier Entitäten ergänzt, die dann spezifische Attribute in das Datenmodell einbringen.
Für die Agrarwirtschaft ist es wichtig zu wissen, wie groß die Anbaufläche ist, welches Produkt angebaut wird und noch wie viele Arbeitnehmer beschäftigt sind.
Bei Industrievorkommen kann die Art der Industrie, Schwerindustrie oder Automobilbau, eine Rolle spielen. Auch kann hier die Arbeitnehmerzahl von Bedeutung sein, so dass diese in das Datenmodell aufgenommen wurde.
Im Bereich der Viehwirtschaft ist natürlich ein entscheidendes Attribut die Viehart, die gehalten wird.
Bei der Modellierung der Stadt stechen neben der Einwohnerzahl noch die Attribute Vorwahl und Stadt, als Namen der Stadt, ins Auge. Diese Attribute wurden in Hinsicht auf die Verwaltungsbehörden mit in das Datenmodell aufgenommen. Die Verwaltungsbehörden werden als nächstes in diesem Datenmodell betrachtet werden.
Jeder Fluss, bzw. jedes Gewässer, kann eine Verwaltungsbehörde besitzen. Diese Verwaltungsbehörden haben ihren Sitz meist in größeren Städten der Region. In den Verwaltungsbehörden ist eine Person als Ansprechpartner für das Gewässer bekannt. Die Verwaltungsbehörde besitzt eine Adresse und es können verschiedene Rufnummern auftreten. Diese Problematik muss ebenfalls in das Datenmodell aufgenommen werden. Dabei ist darauf zu achten, das bereits eine Stadt im Datenmodell vorhanden ist und diese Informationen müssen alle ineinander übergreifen.

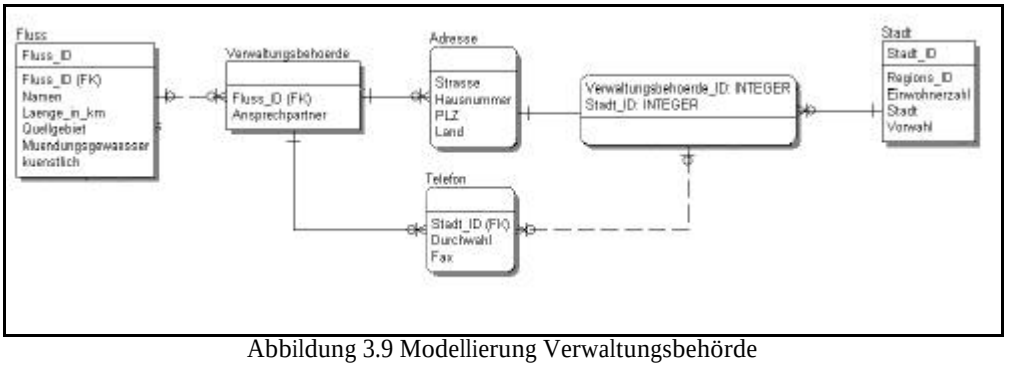


Abbildung 3.9 Modellierung Verwaltungsbehörde

In Abbildung 3.9 ist das entsprechende Datenmodell zur Verwaltungsbehörde zu erkennen. Dabei sind die beiden bereits bekannten Entitäten Fluss und Stadt noch einmal mit aufgeführt worden, um den Zusammenhang besser zu verdeutlichen. In der Abbildung ist zu erkennen, dass einem Fluss eine entsprechende Verwaltungsbehörde zugeordnet wird, diese Entität Verwaltungsbehörde besitzt als Attribut den Ansprechpartner. Diesem Ansprechpartner muss jetzt noch die Adresse und entsprechende Telefonnummern zugeordnet werden, dies geschied mittels der Entitäten Adresse und Telefon. Abschließend muss der Adresse noch eine Stadt zugeordnet werden. Die Entität spielt für die Entität Telefon in der Hinsicht eine entscheidende Rolle, dass eine Stadt immer eine feste Vorwahl besitzt und diese mit der Stadt gekoppelt ist und diese Vorwahl dann natürlich in der Entität Stadt gespeichert werden muss.
Abbildung 3.10 zeigt noch einmal das gesamte Datenmodell, das hier bereits erläutert wurde.

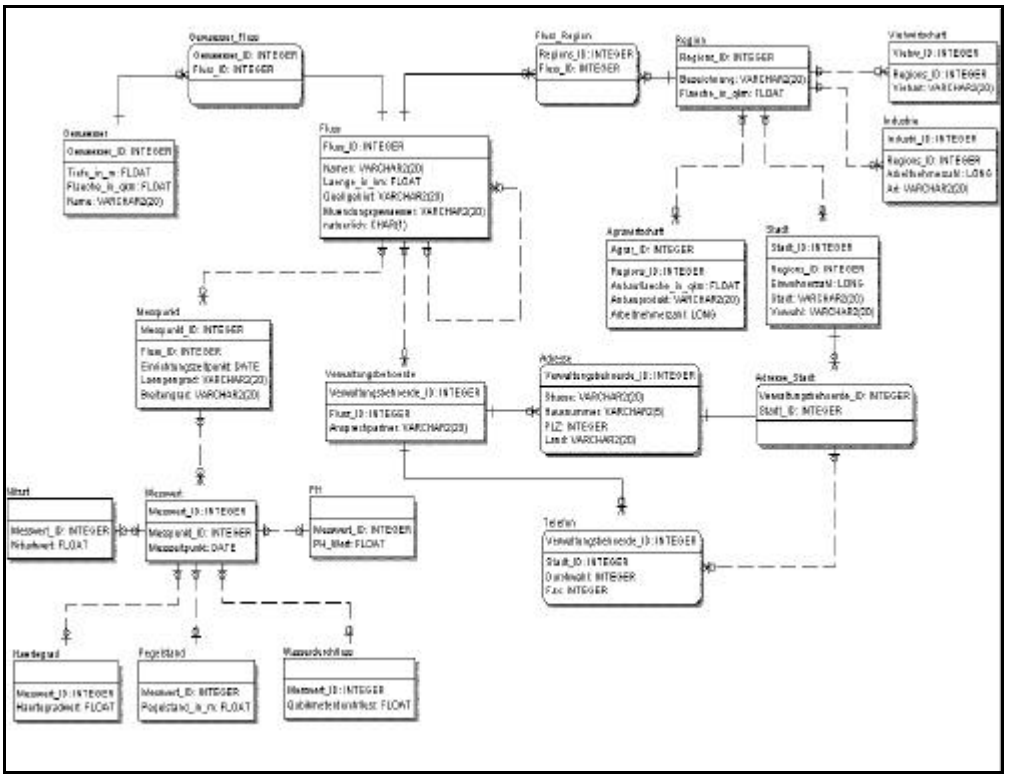


Abbildung 3.10 Gesamtes Datenmodell

Da dieses gesamte Datenmodell zur Demonstration von Datenbankzugriffen etwas zu umfangreich ist, wird ein kleineres Datenmodell entwickelt, welches dann in Kapitel 4 genutzt werden kann, um die Funktionalität der Software zu demonstrieren.

Das gesamte Skript, von beiden Datenmodellen, sowie die Erzeugung von Demodaten ist in Anhang C wiedergegeben.

Abbildung 3.11 zeigt das kleine Datenmodell. In diesem Datenmodell wird eine Reduzierung der Messwerte durchgeführt, da die Technik bei zugriff auf die Messwerte bei allen gleich ist, wird dies nur Beispielhaft an einem , in diesem Fall Messwert, demonstriert. Des weiteren sind die Regionen stark eingeschränkt worden, da es sich hier auch nur um eine beispielhafte Darstellung handeln soll.

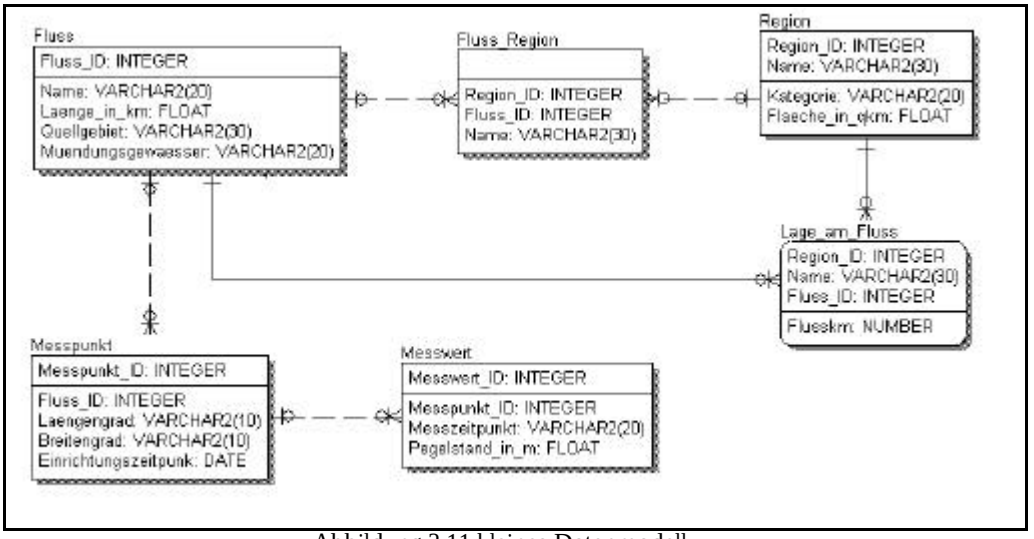


Abbildung 3.11 kleines Datenmodell

3.4 Die Oracle Spatial Technologie

In diesem Kapitel ist der Begriff "Oracle Spatial Technologie" schon des öfteren gefallen. Diese Technologie soll nun etwas genauer betrachtet werden.

In der Oracle Spatial Technologie wird ein objektrelationaler Ansatz verfolgt, der es Oracle 8i ermöglicht, mehrdimensionale räumliche Daten zu speichern, zu verarbeiten, zu analysieren und zu manipulieren. Um dies zu lösen stehen dem User etliche Funktionen und Prozeduren zur Verfügung. Wie schon zu erwarten war, ist der hauptsächliche Einsatzbereich der Oracle Spatial Technologie im Bereich von GIS-Systemen angesiedelt. Andererseits richtet sich die Oracle Spatial Technologie an Unternehmen, die ihre Anwendungen um eine geographische Komponente erweitern wollen und primär dann Daten

visualisieren wollen. Derartige Anwendungen sind häufig Bestandteile von Data Warehouse Lösungen und umfassen Möglichkeiten der raumbezogenen Auswertung, z.B. für Marketingplanung, Umsatzanalysen oder Standortplanungen.

3.5 Der Objekttype GEOMETRY

Die Oracle Spatial Technologie unterstützt drei einfache geometrische Datentypen, welche in Abbildung 3.12 dargestellt sind.
Zur Umsetzung verfolgt Oracle das Objektrelationale Datenmodell, auf das nun eingegangen werden soll.

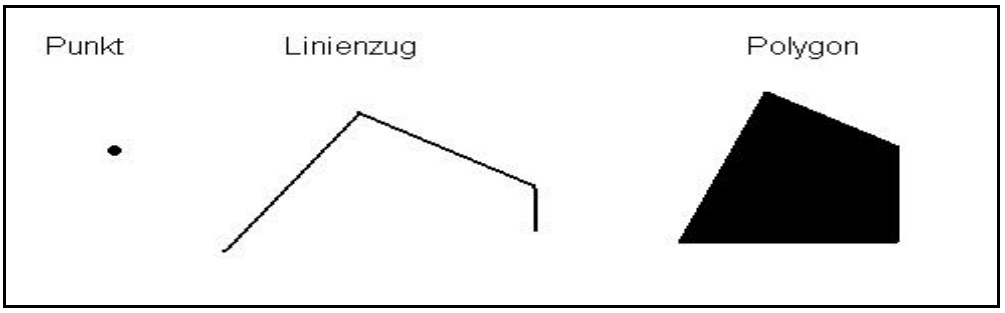


Abbildung 3.12 Einfache geometrische Datentypen

Bei der Umsetzung der Spatial Technologie hält sich Oracle an die Vorgaben der OGC (Open GIS – Consortium).

Um die Geometrien in der Datenbank ablegen zu können, wird in der Spatial Cartridge der Objekttyp “SDO_GEOMETRY“ angelegt. Dieser setzt sich aus mehreren Elementen zusammen.

Diese Elemente sind :

- SDO_GTYPE
- SDO_SRID
- SDO_POINT_TYPE
- SDO_POINT
- SDO_ELEM_INFO
- SDO_ORDINATES

Nun stellt sich die Frage, was sich hinter dem Objekttypen GEOMETRY verbirgt. Betrachten wir den Objekttypen genauer :

?? SDO_GEOMETRY, wird von der OGC vorgegeben. Die Definition für SDO_GEOMETRY lautet :

```
CREATE TYPE SDO_GEOMETRY AS OBJECT (
    SDO_GTYP NUMBER,
    SDO_SRID NUMBER,
    SDO_POINT SDO_POINT_TYPE,
    SDO_ELEM_INFO MDSYS.SDO_ELEM_INFO_ARRAY,
    SDO_ORDINATES MDSYS.SDO_ORDINATE_ARRAY);
```

Was verbirgt sich hinter den einzelnen Elementen ? Dazu betrachten wir die einzelnen Elemente genauer :

?? SDO_GTYPE, legt den Geometrie Type fest. D = Dimensionszahl, wird durch die Beispiele weiter unten deutlich (siehe Tabelle 3.1 SDO_GTYPE Werte).
?? SDO_SRID, wird zur Zeit nicht genutzt, ist aber reserviert für die spätere Angabe eines räumlichen Bezugssystems.
?? SDO_POINT_TYPE ist ein einfacher Typ aus drei Werten, der einen dreidimensionalen Punkt repräsentiert. Zur Zeit werden nur zwei Dimensionen unterstützt.
?? SDO_POINT wird nur verwendet, falls ein Wert vorliegt und die weiteren Dimensionen den Wert "NULL" erhalten sollen.
?? SDO_ELEM_INFO wird mit SDO_ORDINATES zusammen verwendet. SDO_ELEM_INFO ist ein Feld von Zahlen und gibt an wie die Werte in SDO_ORDINATES zu verwenden sind. SDO_ORDINATES ist ebenfalls ein Feld von Zahlen.

Wert	*Geometrieart*	*Beschreibung*
D000	UNKNOWN_GEOMETRY	wird ignoriert
D001	POINT	Punkt
D002	LINESTRING	Streckenzug
D003	POLYGON	Polygon mit oder ohne Öffnung
D004	COLLECTION	Sammlung von Elementen
D005	MULTIPOINT	mehrere Punkte
D006	MULTILINESTRING	mehrere Streckenzüge
D007	MULTIPOLYGON	mehrere disjunkte Polygone

Tabelle 3.1 SDO_GTYPE Werte

Betrachten wir einmal was mit dem Objekttypen GEOMETRY und seinen Elementen möglich ist.

Beispiel 1 : ein einfaches Rechteck.

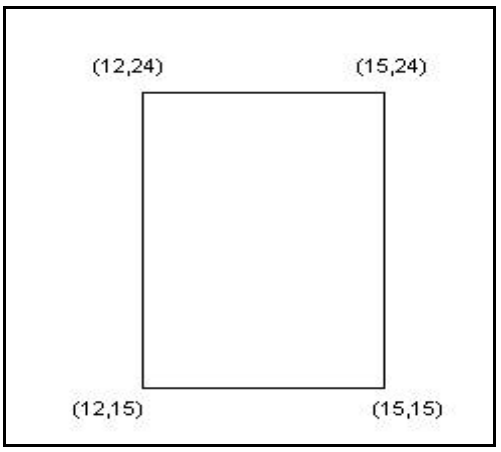


Abbildung 3.13 Rechteck

Das in Abbildung 3.13 gegebene Rechteck soll mittels der Spatial Technologie in einer Datenbank gespeichert werden, dazu wird die SDO_GEOMETRY Definition mit Werten gefüllt.

```
SDO_GEOMETRY Column = (
   SDO_GTYPE = 2003
   SDO_SRID = NULL
   SDO_POINT = NULL
   SDO_ELEM_INFO = (1,1003,3)
   SDO_ORDINATES = (12,15,15,24)   )
```

Betrachten wir nun einmal die eingesetzten Werte :
Bei SDO_GTYPE finden wir den Wert 2003, wie kommt man auf diesen Wert ? Da wir ein Polygon im zweidimensionalen betrachten, bekommt man als Dimensionszahl D = 2. Aus der Tabelle 3.1 können wir für das Polygon den Wert 003 entnehmen. Somit erhält man den Wert 2003.
SDO_SRID wird aus bekannten Gründen auf "NULL" gesetzt, was ebenfalls mit SDO_POINT getan wird.
Das Tripel von SDO_ELEM_INFO setzt sich wie folgt zusammen. Der erste Wert gibt an, bei welchem Wert von SDO_ORDINATES die Bearbeitung beginnt.
Der Wert 1003 gibt an, das es sich um das äußerste Polygon handelt, dies wird in Beispiel 2 klarer. Der letzte Wert dieses Tripel die 3 gibt an, dass es sich bei der

Figur um ein Rechteck handelt, bei dem nur die Ecke links unten und die Ecke rechts oben angegeben sind. In SDO_ORDINATES sind die Koordinaten der beiden Punkte angegeben, zum einen (12,15) zum anderen (15,24).
Es wäre natürlich möglich, dasselbe Rechteck auf eine andere Weise zu beschreiben. Betrachten wir diese Möglichkeit :

```
SDO_GEOMETRY Column = (
   SDO_GTYPE = 2003
   SDO_SRID = NULL
   SDO_POINT = NULL
   SDO_ELEM_INFO = (1,1003,1)
   SDO_ORDINATES = (12,15,15,15,15,24,12,24,12,15)   )
```

Als erstes erkennt man, dass diesmal viel mehr Koordinaten angegeben wurden. Woran liegt das ?
In der SDO_ELEM_INFO wurde der dritte Wert auf 1 gesetzt. Dies besagt, dass einzelne Punkte eines Polygonzuges angegeben werden. Dabei ist darauf zu achten, dass für ein geschlossenes Polygonzug der letzte angegebene Punkt mit dem ersten angegebenen Punkt übereinstimmt. Also sind dann hier in SDO_ORDINATES die einzelnen Eckpunkte des Polygons angegeben.
Weitere Informationen zu SDO_ELEM_INFO findet man in der Oracle Spatial User's Guide and Reference.

Beispiel 2 : ein Polygon mit einer Öffnung

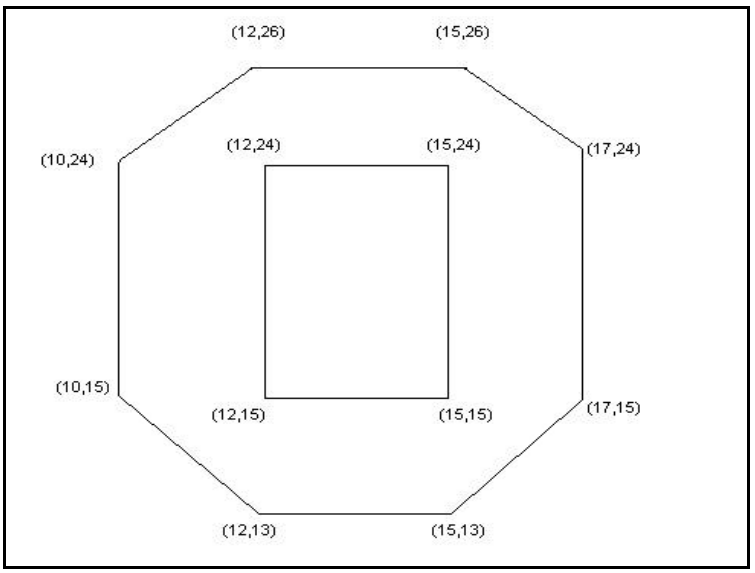


Abbildung 3.14 Polygon mit rechteckiger Öffnung

Die gegebene Geometrie soll nun auch wieder mittels des Objekttyps "GEOMETRY" beschrieben werden, um den Inhalt des Tripels des Element SDO_ELEM_INFO besser verstehen zu können.

```
SDO_GEOMETRY Column = (
   SDO_GTYPE = 2003
   SDO_SRID = NULL
   SDO_POINT = NULL
   SDO_ELEM_INFO = (1,1003,1, 19,2003,1)
   SDO_ORDINATES = (12,13,15,13,17,15,17,24,
                    15,26,12,26,10,24,10,15,
                    12,13,
                    12,15,15,15,15,24,12,24,
                    12,15)  )
```

Schauen wir uns nun das Element SDO_ELEM_INFO an. Man erkennt, das hier zwei Tripel angegeben wurden. Das erste Tripel beschreibt die Informationen für das äußere Polygon. Die 1 steht dafür, dass die Werte ab Position 1 beginnen. 1003 gibt an, dass es sich um ein äußeres Polygon handelt und die nächste 1 gibt an, dass es sich um Punkte handelt, die aufgeführt sind. Das zweite Tripel beschreibt die Informationen für das innere Polygon. Die 19 gibt an, dass die Beschreibung des Polygons mit dem neunzehnten Wert beginnt. 2003 besagt,

dass es sich um ein inneres Polygon handelt und die 1 besagt wieder, dass es sich um einzelne Punkte handelt.
Es wäre auch wieder möglich die Geometrie anders zu beschreiben. Da es sich bei der Öffnung um ein Rechteck handelt, kann natürlich auch die Beschreibung wie aus Beispiel 1 erfolgen (siehe nachfolgende Beschreibung).

```
SDO_GEOMETRY Column = (
   SDO_GTYPE = 2003
   SDO_SRID = NULL
   SDO_POINT = NULL
   SDO_ELEM_INFO = (1,1003,1, 19,2003,3)
   SDO_ORDINATES = (12,13,15,13,17,15,17,24,
                    15,26,12,26,10,24,10,15,
                    12,13,
                    12,15,15,24)   )
```

Abschließend soll noch gezeigt werden, wie effizient es mit diesem Ansatz ist, die Figur aus Abbildung 3.15 zu beschreiben.

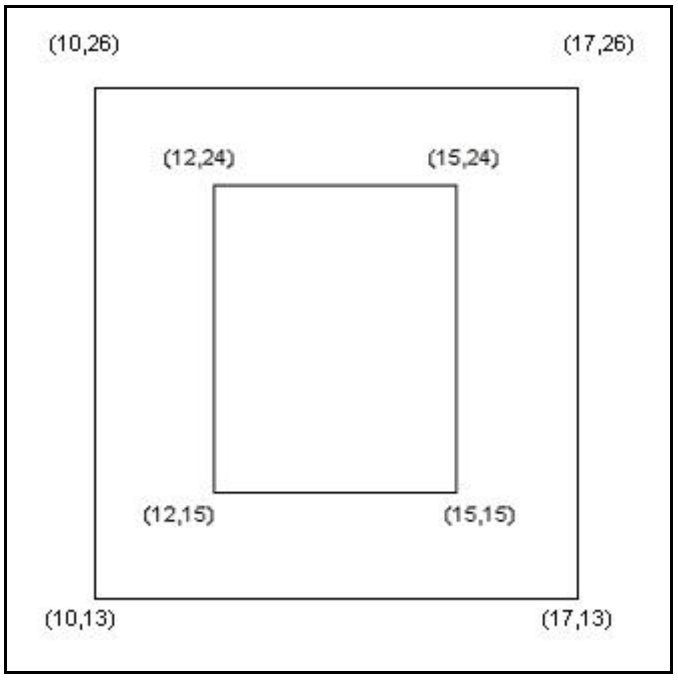


Abbildung 3.15 Rechteckprofil

```
SDO_GEOMETRY Column = (
   SDO_GTYPE = 2003
```

SDO_SRID = NULL
SDO_POINT = NULL
SDO_ELEM_INFO = (1,1003,3, 5,2003,3)
SDO_ORDINATES = (10,13,17,26,
12,15,15,24))

3.6 Indexing

Die Techniken, die im Abschnitt Datenspeicherung und Verarbeitung gezeigt wurden, werden auch in der Oracle Spatial Technologie umgesetzt.
Hier findet man zum einen das “fixed indexing“, das mit einer festen Rasterung arbeitet und zum anderen das “hybrid indexing“, das die Technik des Quadtree verwendet.
Betrachten wir nun, wie diese Technik in der Oracle Spatial Technologie umgesetzt wurde.
Um eine geometrische Figur in einem Feld darzustellen benötigt man ein Raster. In der Spatial Technologie wird für die Rasterung eine bestimmte zählweise eingeführt. Ein Quadrat wird immer wieder in kleinere Quadrate aufgeteilt. Dabei werden diese Quadrate immer nach derselben Methode nummeriert. Das unterste linke Quadrat erhält die 0. Das Quadrat rechts neben der 0 erhält die 1. Über der 0 liegt die 2 und über der 1 die 3, entsprechend wird ein Raster durchnummeriert (siehe Abbildung 3.16).

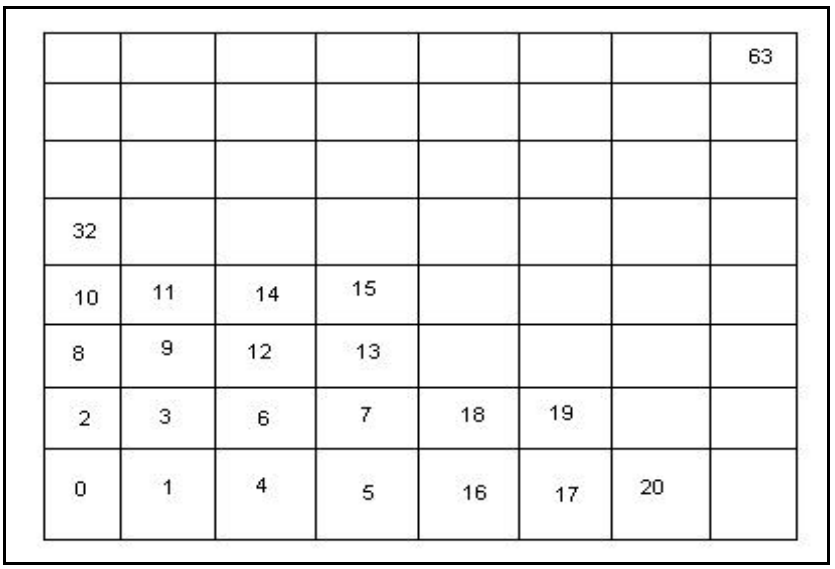


Abbildung 3.16 Rasterung

Bei einem “fixed indexing“ wird diese Rasterung beibehalten und die Felder in denen die Endpunkte einer Figur liegen werden gespeichert. Das Problem liegt aber im Aufwand der getrieben werden muss, um eine weitere Verfeinerung der Rasterung zu erhalten, dies führt zu sehr hohem Rechenaufwand.
Das “hybrid indexing“ nutzt die Idee des Quadtree, siehe auch Kapitel 2.4. Hierbei wird die Nummerierung etwas anders ausgeführt. Die gesamte Fläche wird in vier gleich große Quadrate aufgeteilt, die dann die Nummern 0 bis 3 erhalten. Die Felder die einen Endpunkt enthalten werden entsprechend verfeinert, wobei z.B. das Quadrat 3 wieder verfeinert wird und in vier gleiche Quadrate aufgeteilt wird, die dann als 30, 31, 32 und 33 bezeichnet werden. Diese Verfeinerung kann bis zu einer beliebigen Genauigkeit durchgeführt werden (siehe Abbildung 3.17).

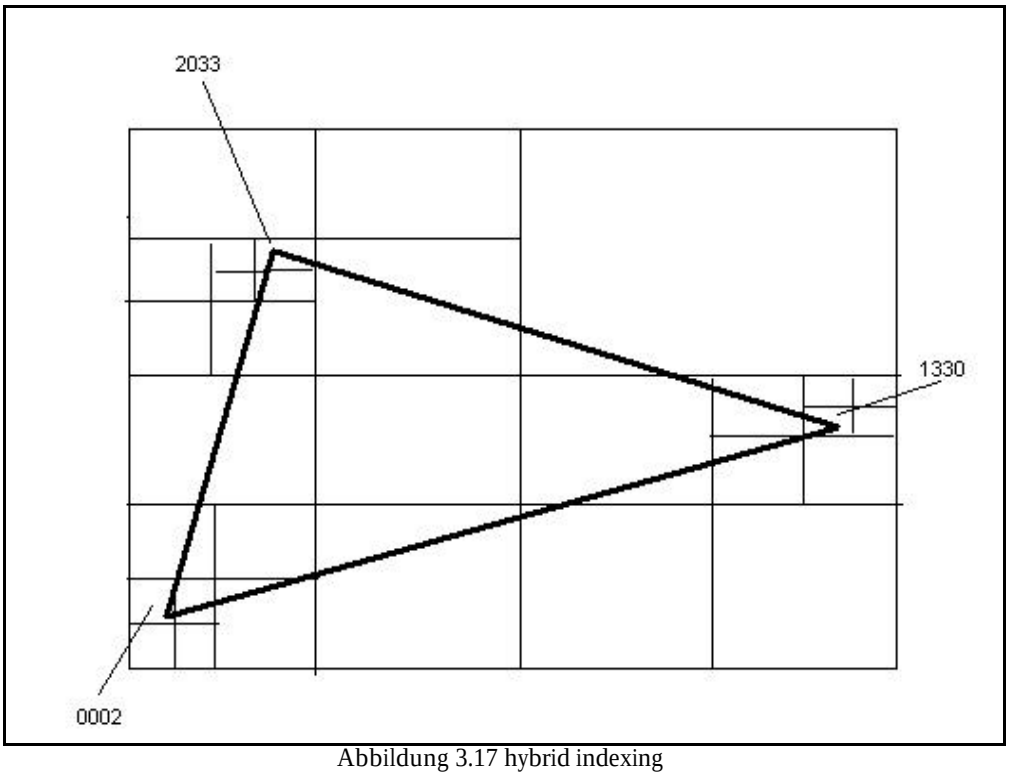


Abbildung 3.17 hybrid indexing

In Abbildung 3.17 ist zu erkennen, dass die Rasterung nur an den Punkten durchgeführt wird, an denen ein Endpunkt der Figur liegt. Die Felder werden bestimmt, wobei die Anzahl der Ziffern von der Genauigkeit herrührt. Um z.B. den Punkt in Feld 0002 genauer zu beschreiben, würde das Feld wieder in vier Quadrate geteilt werden und die Bestimmung würde dann 00021 ergeben.

3.7 Weitere Aspekte der Spatial Technologie

Die Oracle Spatial Technologie beinhaltet weitere verschiedenste Aspekte, die hier nur kurz erwähnt und nicht weiter betrachtet werden sollen.

- ?? Anfragemodell für räumliche Anfragen
- ?? Funktionen und Operatoren des Cartridge Oracle Spatail
- ?? Anfrageoptimierung

Im nachfolgenden Kapitel soll nun gezeigt werden, wie es mittels Java möglich ist, auf Datenbanken zuzugreifen und die gespeicherten Informationen über Java aufzubereiten und darzustellen.

3.8 Was bewirkt die Spatial Technologie

Durch den Einsatz der Spatial Technologie ändert sich am Datenmodell zunächst nichts. Die Speicherung der Daten wird jedoch effektiver unterstützt. Was bedeutet das ? Zunächst einmal wird ein Punkt zumindestens zweidimensional abgespeichert, d.h. ein x-Wert und ein y-Wert. Interessanter sind die Möglichkeiten Landesgrenzen mittels der Spatial Technologie in Datenbanken abzulegen. Das Land, dessen Grenzen abgespeichert werden sollen, wird als Polygon betrachtet. Beginnend an einem beliebigen Punkt können nun die Aussengrenzen des Landes mittels des Polygons, welches mittels der Spatial Technologie beschreibbar wird, in die Datenbank abgelegt werden. Wenn nun die Daten abgerufen werden, können mit deren Hilfe die Aussengrenzen des Landes gezeichnet werden. Desweiteren ist es möglich, Städte als Punkte in der Fläche des Polygons abzulegen; Straßen und Flüsse wären als Linienzüge ablegbar. So wird es möglich, die gesamten Informationen die in einer Karte enthalten sind, in eine Datenbank abzulegen.

Außerdem gewinnt man die Möglichkeit, die Daten die auf oben beschriebene Weise in Datenbanken abgelegt sind, mit weiteren Daten, z.B. Grafiken und Bilder, die ebenfalls in Datenbanken abgelegt sind zu kombinieren. Betrachtet man das Beispiel der Brunnenkarte aus dem Kapitel 2.4, so kann es möglich werden, dass ein Satelitenfoto einer Region als Blob in einer Datenbank abgelegt wird und die Brunnen dann als Punkte mittels der Spatial Technologie gespeichert

werden. Sämtliche Umrechnungen, die für das verändern des Maßstabes von Nöten wären, müssten jedoch noch näher betrachtet werden.

Aber auch für andere Einsatzgebiete bietet die Spatial Technologie die Möglichkeit Daten in Datenbanken abzulegen, so wird es z.B. möglich technische Zeichnungen entsprechend aufzuschlüsseln und so in Datenbanken abzulegen.

Die Spatial Technologie stellt eine interessante Möglichkeit dar geometrische Informationen als zusammengehörige Objekte in Datenbanken abzulegen, ohne diese als Grafiken zu speichern.

Kapitel 4

Technologien mittels Java auf Datenbanken zuzugreifen

In diesem Kapitel werden zwei Technologien beispielhaft vorgestellt, mit denen es möglich wird auf Datenbanken zuzugreifen, die der Nutzer sofort - ohne weiteres Hintergrundwissen - in die Lage versetzen die Datenbank zu nutzen.
Dabei wird in beiden Möglichkeiten der Datenbankzugriff mittels JDBC durchgeführt. Im ersten Beispiel wird jedoch die Benutzeroberfläche mittels SWING implementiert. Im zweiten Beispiel werden Java Server Pages verwendet, dies wird jedoch nur skizenhaft aufgezeigt.
Für beide Technologien ist es wichtig, dass man die Systemarchitektur und den grundlegenden Ablauf des Zugriffs auf das Datenbanksystems klar vor Augen hat. Aus diesem Grund wird hier der Einstieg in dieses Kapitel über die Systemarchitekturen gewählt. Daran schließt sich eine kurze Betrachtung über den prinzipiellen Ablauf einer Datenbankabfrage an, damit die Abläufe klar werden. Mit einer Betrachtung der vier Treibertypen wird diese kurze Zusammenfassung der Grundlagen beendetn um dann den praktischen Einsatz anhand der Beispiele zu verdeutlichen..

4.1 Systemarchitekturen

Prinzipiell findet man bei den Systemarchitekturen zwei Modelle. Zum einen die 2-Schichten-Architektur (two-tier-systems) und zum anderen die 3-Schichten-Architektur (three-tier-systems).
Bei beiden Systemen handelt es sich um sogenannte Client-Server-Architekturen, d.h. der Client stellt eine Anfrage (ggf. über ein Netzwerk) an den Server, dieser bearbeitet die Anfrage und sendet die Antwort dann wieder über das Netzwerk zum Client zurück. Dies setzt die 2-Schicht-Architektur auch direkt so um. Der Client kommuniziert direkt mit der Datenbank (siehe Abbildung 4.1).

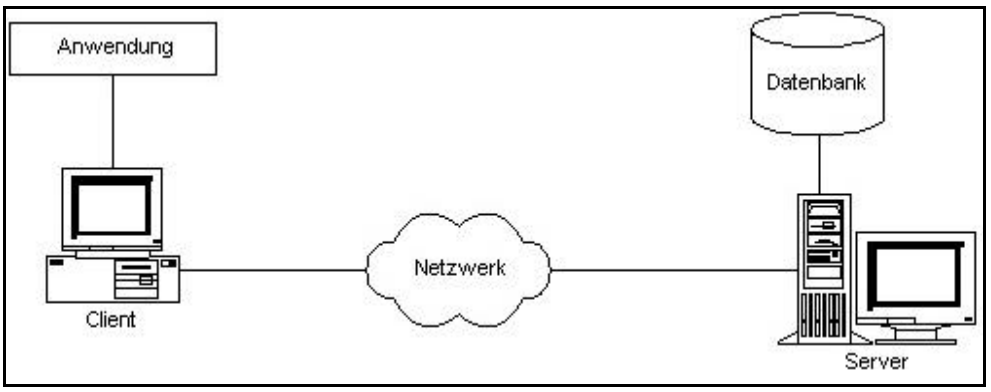


Abbildung 4.1 – 2 – Schichten Architektur

Aus welchen Gründen wählt man die 2-Schichten-Architektur ?

Die 2-Schichten-Architektur ist leicht zu implementieren, so dass man schnell einmal etwas testen kann. Ein weiterer Vorteil ist, dass sie in der Regel schneller als eine 3-Schichten-Architektur ist. Dies spielt z.B. bei Antwortzeiten eine entscheidende Rolle. Der Nachteil dieser Architektur liegt in der Sicherheit. Da Datenbank und WebServer auf einem Rechner laufen, der dann ggf. aus dem Internet (wenn dieses als Netzwerk eingesetzt wird) sichtbar ist.

Um diese Sicherheitslücke zu schließen, wurde die 3-Schicht-Architektur entwickelt. Hierbei werden die Aufgaben des Servers auf zwei Rechner verteilt. Auf einem Rechner wird die Datenbank betrieben und um diese zu schützen wird der zweite Rechner zwischen den Datenbankrechner und das Netzwerk geschaltet. Er wird dann als Anwendungs-Server bezeichnet und der Datenbankrechner als Datenbank-Server (siehe Abbildung 4.2).

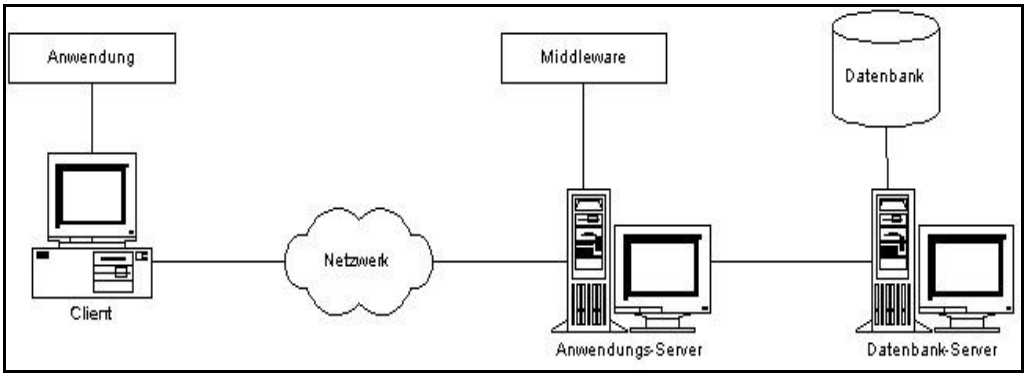


Abbildung 4.2 – 3-Schichten-Architektur

Wie in der Abbildung zu erkennen ist, wird auf dem Anwendungs-Server eine sogenannte Middleware verwendet. Diese Middleware sorgt für die Verbindung

zwischen dem Datenbank-Server und dem Netzwerk, bekannt als Middleware ist z.B. CORBA.
Durch den Einsatz der Middleware ist die Sicherheitslücke, die die 2-Schichten-Architektur beinhaltet geschlossen worden, dadurch treten jedoch längere Laufzeiten für Anfragen des Clients zur Datenbank auf. Ein weiterer Vorteil ist die klare Trennung in die 3-Schichten, ähnlich dem Modell – View – Controller Konzept. Die Präsentation, vergleichbar mit dem View, finden man auf dem Client wieder. Geschäfts-Logik die auf dem Anwendungs-Server umgesetzt wird ist vergleichbar mit dem Controller, der Modell und View miteinander verbindet. Der Datenbank-Server hält die darzustellenden Daten vor und liefert auf Anforderung diese dem Client zur Darstellung..

4.2 Prinzipieller Ablauf einer Datenbankabfrage mittels JDBC

Die Datenbankabfrage mittels JDBC gliedert sich in ihrem Ablauf in fünf Stationen. Als erstes muss der Datenbank-Treiber geladen werden, mit dessen Hilfe dann eine Verbindung zur Datenbank aufgebaut wird. Nachdem dann die Verbindung fehlerfrei aufgebaut wurde, wird in der zweiten Phase das SQL-Statement erzeugt. Diese Statement wird dann in der dritten Phase der Datenbank zur Ausführung übergeben. Wie allgemein bekannt ist, löst jedes SQL-Statement in der Datenbank eine Reaktion aus. Diese Reaktion zu verarbeiten ist dann Aufgabe der vierten Phase, in welcher die Ergebnismenge ausgewertet wird. Die letzte Phase in diesem Ablauf stellt das Schließen der Verbindung dar.
Betrachten wir nun die einzelnen Phasen in der Hinsicht etwas genauer, welchen Code wir hier implementieren und welche Probleme jeweils auftreten können.

1.Phase : DB-Treiber laden und DB-Verbindung herstellen.

Als erstes muss der Datenbanktreiber geladen werden. Dazu muss man diesen zunächst auf dem System identifizieren. Bei Oracle - der hier in den Beispielen verwendeten Datenbank - finden wir diesen in folgendem Unterverzeichnis : oracle/ora81/jdbc/lib/ . Hier sind mehrere Zip - Dateien zu finden.
Unser Interesse richtet sich auf die Dateien Classes111 und Classes12. Wir verwenden Classes12 da wir mit dem SDK1.3.1, oder später, arbeiten. Classes111

wird verwendet, wenn man ein JDK1.1 oder ein wenn Applets programmiert, die noch mit einem 1.1 Java lauffähig sein sollen und auch wenn noch ältere Runtime - Umgebungen verwendet werden sollen.
So finden wir unsere erste Codezeile zur Datenbankanbindung :

```
Class.forName("oracle.jdbc.driver.OracleDriver");
```

Da hierbei der Fehler auftreten kann, dass der Treiber nicht gefunden wird und diese auch als Exception auftreten kann, muss das Laden des Treibers mit der dazugehörgen Exception in einen try und catch – Block gefasst werden. Dabei handelt es sich um eine „Class Not Found Exception“.
Des weiteren muss als nächstes die Verbindung zur Datenbank hergestellt werden. Dazu benötigen wir die Informationen der URL, des Users und das dazugehörige Passwort. User und Passwort sind dieselben, die man auch zur Nutzung einer Datenbank mittels SQL benötigt.
Von Bedeutung ist noch die URL. Wie kommen wir an diese ? In einem größeren System, das von einem Administrator verwaltet wird, wenden wir uns an diesen und erhalten unsere gewünschte Information. Wie sieht es in Systemen aus, wo wir selbst der Administrator sind, bzw. wie setzt sich die URL zusammen ? Dazu erst einmal eine typische, später in den Beispielen verwendetet URL : "jdbc:oracle:thin:@localhost:1521:rsz".
Was ist an dieser URL zu beachten? Zunächst einmal der Rechnername. Dieser wird nach dem @ angegeben und ist in dieser Testumgebung localhost. Er kann aber auch eine IP-Nummer sein oder ein Rechnername zu einer zugehörigen IP-Nummer. Danach ist noch der Port angegeben, auf den der Listener der Datenbank ‚hört'. 1521 ist dabei der Port der automatisch bei einer Standartinstallation von Oracledatenbanken eingerichtet wird.. Es kann aber auch vorkommen, dass ein ganz anderer Port einem Listener zugeordnet wurde. Diese muss dann in der Konfiguration der Datenbank nachgeschaut werden.
Nach dem Port wird noch die SDI der Datenbank angegeben (in diesem Beispiel rsz).
Auch hierbei können wieder Fehler auftreten, wie z.B. das eine falsche URL, User oder Passwort angegeben wird, so dass auch hier wieder die Anweisung in einen try- und catch-Block gefasst werden muss. Die Exception die auftreten kann ist eine SQL-Exception.

Damit ist die erste Phase beendet und wir kommen zur zweiten Phase.

2. Phase : SQL-Statement erzeugen

In dieser Phase muss das zu erzeugende Statement mit den Daten zum Zugriff auf die Datenbank verbunden werden. Dies ist eine vorgegebene Aktion, die nicht weiter besprochen werden muss. Der zu erzeugende Code beinhaltet hauptsächlich folgende Codezeile :

```
v_statement = v_connection.createStatement();
```

wobei v_connection ein Objekt der Verbindung zur Datenbank darstellt. Hierbei können keine weiteren Probleme auftreten, wenn in Phase 1 alles richtig lief.
In der dritten Phase wird das Statement ausgeführt.

3. Phase : SQL-Statement ausführen

In dieser Phase wird das Statement an die Datenbank zur Ausführung übergeben. Hierbei ist darauf zu achten, dass wenn z.B. eine select - Abfrage getätigt wird, auch die entsprechend richtige Syntax verwendet wird. Da es sich um SQL-Anweisungen handelt, kann auch hier wieder eine SQL-Exception auftreten, so dass diese Anweisung auch wieder in einen try- und catch-Block gefasst werden muss. Bei der Implementierung ist vor allem darauf zu achten, dass die SQL-Anweisung am Ende als ein gültiger String der Datenbank zur Ausführung übergeben wird, da es sonst zu Fehlern kommen kann, die einige Verwunderungen hervorrufen können.

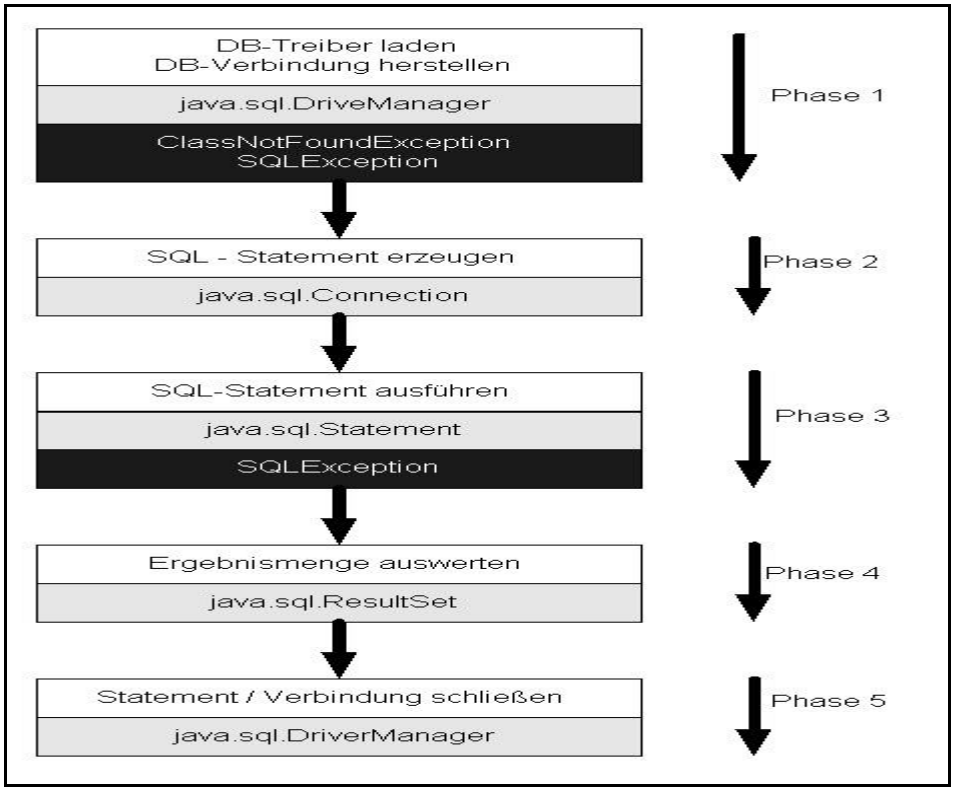
4. Phase : Ergebnismenge auswerten

In dieser Phase ist vor allem darauf zu achten, das Ergebnismengen auftreten können, die nicht beabsichtigt sind, diese müssen dann entsprechend abgefangen und bearbeitet werden. Die richtigen Ergebnisse müssen aber auch zur Weiterverarbeitung entsprechend bearbeitet werden.

5. Phase Statement / Verbindung schließen

Abschließend muss dann noch das Statement, das in Phase 2 geöffnet wurde, geschlossen werden, um dann als letztes die Verbindung zur Datenbank aus Phase 1 zu schließen.

Eine Übersicht über den Ablauf sowie die verwendeten Klassen und die auftretenden Exceptions ist in Abbildung 4.3 zu finden.



4.3 - Ablauf JDBC Anbindung

4.3 Typen von JDBC – Treibern

In Java findet man 4 Typen von JDBC – Treibern, die eng mit der Entwicklung des JDBC's verbunden sind. Bei Sun waren sich die Entwickler von Java sehr schnell bewusst, welches Potential in Java verborgen liegt und welche Möglichkeiten damit verbunden sind, d.h. die plattformunabhängige Entwicklung und auch die Vernetzung verschiedenster Rechnertypen. Da es zur Zeit der Entwicklung bereits ODBC gab und dieses auch im Bereich von C/C++ Programmierung eingesetzt wurde und wird, wollte man natürlich dieses nutzen.

ODBC ist jedoch schwer zu lernen und ODBC stützt sich auch auf Pointer, die es so in Java nicht gibt. Also musste eine Möglichkeit gefunden werden, eine Datenbankanbindung mittels ODBC und Java zu realisieren. So entwickelten sich die Typ 1-Treiber (weiteres siehe unten). Eine andere Möglichkeit war es über den nativen Code mit der Datenbank zu kommunizieren. Damit geht aber die Plattformunabhängigkeit verloren. Dennoch wurde dies in den Typ 2-Treibern realisiert. Type 3-Treiber und Typ 4-Treiber entwickelten sich etwas später, da diese speziell auf und mit Java zugeschnitten wurden.
Eine Übersicht über die Treiber findet man in [HC 99/2] : „

?? Ein Typ 1-Treiber übersetzt JDBC nach ODBC und stützt sich auf einen ODBC-Treiber, um mit der Datenbank zu kommunizieren. Sun bietet im JDK mit der JDBC/ODBC –Brücke einen derartigen Treiber an. Allerdings unterstützt die Brücke nicht JDBC2 und erfordert einen geeignet konfigurierten ODBC-Treiber. Die Brücke ist vor allem praktisch um Datenbankverbindungen zu testen, empfiehlt sich aber nicht für den Produkteinsatz.
?? Ein Typ 2-Treiber ist zum Teil in Java geschrieben und stützt sich auf nativen Code, der mit der Client – API einer Datenbank kommuniziert. Wenn man mit einem derartigen Treiber arbeitet, muss man neben einer Java-Bibliothek auch plattformspezifischen Code installieren.
?? Ein Typ 3-Treiber ist eine reine Client-Bibliothek in Java, die über ein datenbankunabhängiges Protokoll Datenbankanforderungen an eine Serverkomponente schickt, die wiederum die Anforderungen in ein datenbankspezifisches Protokoll umsetzt. Die Client-Bibliothek ist unabhängig von der eigentlichen Datenbank und vereinfacht somit die Entwicklung.
?? Ein Typ 4-Treiber ist eine reine Java-Bibliothek, die JDBC-Anforderungen direkt in ein datenbankspezifisches Protokoll übersetzt.“

In Abbildung 4.4 sind noch einmal die 4 Typen dargestellt. Des weiteren ist die Kommunikation von der Anwendung über die JDBC-API über den Treibermanager, der wiederum auf die Treiber -API zugreift, dargestellt. Die Treiber – API kommuniziert dann mit den einzelnen Treibertypen, die dann die

Verbindung zur Datenbank herstellen. Die Rückkommunikation erfolgt dann auf dem umgekehrten Wege.

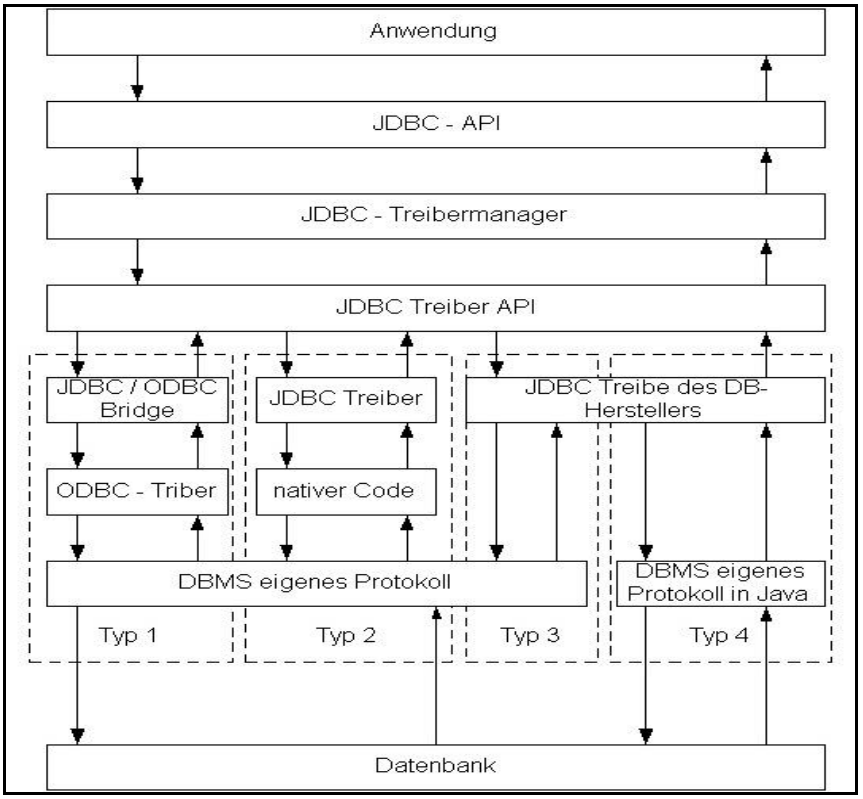


Abbildung 4.4 – JDBC – Treibertypen und Kommunikation Anwendung – DB

Nachdem nun einige grundlegende Ding über JDBC noch einmal zusammengetragen wurden, werden wir nun zu einer ersten Anwendung kommen.

4.4 Datenbankzugriffe unter Nutzung von Swing

Anhand einer Beispiel Implementierung soll nun gezeigt werden, wie es möglich ist, auf Datenbanken mittels Java zuzugreifen. In diesem ersten Beispiel wird die Benutzeroberfläche mittels Swing gestaltet. Zur Entwicklung wurde NetBeans 3.3.1 mit dem SDK 1.4 eingesetzt. Dieses Programm ist aber auch unter SDK 1.3 lauffähig und als Datenbank Oracle 8i in der Version 8.1.6 eingesetzt.

Was soll konkret Implementiert werden ?

Nach dem Starten der Anwendung kommt der User in ein Anmeldefenster, in dem er sein Usernamen und sein Passwort eingibt. Diese werden in der

Datenbank auf ihre Gültigkeit überprüft. Liegt ein Fehler vor, wird eine Meldung ausgegeben und die Anwendung beendet. Wird das Passwort oder der Username nicht angegeben, wird der User darauf aufmerksam gemacht und er kann die Anmeldung vervollständigen. Ist die Anmeldung korrekt vollzogen worden, gelangt der User in das Hauptmenu, in dem er alle Tabellen der Datenbank angezeigt bekommt, auf die er zugreifen darf. Hier wählt der User eine mittels Mausklick aus und kann dann sich die Daten ansehen oder Daten eingeben oder den Tabellenaufbau ansehen. Natürlich kann hier auch die Anwendung beendet werden. Falls der User noch keine Tabelle ausgewählt hat, aber eine der Aktionen ausführen möchte, wird er darauf hingewiesen, dass er erst eine Tabelle auswählen muss. Für die einzelnen Aktionen werden dann wieder separate Fenster geöffnet, in denen dann die Aktionen ausgeführt werden. Die einzelnen Komponenten greifen dann wiederum auf die Datenbank zu, um sich die Informationen zu holen die sie benötigen.

Auf was ist zu achten ?

Beim Zugriff auf die Datenbank muss in der Komponente Anmeldung die URL bekannt sein, Username und Passwort werden hier eingegeben. Alle drei Parameter, URL, User und Passwort müssen dann für die Hauptauswahl zur Verfügung stehen, damit hier die Tabellen ausgelesen werden können. Für die weiteren drei Komponenten „Daten eingeben“, „Daten ansehen“, und „Tabellenaufbau ansehen“ müssen dann alle vier Parameter (URL, User, Passwort, ausgewählte Tabelle) bekannt sein (siehe hierzu auch Abbildung 4.5).

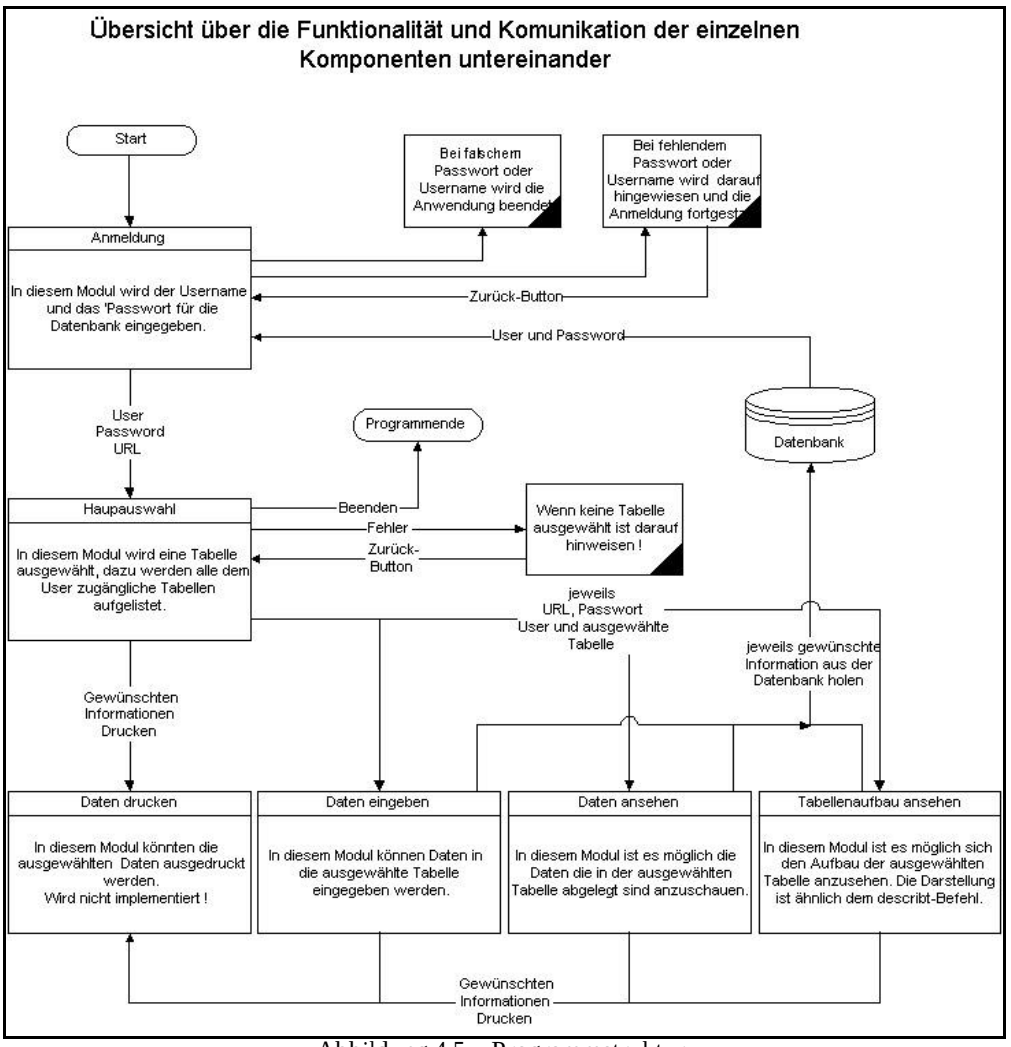


Abbildung 4.5 – Programmstruktur

4.6 Weitere Entwicklung des Tools

Eine weitere Entwicklung für das Tool wäre eine zusätzliche Datenbank anzubinden. In dieser Hinsicht wurde bereits im DB-Starter ein zusätzlicher Button aufgenommen, mit dem zwischen den DBS Oracle und MySQL gewählen werden kann. Der MySQL Pfad ist noch nicht ausgebaut.

Weiter wäre es möglich, dass ein View mit aufgenommen wird. Dieser View wäre z.B. für die Tabelle Fluss zu erstellen. Es würde in einem zusätzlichen Fenster ein Fluss dargestellt werden, auf dem die Messpunkte dargestellt sind. Durch Anklicken des Messpunktes könnten die Daten aufgerufen werden, die für diesen entsprechenden Messpunkt zur Verfügung stehen würden. Dieser View wäre dann natürlich sehr stark mit den entsprechenden Tabellen verknüpft und würde

nur die Möglichkeiten zeigen, welche Darstellungsmöglichkeiten mit den entsprechenden Daten prinzipiell möglich wären.
Über einen weiteren Ausbau des Tools in Bezug auf Management von Benutzern und Rollen wäre nachzudenken. Dies würde zeigen, was und wie dies mit Java zu realisieren ist, für den effektiven Einsatz sei jedoch auf die Tools verwiesen, die mit Oracle ausgeliefert werden, so z.B. Enterpreis Manager Console.

4.7 Java Server Pages

Eine weitere Technik mit deren Hilfe Datenbankinhalte für das Internet aufbereitet werden können soll hier kurz betrachtet werden und wird ansonsten nicht weiter verwendet. Java Server Pages (JSP) basieren auf der Servlettechnologie, sind jedoch vom Aufbau noch sehr stark an HTML angelehnt. Dies bedeutet, das dass gesamte Layout mittels HTML gestaltet wird. Diese Verbindung von Java und HTML bietet einige Vorteile die hier kurz aufgeführt werden sollen.

- ?? Layout durch HTML generierbar, d.h. der Layoutdesigner muss keine Java Kenntnisse besitzen.
- ?? Durch die Einbettung von Java in die HTML Seiten können alle nur erdenkbaren Logiken mittels Java erstellt werden, inklusive von Datenbankzugriffen mittels JDBC.
- ?? Zugriff auf weitere Java API's wie z.B. RMI, JNDI, CORBA, JNDI, EJB, JMS, JTA
- ?? Dynamisch darzustellende Inhalte für Internetseiten.
- ?? Nutzung weiterer Vorteile wie z.B. Sesson Tracking.

Um Java Serverpages nutzen zu können, ist es unabdingbar, dass ein Webserver eingerichtet wird. Dazu kann man beispielsweise Apache Tomcat aus dem Jakarta Projekt nutzen.
Auf das Anfrage- und Antwort-Modell von http soll hier nicht weiter eingegangen werden. Dazu sei verwiesen auf entsprechende Literatur [Ber 01] und [Hal 01].

Die Technik der JSP beruht auf der Technologie der Servlets. Dies bedeutet, dass aus den JSP Seiten beim kompilieren Servlets entstehen, die dann auf dem Server ausgeführt werden.
Betrachten wir ein einfaches Servlet, das die aktuelle Uhrzeit ausgibt. Dazu ist folgender Programmtext auf dem Server abzulegen :

```
public class TimeNow implements Servlet {
    public void service (ServletRequest req, ServletResponse res)
                        throws ServletException, IOException {
            res.setContentType("text/html");
            PrintWriter out = res.getWriter();
            out.println("<html>");
            out.println("    <head>");
            out.println("    <title> Time Now</title>");
            out.println("    </head>");
            out.println("     <body>");
            out.println("      Es ist : " + (new java.util.Date().toString()) );
            out.println("      </body>");
            out.println("</html>);
     }
}
```

Wie schon an diesem kurzen Beispiel zu erkennen ist, ist es sehr aufwendig Servlets von Hand zu programmieren, da sämtliche HTML Befehle jeweils in println-Zeilen ausgedrückt werden müssen. Wenn man nun dasselbe Beispiel als JSP programmiert, wird die Funktionalität des Javaprogramms in die HTML Seite eingebettet. Der Code der JSP Seite sieht folgendermassen aus :

```
<html>
   <head>
      <title> Time Now </title>
    </head>
    <body>
      Es ist : <%= new java.util.Date().toString() %>
     </body>
</html>
```

Diese wenigen Codezeilen bieten dieselbe Funktionalität, wie das Servlet bzw. werden in den entsprechenden Servletcode umgewandelt.

Wie zu erkennen ist, kann das gesamte Layout der Internetseite mittels HTML gestaltet werden. Es ist natürlich auch möglich, Tabellen zu erstellen und mittels Java und einer Datenbank diese mit Informationen zu füllen. Dabei ist wiederum darauf zu achten, dass man auf die Datenbank, wie man bereits oben gesehen hat, in der festgelegten Reihenfolge zugreift. Für entsprechende Beispiele wird auf die Literatur verwiesen, so z.B. [Ber 01] S.171 ff.

ArcInfo und ArcSDE

In diesem Kapitel sollen die Produkte ArcInfo und ArcSDE von Esri kurz betrachtet werden und dargestellt werden, wie diese Programme miteinander arbeiten. Über die Installation der Produkte sind entsprechende Informationen im Anhang zu finden.

5.1 ArcInfo

ArcInfo setzt sich aus drei Tools zusammen. Das Haupttool stellt eine Art Explorer zur Verfügung, mit dessen Hilfe die weiteren Tools angewählt werden können. Das Haupttool ist ArcCatalog. Hier werden sämtliche Einstellungen für die weitere Arbeit getätigt. In ArcCatalog findet man auch später die Anbindung an die entsprechenden Datenbanken. Hier werden auch weitere zusätzliche Tools und Funktionalitäten eingebunden, so z.B. ArcIMS (Internet Map Server) der dann mit entsprechenden Servern zusammenarbeitet und die gewünschten Informationen aus der angebundenen Datenbank holt.

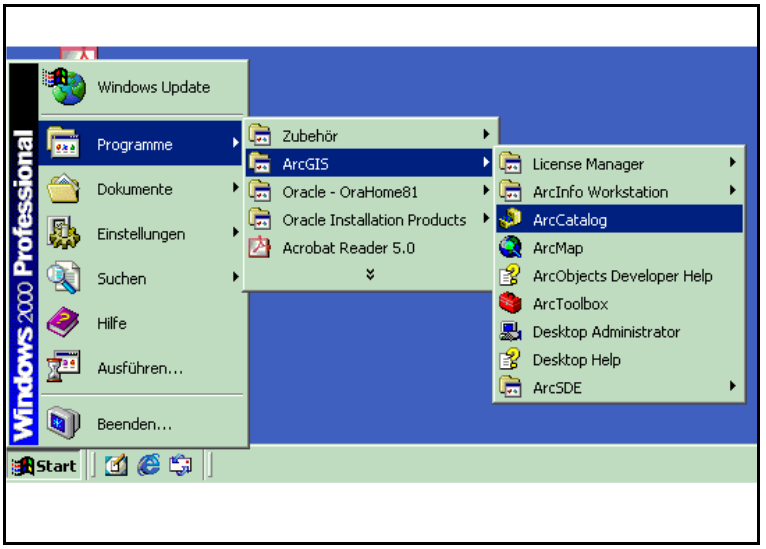


Abbildung 5.1 ArcGIS

Nachdem ArcCatalog gestartet wurde, zeigt sich der Explorer mit entsprechenden Wahlmöglichkeiten. Hier können dann auch die weiteren Tools gestartet werden (siehe Abbildung 5.1 und 5.2).

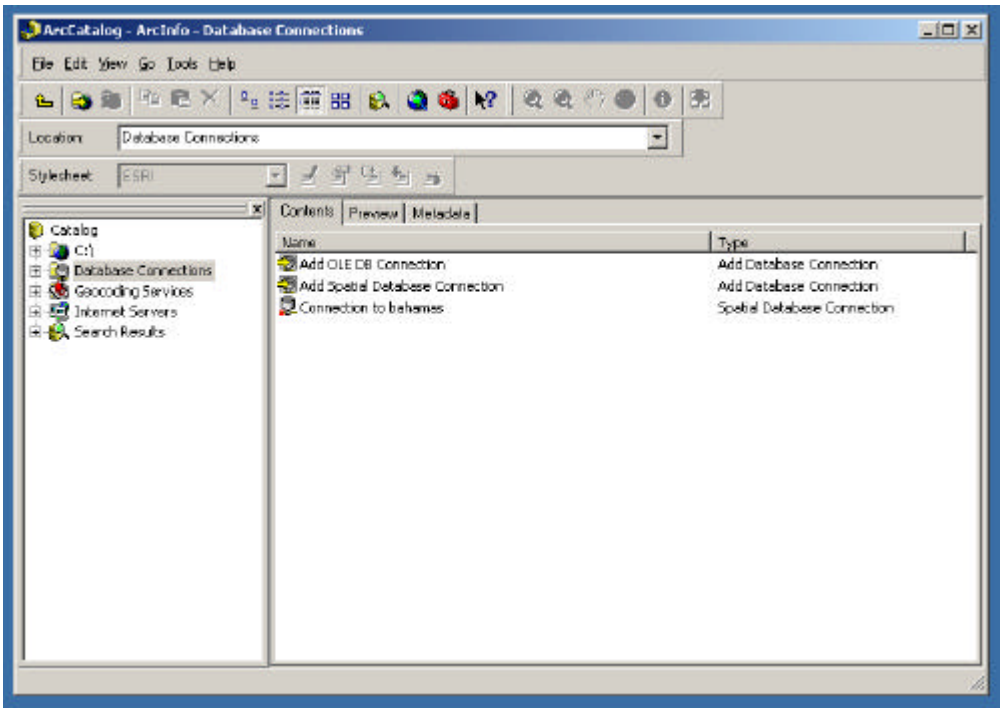


Abbildung 5.2 ArcCatalog

Die weiteren Tools sind :

?? ArcMap

Mit diesem Tool ist man in der Lage, Karten zu erstellen und verschiedene Operationen mit diesen Karten auszuführen (siehe Abbildung 5.3).

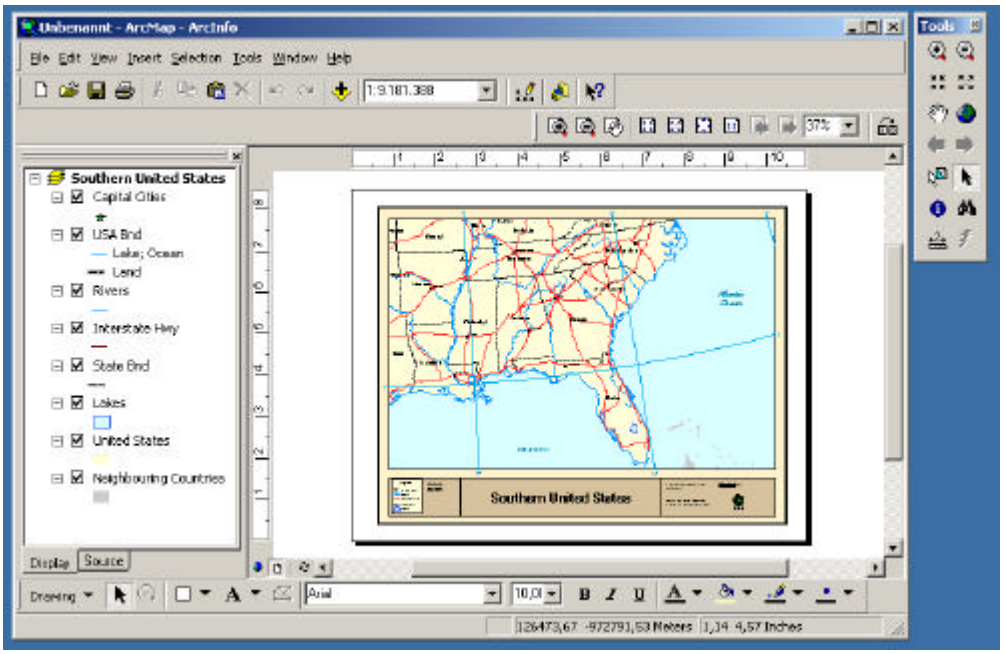


Abbildung 5.3 ArcMap

?? ArcToolbox

Dieses Tool stellt einen Werkzeugkasten dar, in dem verschiedenste Funktionen abgelegt sind (siehe Abbildung 5.4).

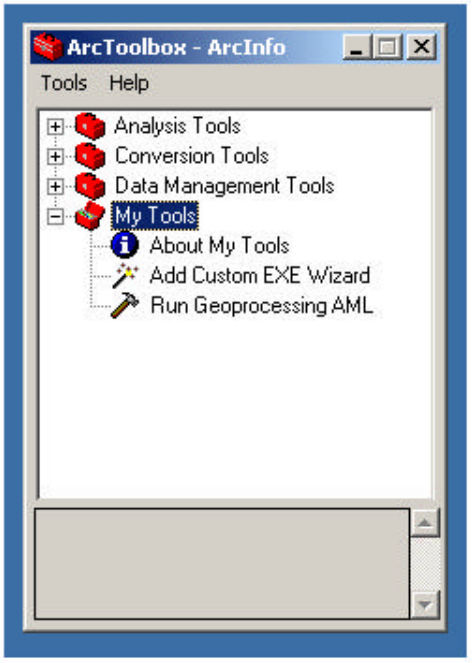


Abbildung 5.4 ArcToolbox

5.2 ArcSDE

ArcSDE stellt einen Application Server dar, mit dem es möglich ist, dass die verschiedensten Produkte der Esri-Familie auf Datenbanken zugreifen können (siehe Abbildung 5.5).

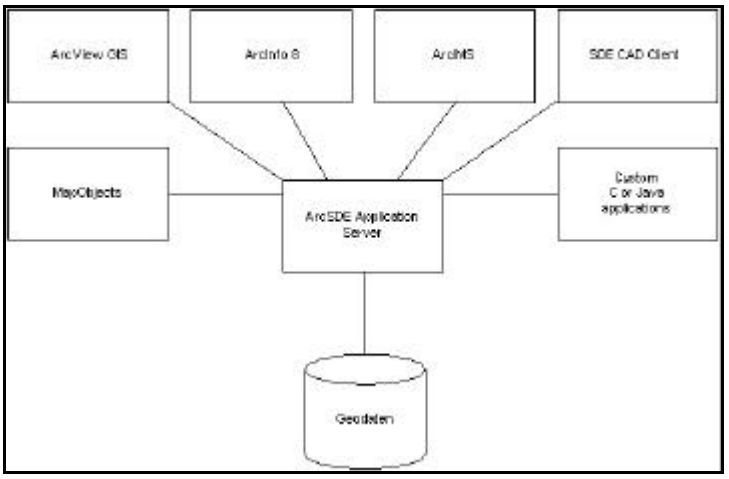


Abbildung 5.5 ArcSDE

Um ArcSDE richtig nutzen zu können, muss die Verbindung zur Datenbank hergestellt werden. Diese Verbindung wird in Arccatalog unter dem Menüpunkt “Database Connection“ eingerichtet. In unserem Fall wurde eine Verbindung zu einer Oracle 8i Datenbank aufgebaut. Bei korrekter Einrichtung wird die Datenbankverbindung mit entsprechend gefundenen Tabellen in ArcCatalog angezeigt (siehe Abbildung 5.6).

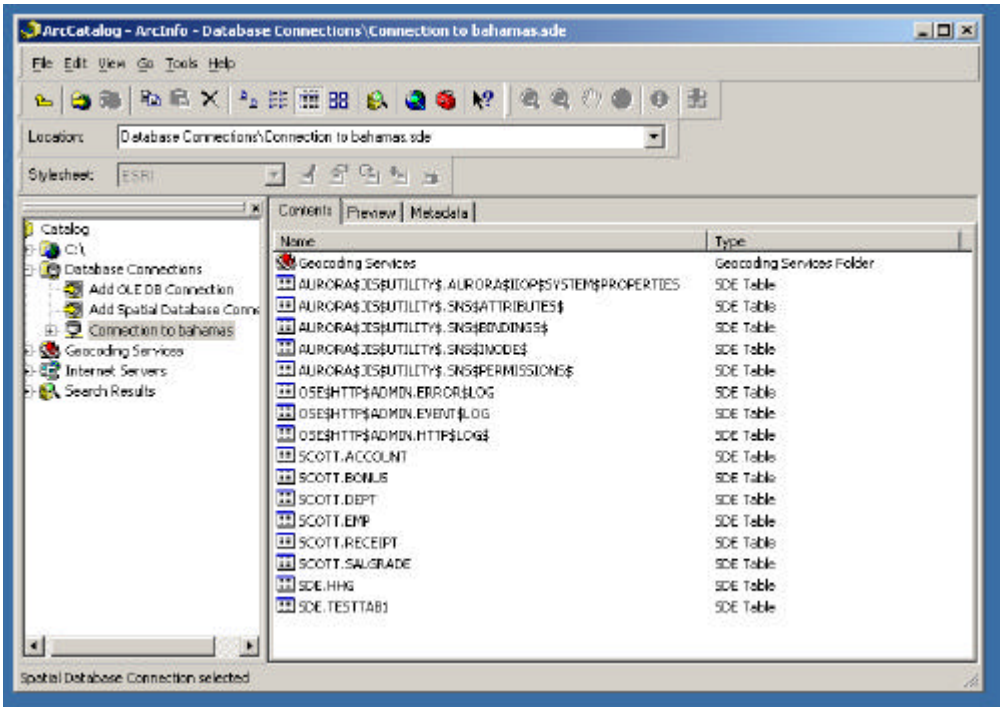


Abbildung 5.6 Datenbankconnect

ANHANG

A 1 Literaturverzeichnis

[ACA 00] Oracle 8i für Einsteiger
Abbey / Corey / Abramson
Verlag : Hanser
ISBN : 3 – 446 – 21432 – 1

[ALG 00] Skript zur Vorlesung Algorithmen III an der FH Köln
gelesen von Herrn Prof. Heiner Klocke

[Bar 95] Geoinformatik – Modelle, Strukturen, Funktionen
Bartelme
Verlag : Springer
ISBN : 3 – 540 – 65988 – 9

[Ber 01] Java Server Pages
Hans Bergsten
Verlag : O'Reilly
ISBN : 3 – 89721 – 281 – 1

[Bil 99/1] Grundlagen der Geo-Informationssysteme
Band 1 : Hardware, Software und Daten
Ralf Bill
Verlag : Wichmann
ISBN : 3 – 87907 – 325 – 2

[Bil 99/2] Grundlagen der Geo-Informationssysteme
Band 2 : Analyse, Anwendungen und neue Entwicklungen
Ralf Bill
Verlag : Wichmann
ISBN : 3 – 87970 – 326 – 0

[BMR 98] Patternorientierte Softwarearchitektur
Buschmann / Meunier / Rohnert / Sommerlad / Stal
Verlag : Addison-Wesley
ISBN : 3 – 8273 – 1282 – 5

[BW 01] GIS-Report 2001
Software Daten Firmen
Buhmann / Wiesel
Verlag : Harzer
ISBN : 3 – 9803128 – 7 – 9

[DBS 00] Skript zur Vorlesung Datenbanken und Informationssysteme
an der FH Köln gelesen von
Frau Prof. Dr. Heide Faeskorn – Woyke

[ESK 96/1] Graphische Datenverarbeitung 1
Encarnacao / Strasser / Klein
Verlag : Oldenbourg
ISBN : 3 – 486 – 23223 – 1

[Hal 01] Servlets und Java Server Pages
Marty Hall
Verlag : Markt und Technik
ISBN : 3 – 8272 – 5945 - 2

[HC 99/1] Core Java 2
Band 1 - Grundlagen
Horstmann / Corell
Verlag : Markt und Technik
ISBN : 3 – 8272 – 9565 – 3

[HC 99/2] Core Java 2
Band 2 – Expertenwissen
Horstmann / Corell
Verlag : Markt und Technik
ISBN : 3 – 8272 – 9566 – 1

[Loc 97] Oracle 8 Datenbankentwicklung
David Lockman
Verlag : Markt + Technik
ISBN : 3 – 8272 – 2017 – 3

[Mas 01] MySQL in 21 Tagen
Mark Maslakowski
Verlag : Markt + Technik
ISBN : 3 – 8272 – 5850 – 2

[MS 99] Java Programmierhandbuch und Referenz
Middendorf / Singer
Verlag : dpunkt
ISBN : 3 – 920993 – 82 – 9

[Sed 99] Algorithmen in C++
Robert Sedgewick
Verlag : Addison – Wesley
ISBN : 3 – 89319 – 462 – 2

Allgemein genutzte Literatur ohne weiteren Verweiss :

Computer Fachlexikon / Fachwörterbuch
Ausgabe 2000
Verlag : Microsoft Press
ISBN : 3 – 86063 – 822 – X

A 2 Internetseiten

Allgemein

http://www.gm.fh-koeln.de/

http://www.dokuwelt.de/

Java

http://java.sun.com/

http://www.javabuch.de/

http://javaforum.breed.de/

http://www.javamagazin.de/

MySQL

http://mmmysql.sourceforge.net/

Oracle

http://www.gm.fh-koeln.de/faeskorn/

http://www.oracle.com/de/

A 3 Verwendete Software

?? Oracle 8i Version 8.1.7 Enterpreise Edition

?? ArcInfo

?? ArcSDE 8.1

?? Java SDK 1.4

?? NetBeans 3.3.1

?? ERWIN

?? TOAD

B SoftwareInstallationen

Bei der Installation der Software ist auf eine feste Reihenfolge zu achten. Diese ist :

1. Oracle 8i Version 8.1.6 oder 8.1.7
2. Lizensemanager für Esri-Produkte
3. Lizensemanager konfigurieren
4. Server neu booten
5. ArcInfo Server installieren
6. Lizensemanager bestätigen
7. ArcInfo Client installieren
8. Lizensmanager bestätigen
9. ArcSDE Server installieren

9.1 anlegen des Users SDE in Oracle, Zuweisung der entsprechenden Rechte : DBA, RESORCE und CONECT

9.2 mitgeliefertes SQL Skript starten, vorher entsprechende Passwörter eintragen

9.3 ArcSDE Server installieren

10. Lizensemanager bestätigen
11. Server neu booten
12. Service von ArcSDE löschen und neu konfigurieren
13. Server neu booten

Hierbei fällt auf ,dass der Lizensemanager häufig bestätigt werden muss. Dies sollte man bei der Installation konsequent durchführen, da es sonst zu enormen Problemen kommen kann. Ebenso das Booten des Servers vor und nach dem der Service von ArcSDE gelöscht und neu konfigurirt wurde.

Achtung : ArcSDE 8.1 arbeitet nicht mit Oracle 9i zusammen. Dies bedeutet weiter, dass ArcSDE 8.1 nicht mit einem Pentium 4 Prozessor lauffähig ist, da Oracle 8i nicht fehlerfrei auf diesen Prozessoren arbeitet. ArcInfo ist nicht mit dem Betriessystem XP fehlerfrei zu installieren.

C DatenbankSkript

C 1 SQL - Skript Flussmodell

```
DROP TABLE Gewaesser_Fluss CASCADE CONSTRAINTS;
CREATE TABLE Gewaesser_Fluss (
       Gewaesser_ID         INTEGER NOT NULL,
       Fluss_ID             INTEGER NOT NULL
);
ALTER TABLE Gewaesser_Fluss
       ADD  ( PRIMARY KEY (Gewaesser_ID, Fluss_ID) ) ;
DROP TABLE Telefon CASCADE CONSTRAINTS;
CREATE TABLE Telefon (
       Verwaltungsbehoerde_ID INTEGER NOT NULL,
       Stadt_ID             INTEGER NULL,
       Durchwahl            INTEGER NULL,
       Fax                  INTEGER NULL
);
ALTER TABLE Telefon
       ADD  ( PRIMARY KEY (Verwaltungsbehoerde_ID) ) ;
DROP TABLE Adresse_Stadt CASCADE CONSTRAINTS;
CREATE TABLE Adresse_Stadt (
       Verwaltungsbehoerde_ID INTEGER NOT NULL,
       Stadt_ID             INTEGER NOT NULL
);
ALTER TABLE Adresse_Stadt
       ADD  ( PRIMARY KEY (Verwaltungsbehoerde_ID, Stadt_ID) ) ;
DROP TABLE Adresse CASCADE CONSTRAINTS;
CREATE TABLE Adresse (
       Verwaltungsbehoerde_ID INTEGER NOT NULL,
       Strasse              VARCHAR2(20) NULL,
       Hausnummer           VARCHAR2(5) NULL,
       PLZ                  INTEGER NULL,
       Land                 VARCHAR2(20) NULL
);
ALTER TABLE Adresse
       ADD  ( PRIMARY KEY (Verwaltungsbehoerde_ID) ) ;
DROP TABLE Verwaltungsbehoerde CASCADE CONSTRAINTS;
CREATE TABLE Verwaltungsbehoerde (
       Verwaltungsbehoerde_ID INTEGER NOT NULL,
       Fluss_ID             INTEGER NULL,
       Ansprechpartner      VARCHAR2(20) NULL
);
ALTER TABLE Verwaltungsbehoerde
       ADD  ( PRIMARY KEY (Verwaltungsbehoerde_ID) ) ;
DROP TABLE PH CASCADE CONSTRAINTS;
CREATE TABLE PH (
       Messwert_ID          INTEGER NULL,
       PH_Wert              FLOAT NULL
);
DROP TABLE Wasserdurchfluss CASCADE CONSTRAINTS;
```

```
CREATE TABLE Wasserdurchfluss (
       Messwert_ID          INTEGER NULL,
       Qubikmeterdurchfluss FLOAT NULL
);
DROP TABLE Pegelstand CASCADE CONSTRAINTS;
CREATE TABLE Pegelstand (
       Messwert_ID          INTEGER NULL,
       Pegelstand_in_m      FLOAT NULL
);
DROP TABLE Haertegrad CASCADE CONSTRAINTS;
CREATE TABLE Haertegrad (
       Messwert_ID          INTEGER NULL,
       Haertegradwert       FLOAT NULL
);
DROP TABLE Nitrat CASCADE CONSTRAINTS;
CREATE TABLE Nitrat (
       Messwert_ID          INTEGER NULL,
       Nitrartwert          FLOAT NULL
);
DROP TABLE Messwert CASCADE CONSTRAINTS;
CREATE TABLE Messwert (
       Messwert_ID          INTEGER NOT NULL,
       Messpunkt_ID         INTEGER NOT NULL,
       Messzeitpunkt        DATE NULL
);
ALTER TABLE Messwert
       ADD  ( PRIMARY KEY (Messwert_ID) ) ;
DROP TABLE Messpunkt CASCADE CONSTRAINTS;
CREATE TABLE Messpunkt (
       Messpunkt_ID         INTEGER NOT NULL,
       Fluss_ID             INTEGER NOT NULL,
       Einrichtungszeitpunkt DATE NULL,
       Laengengrad          VARCHAR2(20) NULL,
       Breitengrad          VARCHAR2(20) NULL
);
ALTER TABLE Messpunkt
       ADD  ( PRIMARY KEY (Messpunkt_ID) ) ;
DROP TABLE Fluss_Region CASCADE CONSTRAINTS;
CREATE TABLE Fluss_Region (
       Regions_ID           INTEGER NOT NULL,
       Fluss_ID             INTEGER NOT NULL
);
ALTER TABLE Fluss_Region
       ADD  ( PRIMARY KEY (Regions_ID, Fluss_ID) ) ;
DROP TABLE Fluss CASCADE CONSTRAINTS;
CREATE TABLE Fluss (
       Fluss_ID             CHAR(18) NOT NULL,
       Namen                VARCHAR2(20) NULL,
       Laenge_in_km         FLOAT NULL,
       Quellgebiet          VARCHAR2(20) NULL,
       Muendungsgewaesser   VARCHAR2(20) NULL,
       natuerlich           CHAR(1) NULL
```

```
);
ALTER TABLE Fluss
       ADD  ( PRIMARY KEY (Fluss_ID) ) ;
DROP TABLE Viehwirtschaft CASCADE CONSTRAINTS;
CREATE TABLE Viehwirtschaft (
       Viehw_ID             INTEGER NOT NULL,
       Regions_ID           INTEGER NULL,
       Viehart              VARCHAR2(20) NULL
                           CHECK (Viehart IN ('Rinder', 'Schweine', 'Hühner'))
);
ALTER TABLE Viehwirtschaft
       ADD  ( PRIMARY KEY (Viehw_ID) ) ;
DROP TABLE Agrawirtschaft CASCADE CONSTRAINTS;
CREATE TABLE Agrawirtschaft (
       Agrar_ID             INTEGER NOT NULL,
       Regions_ID           INTEGER NULL,
       Anbauflaeche_in_qkm  FLOAT NULL,
       Anbauprodukt         VARCHAR2(20) NULL
                           CHECK (Anbauprodukt IN ('Weizen', 'Mais', 'Obst',
'Reis')),
       Arbeitnehmerzahl     LONG NULL
);
ALTER TABLE Agrawirtschaft
       ADD  ( PRIMARY KEY (Agrar_ID) ) ;
DROP TABLE Industrie CASCADE CONSTRAINTS;
CREATE TABLE Industrie (
       Industri_ID          INTEGER NOT NULL,
       Regions_ID           INTEGER NULL,
       Arbeitnehmerzahl     LONG NULL,
       Art                  VARCHAR2(20) NULL
                                   CHECK (Art IN ('Auto', 'Chemische',
'Keaftwerk'))
);
ALTER TABLE Industrie
       ADD  ( PRIMARY KEY (Industri_ID) ) ;
DROP TABLE Stadt CASCADE CONSTRAINTS;
CREATE TABLE Stadt (
       Stadt_ID             INTEGER NOT NULL,
       Regions_ID           INTEGER NULL,
       Einwohnerzahl        LONG NULL,
       Stadt                VARCHAR2(20) NULL,
       Vorwahl              VARCHAR2(20) NULL
);
ALTER TABLE Stadt
       ADD  ( PRIMARY KEY (Stadt_ID) ) ;
DROP TABLE Region CASCADE CONSTRAINTS;
CREATE TABLE Region (
       Regions_ID           INTEGER NOT NULL,
       Bezeichnung          VARCHAR2(20) NULL,
       Flaeche_in_qkm       FLOAT NULL
);
ALTER TABLE Region
```

```
       ADD  ( PRIMARY KEY (Regions_ID) ) ;
DROP TABLE Gewaesser CASCADE CONSTRAINTS;
CREATE TABLE Gewaesser (
       Gewaesser_ID         INTEGER NOT NULL,
       Tiefe_in_m           FLOAT NULL,
       Flaeche_in_qkm       FLOAT NULL,
       Name                 VARCHAR2(20) NULL
);
ALTER TABLE Gewaesser
       ADD  ( PRIMARY KEY (Gewaesser_ID) ) ;
ALTER TABLE Gewaesser_Fluss
       ADD  ( FOREIGN KEY (Fluss_ID)
                            REFERENCES Fluss ) ;
ALTER TABLE Gewaesser_Fluss
       ADD  ( FOREIGN KEY (Gewaesser_ID)
                            REFERENCES Gewaesser ) ;
ALTER TABLE Telefon
       ADD  ( FOREIGN KEY (Verwaltungsbehoerde_ID, Stadt_ID)
                            REFERENCES Adresse_Stadt ) ;
ALTER TABLE Telefon
       ADD  ( FOREIGN KEY (Verwaltungsbehoerde_ID)
                            REFERENCES Verwaltungsbehoerde ) ;
ALTER TABLE Adresse_Stadt
       ADD  ( FOREIGN KEY (Stadt_ID)
                            REFERENCES Stadt ) ;
ALTER TABLE Adresse_Stadt
       ADD  ( FOREIGN KEY (Verwaltungsbehoerde_ID)
                            REFERENCES Adresse ) ;
ALTER TABLE Adresse
       ADD  ( FOREIGN KEY (Verwaltungsbehoerde_ID)
                            REFERENCES Verwaltungsbehoerde ) ;
ALTER TABLE Verwaltungsbehoerde
       ADD  ( FOREIGN KEY (Fluss_ID)
                            REFERENCES Fluss ) ;
ALTER TABLE PH
       ADD  ( FOREIGN KEY (Messwert_ID)
                            REFERENCES Messwert ) ;
ALTER TABLE Wasserdurchfluss
       ADD  ( FOREIGN KEY (Messwert_ID)
                            REFERENCES Messwert ) ;
ALTER TABLE Pegelstand
       ADD  ( FOREIGN KEY (Messwert_ID)
                            REFERENCES Messwert ) ;
ALTER TABLE Haertegrad
       ADD  ( FOREIGN KEY (Messwert_ID)
                            REFERENCES Messwert ) ;
ALTER TABLE Nitrat
       ADD  ( FOREIGN KEY (Messwert_ID)
                            REFERENCES Messwert ) ;
ALTER TABLE Messwert
       ADD  ( FOREIGN KEY (Messpunkt_ID)
                            REFERENCES Messpunkt ) ;
```

```
ALTER TABLE Messpunkt
       ADD  ( FOREIGN KEY (Fluss_ID)
                           REFERENCES Fluss ) ;
ALTER TABLE Fluss_Region
       ADD  ( FOREIGN KEY (Regions_ID)
                           REFERENCES Region ) ;
ALTER TABLE Fluss_Region
       ADD  ( FOREIGN KEY (Fluss_ID)
                           REFERENCES Fluss ) ;
ALTER TABLE Fluss
       ADD  ( FOREIGN KEY (Fluss_ID)
                           REFERENCES Fluss ) ;
ALTER TABLE Viehwirtschaft
       ADD  ( FOREIGN KEY (Regions_ID)
                           REFERENCES Region ) ;
ALTER TABLE Agrawirtschaft
       ADD  ( FOREIGN KEY (Regions_ID)
                           REFERENCES Region ) ;
ALTER TABLE Industrie
       ADD  ( FOREIGN KEY (Regions_ID)
                           REFERENCES Region ) ;
ALTER TABLE Stadt
       ADD  ( FOREIGN KEY (Regions_ID)
                           REFERENCES Region ) ;
create trigger tI_Gewaesser_Fluss after INSERT on Gewaesser_Fluss for each row
-- INSERT trigger on Gewaesser_Fluss
declare numrows INTEGER;
begin
    /* Fluss R/25 Gewaesser_Fluss ON CHILD INSERT RESTRICT */
    select count(*) into numrows
      from Fluss
      where
        /* %JoinFKPK(:%New,Fluss," = "," and") */
        :new.Fluss_ID = Fluss.Fluss_ID;
    if (
      /* %NotnullFK(:%New," is not null and") */

      numrows = 0
    )
    then
      raise_application_error(
        -20002,
        'Cannot INSERT Gewaesser_Fluss because Fluss does not exist.'
      );
    end if;
/* Gewaesser R/22 Gewaesser_Fluss ON CHILD INSERT RESTRICT   */
    select count(*) into numrows
      from Gewaesser
      where
        /* %JoinFKPK(:%New,Gewaesser," = "," and") */
        :new.Gewaesser_ID = Gewaesser.Gewaesser_ID;
    if (
```

```
      /* %NotnullFK(:%New," is not null and") */

      numrows = 0
    )
    then
      raise_application_error(
        -20002,
        'Cannot INSERT Gewaesser_Fluss because Gewaesser does not exist.'
      );
    end if;
end;
/
create trigger tU_Gewaesser_Fluss after UPDATE on Gewaesser_Fluss for each row
declare numrows INTEGER;
begin
  /* Fluss R/25 Gewaesser_Fluss ON CHILD UPDATE RESTRICT */
  select count(*) into numrows
    from Fluss
    where
      /* %JoinFKPK(:%New,Fluss," = "," and") */
      :new.Fluss_ID = Fluss.Fluss_ID;
  if (
    /* %NotnullFK(:%New," is not null and") */
    numrows = 0
  )
  then
    raise_application_error(
      -20007,
      'Cannot UPDATE Gewaesser_Fluss because Fluss does not exist.'
    );
  end if;
/* Gewaesser R/22 Gewaesser_Fluss ON CHILD UPDATE RESTRICT */
  select count(*) into numrows
    from Gewaesser
    where
      /* %JoinFKPK(:%New,Gewaesser," = "," and") */
      :new.Gewaesser_ID = Gewaesser.Gewaesser_ID;
  if (
    /* %NotnullFK(:%New," is not null and") */
    numrows = 0
  )
  then
    raise_application_error(
      -20007,
      'Cannot UPDATE Gewaesser_Fluss because Gewaesser does not exist.'
    );
  end if;
end;
/
create trigger tI_Telefon after INSERT on Telefon for each row
-- INSERT trigger on Telefon
declare numrows INTEGER;
```

```
begin
    /* Adresse_Stadt R/34 Telefon ON CHILD INSERT SET NULL */
    update Telefon
      set
        /* %SetFK(Telefon,NULL) */
        Telefon.Verwaltungsbehoerde_ID = NULL,
        Telefon.Stadt_ID = NULL
      where
        not exists (
          select * from Adresse_Stadt
            where
              /* %JoinFKPK(:%New,Adresse_Stadt," = "," and") */
              :new.Verwaltungsbehoerde_ID                                =
Adresse_Stadt.Verwaltungsbehoerde_ID and
              :new.Stadt_ID = Adresse_Stadt.Stadt_ID
        ) and
        /* %JoinPKPK(Telefon,:%New," = "," and") */
        Telefon.Verwaltungsbehoerde_ID = :new.Verwaltungsbehoerde_ID;
/* Verwaltungsbehoerde R/27 Telefon ON CHILD INSERT RESTRICT */
    select count(*) into numrows
      from Verwaltungsbehoerde
      where
        /* %JoinFKPK(:%New,Verwaltungsbehoerde," = "," and") */
        :new.Verwaltungsbehoerde_ID                                      =
Verwaltungsbehoerde.Verwaltungsbehoerde_ID;
    if (
      /* %NotnullFK(:%New," is not null and") */
      numrows = 0
    )
    then
      raise_application_error(
        -20002,
        'Cannot INSERT Telefon because Verwaltungsbehoerde does not exist.'
      );
    end if;
end;
/
create trigger tU_Telefon after UPDATE on Telefon for each row
-- UPDATE trigger on Telefon
declare numrows INTEGER;
begin
/* Adresse_Stadt R/34 Telefon ON CHILD UPDATE SET NULL */
    update Telefon
      set
        /* %SetFK(Telefon,NULL) */
        Telefon.Verwaltungsbehoerde_ID = NULL,
        Telefon.Stadt_ID = NULL
      where
        not exists (
          select * from Adresse_Stadt
            where
              /* %JoinFKPK(:%New,Adresse_Stadt," = "," and") */
```

```
              :new.Verwaltungsbehoerde_ID                              =
Adresse_Stadt.Verwaltungsbehoerde_ID and
              :new.Stadt_ID = Adresse_Stadt.Stadt_ID
        ) and
        /* %JoinPKPK(Telefon,:%New," = "," and") */
        Telefon.Verwaltungsbehoerde_ID = :new.Verwaltungsbehoerde_ID;
/* Verwaltungsbehoerde R/27 Telefon ON CHILD UPDATE RESTRICT */
  select count(*) into numrows
    from Verwaltungsbehoerde
    where
      /* %JoinFKPK(:%New,Verwaltungsbehoerde," = "," and") */
      :new.Verwaltungsbehoerde_ID = Verwaltungsbehoerde.Verwaltungsbehoerde_ID;
  if (
    /* %NotnullFK(:%New," is not null and") */
    numrows = 0
  )
  then
    raise_application_error(
      -20007,
      'Cannot UPDATE Telefon because Verwaltungsbehoerde does not exist.'
    );
  end if;
end;
/
create trigger tD_Adresse_Stadt after DELETE on Adresse_Stadt for each row
-- DELETE trigger on Adresse_Stadt
declare numrows INTEGER;
begin
    /* Adresse_Stadt R/34 Telefon ON PARENT DELETE SET NULL */
    update Telefon
      set
        /* %SetFK(Telefon,NULL) */
        Telefon.Verwaltungsbehoerde_ID = NULL,
        Telefon.Stadt_ID = NULL
      where
        /* %JoinFKPK(Telefon,:%Old," = "," and") */
        Telefon.Verwaltungsbehoerde_ID = :old.Verwaltungsbehoerde_ID and
        Telefon.Stadt_ID = :old.Stadt_ID;
end;
/
create trigger tI_Adresse_Stadt after INSERT on Adresse_Stadt for each row
-- INSERT trigger on Adresse_Stadt
declare numrows INTEGER;
begin
    /* Stadt R/33 Adresse_Stadt ON CHILD INSERT RESTRICT */
    select count(*) into numrows
      from Stadt
      where
        /* %JoinFKPK(:%New,Stadt," = "," and") */
        :new.Stadt_ID = Stadt.Stadt_ID;
    if (
      /* %NotnullFK(:%New," is not null and") */
```

```
      numrows = 0
    )
    then
      raise_application_error(
        -20002,
        'Cannot INSERT Adresse_Stadt because Stadt does not exist.'
      );
    end if;
    /* Adresse R/32 Adresse_Stadt ON CHILD INSERT RESTRICT */
    select count(*) into numrows
      from Adresse
      where
        /* %JoinFKPK(:%New,Adresse," = "," and") */
        :new.Verwaltungsbehoerde_ID = Adresse.Verwaltungsbehoerde_ID;
    if (
      /* %NotnullFK(:%New," is not null and") */
      numrows = 0
    )
    then
      raise_application_error(
        -20002,
        'Cannot INSERT Adresse_Stadt because Adresse does not exist.'
      );
    end if;
end;
/
create trigger tU_Adresse_Stadt after UPDATE on Adresse_Stadt for each row
-- UPDATE trigger on Adresse_Stadt
declare numrows INTEGER;
begin
  /* Adresse_Stadt R/34 Telefon ON PARENT UPDATE SET NULL */
  if
    /* %JoinPKPK(:%Old,:%New," <> "," or " */
    :old.Verwaltungsbehoerde_ID <> :new.Verwaltungsbehoerde_ID or
    :old.Stadt_ID <> :new.Stadt_ID
  then
    update Telefon
      set
        /* %SetFK(Telefon,NULL) */
        Telefon.Verwaltungsbehoerde_ID = NULL,
        Telefon.Stadt_ID = NULL
      where
        /* %JoinFKPK(Telefon,:%Old," = ","") */
        Telefon.Verwaltungsbehoerde_ID = :old.Verwaltungsbehoerde_ID and
        Telefon.Stadt_ID = :old.Stadt_ID;
  end if;
  /* Stadt R/33 Adresse_Stadt ON CHILD UPDATE RESTRICT */
  select count(*) into numrows
    from Stadt
    where
      /* %JoinFKPK(:%New,Stadt," = "," and") */
      :new.Stadt_ID = Stadt.Stadt_ID;
```

```
  if (
    /* %NotnullFK(:%New," is not null and") */
    numrows = 0
  )
  then
    raise_application_error(
      -20007,
      'Cannot UPDATE Adresse_Stadt because Stadt does not exist.'
    );
  end if;
  /* Adresse R/32 Adresse_Stadt ON CHILD UPDATE RESTRICT */
  select count(*) into numrows
    from Adresse
    where
      /* %JoinFKPK(:%New,Adresse," = "," and") */
      :new.Verwaltungsbehoerde_ID = Adresse.Verwaltungsbehoerde_ID;
  if (
    /* %NotnullFK(:%New," is not null and") */


    numrows = 0
  )
  then
    raise_application_error(
      -20007,
      'Cannot UPDATE Adresse_Stadt because Adresse does not exist.'
    );
  end if;
end;
create trigger tD_Adresse after DELETE on Adresse for each row
-- DELETE trigger on Adresse
declare numrows INTEGER;
begin
    /* Adresse R/32 Adresse_Stadt ON PARENT DELETE RESTRICT */
    select count(*) into numrows
      from Adresse_Stadt
      where
        /*  %JoinFKPK(Adresse_Stadt,:%Old," = "," and") */
        Adresse_Stadt.Verwaltungsbehoerde_ID = :old.Verwaltungsbehoerde_ID;
    if (numrows > 0)
    then
      raise_application_error(
        -20001,
        'Cannot DELETE Adresse because Adresse_Stadt exists.'
      );
    end if;
end;
create trigger tI_Adresse after INSERT on Adresse for each row
-- INSERT trigger on Adresse
declare numrows INTEGER;
begin
    /* ERwin Builtin Sat Jun 29 11:17:28 2002 */
```

```
    /* Verwaltungsbehoerde R/26 Adresse ON CHILD INSERT RESTRICT */
    select count(*) into numrows
      from Verwaltungsbehoerde
      where
        /* %JoinFKPK(:%New,Verwaltungsbehoerde," = "," and") */
        :new.Verwaltungsbehoerde_ID                                    =
Verwaltungsbehoerde.Verwaltungsbehoerde_ID;
    if (
      /* %NotnullFK(:%New," is not null and") */
      numrows = 0
    )
    then
      raise_application_error(
        -20002,
        'Cannot INSERT Adresse because Verwaltungsbehoerde does not exist.'
      );
    end if;
end;
create trigger tU_Adresse after UPDATE on Adresse for each row
-- UPDATE trigger on Adresse
declare numrows INTEGER;
begin
  /* Adresse R/32 Adresse_Stadt ON PARENT UPDATE RESTRICT */
  if
    /* %JoinPKPK(:%Old,:%New," <> "," or ") */
    :old.Verwaltungsbehoerde_ID <> :new.Verwaltungsbehoerde_ID
  then
    select count(*) into numrows
      from Adresse_Stadt
      where
        /*  %JoinFKPK(Adresse_Stadt,:%Old," = "," and") */
        Adresse_Stadt.Verwaltungsbehoerde_ID = :old.Verwaltungsbehoerde_ID;
    if (numrows > 0)
    then
      raise_application_error(
        -20005,
        'Cannot UPDATE Adresse because Adresse_Stadt exists.'
      );
    end if;
  end if;
  /* Verwaltungsbehoerde R/26 Adresse ON CHILD UPDATE RESTRICT */
  select count(*) into numrows
    from Verwaltungsbehoerde
    where
      /* %JoinFKPK(:%New,Verwaltungsbehoerde," = "," and") */
      :new.Verwaltungsbehoerde_ID = Verwaltungsbehoerde.Verwaltungsbehoerde_ID;
  if (
    /* %NotnullFK(:%New," is not null and") */
    numrows = 0
  )
  then
    raise_application_error(
```

```
      -20007,
      'Cannot UPDATE Adresse because Verwaltungsbehoerde does not exist.'
    );
  end if;
end;
/
create trigger tD_Verwaltungsbehoerde after DELETE on Verwaltungsbehoerde for
each row
-- DELETE trigger on Verwaltungsbehoerde
declare numrows INTEGER;
begin
    /* Verwaltungsbehoerde R/27 Telefon ON PARENT DELETE RESTRICT */
    select count(*) into numrows
      from Telefon
      where
        /*  %JoinFKPK(Telefon,:%Old," = "," and") */
 Telefon.Verwaltungsbehoerde_ID=:old.Verwaltungsbehoerde_ID;
    if (numrows > 0)
    then
      raise_application_error(
        -20001,
        'Cannot DELETE Verwaltungsbehoerde because Telefon exists.'
      );
    end if;
/* Verwaltungsbehoerde R/26 Adresse ON PARENT DELETE RESTRICT */
    select count(*) into numrows
      from Adresse
      where
        /*  %JoinFKPK(Adresse,:%Old," = "," and") */
        Adresse.Verwaltungsbehoerde_ID = :old.Verwaltungsbehoerde_ID;
    if (numrows > 0)
    then
      raise_application_error(
        -20001,
        'Cannot DELETE Verwaltungsbehoerde because Adresse exists.'
      );
    end if;
end;
create trigger tI_Verwaltungsbehoerde after INSERT on Verwaltungsbehoerde for
each row
-- INSERT trigger on Verwaltungsbehoerde
declare numrows INTEGER;
begin
/* Fluss R/16 Verwaltungsbehoerde ON CHILD INSERT SET NULL */
    update Verwaltungsbehoerde
      set
        /* %SetFK(Verwaltungsbehoerde,NULL) */
        Verwaltungsbehoerde.Fluss_ID = NULL
      where
        not exists (
          select * from Fluss
            where
```

```
              /* %JoinFKPK(:%New,Fluss," = "," and") */
              :new.Fluss_ID = Fluss.Fluss_ID
        ) and
        /* %JoinPKPK(Verwaltungsbehoerde,:%New," = "," and") */
        Verwaltungsbehoerde.Verwaltungsbehoerde_ID                    =
:new.Verwaltungsbehoerde_ID;
end;
create trigger tU_Verwaltungsbehoerde after UPDATE on Verwaltungsbehoerde for
each row
-- UPDATE trigger on Verwaltungsbehoerde
declare numrows INTEGER;
begin
/*Verwaltungsbehoerde R/27 Telefon ON PARENT UPDATE RESTRICT */
  if
    /* %JoinPKPK(:%Old,:%New," <> "," or ") */
    :old.Verwaltungsbehoerde_ID <> :new.Verwaltungsbehoerde_ID
  then
    select count(*) into numrows
      from Telefon
      where
        /*  %JoinFKPK(Telefon,:%Old," = "," and") */
        Telefon.Verwaltungsbehoerde_ID = :old.Verwaltungsbehoerde_ID;
    if (numrows > 0)
    then
      raise_application_error(
        -20005,
        'Cannot UPDATE Verwaltungsbehoerde because Telefon exists.'
      );
    end if;
  end if;
/*Verwaltungsbehoerde R/26 Adresse ON PARENT UPDATE RESTRICT */
  if
    /* %JoinPKPK(:%Old,:%New," <> "," or ") */
    :old.Verwaltungsbehoerde_ID <> :new.Verwaltungsbehoerde_ID
  then
    select count(*) into numrows
      from Adresse
      where
        /*  %JoinFKPK(Adresse,:%Old," = "," and") */
        Adresse.Verwaltungsbehoerde_ID = :old.Verwaltungsbehoerde_ID;
    if (numrows > 0)
    then
      raise_application_error(
        -20005,
        'Cannot UPDATE Verwaltungsbehoerde because Adresse exists.'
      );
    end if;
  end if;
/* Fluss R/16 Verwaltungsbehoerde ON CHILD UPDATE SET NULL */
    update Verwaltungsbehoerde
      set
        /* %SetFK(Verwaltungsbehoerde,NULL) */
```

```
        Verwaltungsbehoerde.Fluss_ID = NULL
      where
        not exists (
          select * from Fluss
            where
              /* %JoinFKPK(:%New,Fluss," = "," and") */
              :new.Fluss_ID = Fluss.Fluss_ID
        ) and
        /* %JoinPKPK(Verwaltungsbehoerde,:%New," = "," and") */
        Verwaltungsbehoerde.Verwaltungsbehoerde_ID                    =
:new.Verwaltungsbehoerde_ID;
end;
create trigger tI_PH after INSERT on PH for each row
-- INSERT trigger on PH
declare numrows INTEGER;
begin
    /* Messwert R/10 PH ON CHILD INSERT SET NULL */
    update PH
      set
        /* %SetFK(PH,NULL) */
        PH.Messwert_ID = NULL
      where
        not exists (
          select * from Messwert
            where
              /* %JoinFKPK(:%New,Messwert," = "," and") */
              :new.Messwert_ID = Messwert.Messwert_ID
        ) and
        /* %JoinPKPK(PH,:%New," = "," and") */
        ;
end;
create trigger tU_PH after UPDATE on PH for each row
-- UPDATE trigger on PH
declare numrows INTEGER;
begin
    /* Messwert R/10 PH ON CHILD UPDATE SET NULL */
    update PH
      set
        /* %SetFK(PH,NULL) */
        PH.Messwert_ID = NULL
      where
        not exists (
          select * from Messwert
            where
              /* %JoinFKPK(:%New,Messwert," = "," and") */
              :new.Messwert_ID = Messwert.Messwert_ID
        ) and
        /* %JoinPKPK(PH,:%New," = "," and") */
        ;
end;
create trigger tI_Wasserdurchfluss after INSERT on Wasserdurchfluss for each
row
```

```
-- INSERT trigger on Wasserdurchfluss
declare numrows INTEGER;
begin
/* Messwert R/9 Wasserdurchfluss ON CHILD INSERT SET NULL */
    update Wasserdurchfluss
      set
        /* %SetFK(Wasserdurchfluss,NULL) */
        Wasserdurchfluss.Messwert_ID = NULL
      where
        not exists (
          select * from Messwert
            where
              /* %JoinFKPK(:%New,Messwert," = "," and") */
              :new.Messwert_ID = Messwert.Messwert_ID
        ) and
        /* %JoinPKPK(Wasserdurchfluss,:%New," = "," and") */
        ;
end;
create trigger tU_Wasserdurchfluss after UPDATE on Wasserdurchfluss for each
row
-- UPDATE trigger on Wasserdurchfluss
declare numrows INTEGER;
begin
 /* Messwert R/9 Wasserdurchfluss ON CHILD UPDATE SET NULL */
    update Wasserdurchfluss
      set
        /* %SetFK(Wasserdurchfluss,NULL) */
        Wasserdurchfluss.Messwert_ID = NULL
      where
        not exists (
          select * from Messwert
            where
              /* %JoinFKPK(:%New,Messwert," = "," and") */
              :new.Messwert_ID = Messwert.Messwert_ID
        ) and
        /* %JoinPKPK(Wasserdurchfluss,:%New," = "," and") */
        ;
end;
/
create trigger tI_Pegelstand after INSERT on Pegelstand for each row
-- INSERT trigger on Pegelstand
declare numrows INTEGER;
begin
    /* Messwert R/8 Pegelstand ON CHILD INSERT SET NULL */
    update Pegelstand
      set
        /* %SetFK(Pegelstand,NULL) */
        Pegelstand.Messwert_ID = NULL
      where
        not exists (
          select * from Messwert
            where
```

```
              /* %JoinFKPK(:%New,Messwert," = "," and") */
              :new.Messwert_ID = Messwert.Messwert_ID
        ) and
        /* %JoinPKPK(Pegelstand,:%New," = "," and") */
        ;
end;
create trigger tU_Pegelstand after UPDATE on Pegelstand for each row
-- UPDATE trigger on Pegelstand
declare numrows INTEGER;
begin
    /* Messwert R/8 Pegelstand ON CHILD UPDATE SET NULL */
    update Pegelstand
      set
        /* %SetFK(Pegelstand,NULL) */
        Pegelstand.Messwert_ID = NULL
      where
        not exists (
          select * from Messwert
            where
              /* %JoinFKPK(:%New,Messwert," = "," and") */
              :new.Messwert_ID = Messwert.Messwert_ID
        ) and
        /* %JoinPKPK(Pegelstand,:%New," = "," and") */
        ;
end;
create trigger tI_Haertegrad after INSERT on Haertegrad for each row
-- INSERT trigger on Haertegrad
declare numrows INTEGER;
begin
    /* Messwert R/7 Haertegrad ON CHILD INSERT SET NULL */
    update Haertegrad
      set
        /* %SetFK(Haertegrad,NULL) */
        Haertegrad.Messwert_ID = NULL
      where
        not exists (
          select * from Messwert
            where
              /* %JoinFKPK(:%New,Messwert," = "," and") */
              :new.Messwert_ID = Messwert.Messwert_ID
        ) and
        /* %JoinPKPK(Haertegrad,:%New," = "," and") */
        ;
end;
create trigger tU_Haertegrad after UPDATE on Haertegrad for each row
-- UPDATE trigger on Haertegrad
declare numrows INTEGER;
begin
    /* Messwert R/7 Haertegrad ON CHILD UPDATE SET NULL */
    update Haertegrad
      set
        /* %SetFK(Haertegrad,NULL) */
```

```
        Haertegrad.Messwert_ID = NULL
      where
        not exists (
          select * from Messwert
            where
              /* %JoinFKPK(:%New,Messwert," = "," and") */
              :new.Messwert_ID = Messwert.Messwert_ID
        ) and
        /* %JoinPKPK(Haertegrad,:%New," = "," and") */
        ;
end;
create trigger tI_Nitrat after INSERT on Nitrat for each row
-- INSERT trigger on Nitrat
declare numrows INTEGER;
begin
    /* Messwert R/6 Nitrat ON CHILD INSERT SET NULL */
    update Nitrat
      set
        /* %SetFK(Nitrat,NULL) */
        Nitrat.Messwert_ID = NULL
      where
        not exists (
          select * from Messwert
            where
              /* %JoinFKPK(:%New,Messwert," = "," and") */
              :new.Messwert_ID = Messwert.Messwert_ID
        ) and
        /* %JoinPKPK(Nitrat,:%New," = "," and") */
        ;
end;
create trigger tU_Nitrat after UPDATE on Nitrat for each row
-- UPDATE trigger on Nitrat
declare numrows INTEGER;
begin
    /* ERwin Builtin Sat Jun 29 11:17:28 2002 */
    /* Messwert R/6 Nitrat ON CHILD UPDATE SET NULL */
    update Nitrat
      set
        /* %SetFK(Nitrat,NULL) */
        Nitrat.Messwert_ID = NULL
      where
        not exists (
          select * from Messwert
            where
              /* %JoinFKPK(:%New,Messwert," = "," and") */
              :new.Messwert_ID = Messwert.Messwert_ID
        ) and
        /* %JoinPKPK(Nitrat,:%New," = "," and") */
        ;
end;
create trigger tD_Messwert after DELETE on Messwert for each row
-- DELETE trigger on Messwert
```

```
declare numrows INTEGER;
begin
    /* Messwert R/10 PH ON PARENT DELETE SET NULL */
    update PH
      set
        /* %SetFK(PH,NULL) */
        PH.Messwert_ID = NULL
      where
        /* %JoinFKPK(PH,:%Old," = "," and") */
        PH.Messwert_ID = :old.Messwert_ID;
/* Messwert R/9 Wasserdurchfluss ON PARENT DELETE SET NULL */
    update Wasserdurchfluss
      set
        /* %SetFK(Wasserdurchfluss,NULL) */
        Wasserdurchfluss.Messwert_ID = NULL
      where
        /* %JoinFKPK(Wasserdurchfluss,:%Old," = "," and") */
        Wasserdurchfluss.Messwert_ID = :old.Messwert_ID;
 /* Messwert R/8 Pegelstand ON PARENT DELETE SET NULL */
    update Pegelstand
      set
        /* %SetFK(Pegelstand,NULL) */
        Pegelstand.Messwert_ID = NULL
      where
        /* %JoinFKPK(Pegelstand,:%Old," = "," and") */
        Pegelstand.Messwert_ID = :old.Messwert_ID;
/* Messwert R/7 Haertegrad ON PARENT DELETE SET NULL */
    update Haertegrad
      set
        /* %SetFK(Haertegrad,NULL) */
        Haertegrad.Messwert_ID = NULL
      where
        /* %JoinFKPK(Haertegrad,:%Old," = "," and") */
        Haertegrad.Messwert_ID = :old.Messwert_ID;
/* Messwert R/6 Nitrat ON PARENT DELETE SET NULL */
    update Nitrat
      set
        /* %SetFK(Nitrat,NULL) */
        Nitrat.Messwert_ID = NULL
      where
        /* %JoinFKPK(Nitrat,:%Old," = "," and") */
        Nitrat.Messwert_ID = :old.Messwert_ID;
end;
create trigger tI_Messwert after INSERT on Messwert for each row
-- INSERT trigger on Messwert
declare numrows INTEGER;
begin
/* Messpunkt R/5 Messwert ON CHILD INSERT SET NULL */
    update Messwert
      set
        /* %SetFK(Messwert,NULL) */
        Messwert.Messpunkt_ID = NULL
```

```
      where
        not exists (
          select * from Messpunkt
            where
              /* %JoinFKPK(:%New,Messpunkt," = "," and") */
              :new.Messpunkt_ID = Messpunkt.Messpunkt_ID
        ) and
        /* %JoinPKPK(Messwert,:%New," = "," and") */
        Messwert.Messwert_ID = :new.Messwert_ID;
end;
create trigger tU_Messwert after UPDATE on Messwert for each row
-- UPDATE trigger on Messwert
declare numrows INTEGER;
begin
  /* Messwert R/10 PH ON PARENT UPDATE SET NULL */
  if
    /* %JoinPKPK(:%Old,:%New," <> "," or " */
    :old.Messwert_ID <> :new.Messwert_ID
  then
    update PH
      set
        /* %SetFK(PH,NULL) */
        PH.Messwert_ID = NULL
      where
        /* %JoinFKPK(PH,:%Old," = ","") */
        PH.Messwert_ID = :old.Messwert_ID;
  end if;
  /* Messwert R/9 Wasserdurchfluss ON PARENT UPDATE SET NULL */
  if
    /* %JoinPKPK(:%Old,:%New," <> "," or " */
    :old.Messwert_ID <> :new.Messwert_ID
  then
    update Wasserdurchfluss
      set
        /* %SetFK(Wasserdurchfluss,NULL) */
        Wasserdurchfluss.Messwert_ID = NULL
      where
        /* %JoinFKPK(Wasserdurchfluss,:%Old," = ","") */
        Wasserdurchfluss.Messwert_ID = :old.Messwert_ID;
  end if;
  /* Messwert R/8 Pegelstand ON PARENT UPDATE SET NULL */
  if
    /* %JoinPKPK(:%Old,:%New," <> "," or " */
    :old.Messwert_ID <> :new.Messwert_ID
  then
    update Pegelstand
      set
        /* %SetFK(Pegelstand,NULL) */
        Pegelstand.Messwert_ID = NULL
      where
        /* %JoinFKPK(Pegelstand,:%Old," = ","") */
        Pegelstand.Messwert_ID = :old.Messwert_ID;
```

```
  end if;
/* Messwert R/7 Haertegrad ON PARENT UPDATE SET NULL */
  if
    /* %JoinPKPK(:%Old,:%New," <> "," or " */
    :old.Messwert_ID <> :new.Messwert_ID
  then
    update Haertegrad
      set
        /* %SetFK(Haertegrad,NULL) */
        Haertegrad.Messwert_ID = NULL
      where
        /* %JoinFKPK(Haertegrad,:%Old," = ","," ) */
        Haertegrad.Messwert_ID = :old.Messwert_ID;
  end if;
/* Messwert R/6 Nitrat ON PARENT UPDATE SET NULL */
  if
    /* %JoinPKPK(:%Old,:%New," <> "," or " */
    :old.Messwert_ID <> :new.Messwert_ID
  then
    update Nitrat
      set
        /* %SetFK(Nitrat,NULL) */
        Nitrat.Messwert_ID = NULL
      where
        /* %JoinFKPK(Nitrat,:%Old," = ","," ) */
        Nitrat.Messwert_ID = :old.Messwert_ID;
  end if;
/* Messpunkt R/5 Messwert ON CHILD UPDATE SET NULL */
    update Messwert
      set
        /* %SetFK(Messwert,NULL) */
        Messwert.Messpunkt_ID = NULL
      where
        not exists (
          select * from Messpunkt
            where
              /* %JoinFKPK(:%New,Messpunkt," = "," and") */
              :new.Messpunkt_ID = Messpunkt.Messpunkt_ID
        ) and
        /* %JoinPKPK(Messwert,:%New," = "," and") */
        Messwert.Messwert_ID = :new.Messwert_ID;
end;
create trigger tD_Messpunkt after DELETE on Messpunkt for each row
-- DELETE trigger on Messpunkt
declare numrows INTEGER;
begin
    /* Messpunkt R/5 Messwert ON PARENT DELETE SET NULL */
    update Messwert
      set
        /* %SetFK(Messwert,NULL) */
        Messwert.Messpunkt_ID = NULL
      where
```

```
        /* %JoinFKPK(Messwert,:%Old," = "," and") */
        Messwert.Messpunkt_ID = :old.Messpunkt_ID;
end;
create trigger tI_Messpunkt after INSERT on Messpunkt for each row
-- INSERT trigger on Messpunkt
declare numrows INTEGER;
begin
    /* Fluss R/4 Messpunkt ON CHILD INSERT SET NULL */
    update Messpunkt
      set
        /* %SetFK(Messpunkt,NULL) */
        Messpunkt.Fluss_ID = NULL
      where
        not exists (
          select * from Fluss
            where
              /* %JoinFKPK(:%New,Fluss," = "," and") */
              :new.Fluss_ID = Fluss.Fluss_ID
        ) and
        /* %JoinPKPK(Messpunkt,:%New," = "," and") */
        Messpunkt.Messpunkt_ID = :new.Messpunkt_ID;
end;
create trigger tU_Messpunkt after UPDATE on Messpunkt for each row
-- UPDATE trigger on Messpunkt
declare numrows INTEGER;
begin
  /* Messpunkt R/5 Messwert ON PARENT UPDATE SET NULL */
  if
    /* %JoinPKPK(:%Old,:%New," <> "," or " */
    :old.Messpunkt_ID <> :new.Messpunkt_ID
  then
    update Messwert
      set
        /* %SetFK(Messwert,NULL) */
        Messwert.Messpunkt_ID = NULL
      where
        /* %JoinFKPK(Messwert,:%Old," = ","," ) */
        Messwert.Messpunkt_ID = :old.Messpunkt_ID;
  end if;
/* Fluss R/4 Messpunkt ON CHILD UPDATE SET NULL */
    update Messpunkt
      set
        /* %SetFK(Messpunkt,NULL) */
        Messpunkt.Fluss_ID = NULL
      where
        not exists (
          select * from Fluss
            where
              /* %JoinFKPK(:%New,Fluss," = "," and") */
              :new.Fluss_ID = Fluss.Fluss_ID
        ) and
        /* %JoinPKPK(Messpunkt,:%New," = "," and") */
```

```
        Messpunkt.Messpunkt_ID = :new.Messpunkt_ID;
end;
create trigger tI_Fluss_Region after INSERT on Fluss_Region for each row
-- INSERT trigger on Fluss_Region
declare numrows INTEGER;
begin
    /* Region R/3 Fluss_Region ON CHILD INSERT RESTRICT */
    select count(*) into numrows
      from Region
      where
        /* %JoinFKPK(:%New,Region," = "," and") */
        :new.Regions_ID = Region.Regions_ID;
    if (
      /* %NotnullFK(:%New," is not null and") */
      numrows = 0
    )
    then
      raise_application_error(
        -20002,
        'Cannot INSERT Fluss_Region because Region does not exist.'
      );
    end if;
/* Fluss R/2 Fluss_Region ON CHILD INSERT RESTRICT */
    select count(*) into numrows
      from Fluss
      where
        /* %JoinFKPK(:%New,Fluss," = "," and") */
        :new.Fluss_ID = Fluss.Fluss_ID;
    if (
      /* %NotnullFK(:%New," is not null and") */
      numrows = 0
    )
    then
      raise_application_error(
        -20002,
        'Cannot INSERT Fluss_Region because Fluss does not exist.'
      );
    end if;
end;
create trigger tU_Fluss_Region after UPDATE on Fluss_Region for each row
-- UPDATE trigger on Fluss_Region
declare numrows INTEGER;
begin
  /* Region R/3 Fluss_Region ON CHILD UPDATE RESTRICT */
  select count(*) into numrows
    from Region
    where
      /* %JoinFKPK(:%New,Region," = "," and") */
      :new.Regions_ID = Region.Regions_ID;
  if (
    /* %NotnullFK(:%New," is not null and") */
    numrows = 0
```

```
  )
  then
    raise_application_error(
      -20007,
      'Cannot UPDATE Fluss_Region because Region does not exist.'
    );
  end if;
/* Fluss R/2 Fluss_Region ON CHILD UPDATE RESTRICT */
  select count(*) into numrows
    from Fluss
    where
      /* %JoinFKPK(:%New,Fluss," = "," and") */
      :new.Fluss_ID = Fluss.Fluss_ID;
  if (
    /* %NotnullFK(:%New," is not null and") */
    numrows = 0
  )
  then
    raise_application_error(
      -20007,
      'Cannot UPDATE Fluss_Region because Fluss does not exist.'
    );
  end if;
end;
create trigger tD_Fluss after DELETE on Fluss for each row
-- DELETE trigger on Fluss
declare numrows INTEGER;
begin
    /* ERwin Builtin Sat Jun 29 11:17:29 2002 */
    /* Fluss R/25 Gewaesser_Fluss ON PARENT DELETE RESTRICT */
    select count(*) into numrows
      from Gewaesser_Fluss
      where
        /*  %JoinFKPK(Gewaesser_Fluss,:%Old," = "," and") */
        Gewaesser_Fluss.Fluss_ID = :old.Fluss_ID;
    if (numrows > 0)
    then
      raise_application_error(
        -20001,
        'Cannot DELETE Fluss because Gewaesser_Fluss exists.'
      );
    end if;
/* Fluss R/17 Fluss ON PARENT DELETE SET NULL */
    update Fluss
      set
        /* %SetFK(Fluss,NULL) */
        Fluss.Fluss_ID = NULL
      where
        /* %JoinFKPK(Fluss,:%Old," = "," and") */
        Fluss.Fluss_ID = :old.Fluss_ID;
/* Fluss R/16 Verwaltungsbehoerde ON PARENT DELETE SET NULL */
    update Verwaltungsbehoerde
```

```
      set
        /* %SetFK(Verwaltungsbehoerde,NULL) */
        Verwaltungsbehoerde.Fluss_ID = NULL
      where
        /* %JoinFKPK(Verwaltungsbehoerde,:%Old," = "," and") */
        Verwaltungsbehoerde.Fluss_ID = :old.Fluss_ID;
/* Fluss R/4 Messpunkt ON PARENT DELETE SET NULL */
    update Messpunkt
      set
        /* %SetFK(Messpunkt,NULL) */
        Messpunkt.Fluss_ID = NULL
      where
        /* %JoinFKPK(Messpunkt,:%Old," = "," and") */
        Messpunkt.Fluss_ID = :old.Fluss_ID;
/* Fluss R/2 Fluss_Region ON PARENT DELETE RESTRICT */
    select count(*) into numrows
      from Fluss_Region
      where
        /*  %JoinFKPK(Fluss_Region,:%Old," = "," and") */
        Fluss_Region.Fluss_ID = :old.Fluss_ID;
    if (numrows > 0)
    then
      raise_application_error(
        -20001,
        'Cannot DELETE Fluss because Fluss_Region exists.'
      );
    end if;
end;
create trigger tI_Fluss after INSERT on Fluss for each row
-- INSERT trigger on Fluss
declare numrows INTEGER;
begin
    /* Fluss R/17 Fluss ON CHILD INSERT SET NULL */
    update Fluss
      set
        /* %SetFK(Fluss,NULL) */
        Fluss.Fluss_ID = NULL
      where
        not exists (
          select * from Fluss
            where
              /* %JoinFKPK(:%New,Fluss," = "," and") */
              :new.Fluss_ID = Fluss.Fluss_ID
        ) and
        /* %JoinPKPK(Fluss,:%New," = "," and") */
        Fluss.Fluss_ID = :new.Fluss_ID;
end;
create trigger tU_Fluss after UPDATE on Fluss for each row
-- UPDATE trigger on Fluss
declare numrows INTEGER;
begin
  /* Fluss R/25 Gewaesser_Fluss ON PARENT UPDATE RESTRICT */
```

```
  if
    /* %JoinPKPK(:%Old,:%New," <> "," or ") */
    :old.Fluss_ID <> :new.Fluss_ID
  then
    select count(*) into numrows
      from Gewaesser_Fluss
      where
        /*  %JoinFKPK(Gewaesser_Fluss,:%Old," = "," and") */
        Gewaesser_Fluss.Fluss_ID = :old.Fluss_ID;
    if (numrows > 0)
    then
      raise_application_error(
        -20005,
        'Cannot UPDATE Fluss because Gewaesser_Fluss exists.'
      );
    end if;
  end if;
/* Fluss R/17 Fluss ON PARENT UPDATE SET NULL */
  if
    /* %JoinPKPK(:%Old,:%New," <> "," or " */
    :old.Fluss_ID <> :new.Fluss_ID
  then
    update Fluss
      set
        /* %SetFK(Fluss,NULL) */
        Fluss.Fluss_ID = NULL
      where
        /* %JoinFKPK(Fluss,:%Old," = ","") */
        Fluss.Fluss_ID = :old.Fluss_ID;
  end if;
/* Fluss R/16 Verwaltungsbehoerde ON PARENT UPDATE SET NULL */
  if
    /* %JoinPKPK(:%Old,:%New," <> "," or " */
    :old.Fluss_ID <> :new.Fluss_ID
  then
    update Verwaltungsbehoerde
      set
        /* %SetFK(Verwaltungsbehoerde,NULL) */
        Verwaltungsbehoerde.Fluss_ID = NULL
      where
        /* %JoinFKPK(Verwaltungsbehoerde,:%Old," = ","") */
        Verwaltungsbehoerde.Fluss_ID = :old.Fluss_ID;
  end if;
/* Fluss R/4 Messpunkt ON PARENT UPDATE SET NULL */
  if
    /* %JoinPKPK(:%Old,:%New," <> "," or " */
    :old.Fluss_ID <> :new.Fluss_ID
  then
    update Messpunkt
      set
        /* %SetFK(Messpunkt,NULL) */
        Messpunkt.Fluss_ID = NULL
```

```
      where
        /* %JoinFKPK(Messpunkt,:%Old," = ","") */
        Messpunkt.Fluss_ID = :old.Fluss_ID;
  end if;
/* Fluss R/2 Fluss_Region ON PARENT UPDATE RESTRICT */
  if
    /* %JoinPKPK(:%Old,:%New," <> "," or ") */
    :old.Fluss_ID <> :new.Fluss_ID
  then
    select count(*) into numrows
      from Fluss_Region
      where
        /*  %JoinFKPK(Fluss_Region,:%Old," = "," and") */
        Fluss_Region.Fluss_ID = :old.Fluss_ID;
    if (numrows > 0)
    then
      raise_application_error(
        -20005,
        'Cannot UPDATE Fluss because Fluss_Region exists.'
      );
    end if;
  end if;
/* Fluss R/17 Fluss ON CHILD UPDATE SET NULL */
    update Fluss
      set
        /* %SetFK(Fluss,NULL) */
        Fluss.Fluss_ID = NULL
      where
        not exists (
          select * from Fluss
            where
              /* %JoinFKPK(:%New,Fluss," = "," and") */
              :new.Fluss_ID = Fluss.Fluss_ID
        ) and
        /* %JoinPKPK(Fluss,:%New," = "," and") */
        Fluss.Fluss_ID = :new.Fluss_ID;
end;
create trigger tI_Viehwirtschaft after INSERT on Viehwirtschaft for each row
-- INSERT trigger on Viehwirtschaft
declare numrows INTEGER;
begin
    /* Region R/14 Viehwirtschaft ON CHILD INSERT SET NULL */
    update Viehwirtschaft
      set
        /* %SetFK(Viehwirtschaft,NULL) */
        Viehwirtschaft.Regions_ID = NULL
      where
        not exists (
          select * from Region
            where
              /* %JoinFKPK(:%New,Region," = "," and") */
              :new.Regions_ID = Region.Regions_ID
```

```
        ) and
        /* %JoinPKPK(Viehwirtschaft,:%New," = "," and") */
        Viehwirtschaft.Viehw_ID = :new.Viehw_ID;
end;
create trigger tU_Viehwirtschaft after UPDATE on Viehwirtschaft for each row
-- UPDATE trigger on Viehwirtschaft
declare numrows INTEGER;
begin
    /* Region R/14 Viehwirtschaft ON CHILD UPDATE SET NULL */
    update Viehwirtschaft
      set
        /* %SetFK(Viehwirtschaft,NULL) */
        Viehwirtschaft.Regions_ID = NULL
      where
        not exists (
          select * from Region
            where
              /* %JoinFKPK(:%New,Region," = "," and") */
              :new.Regions_ID = Region.Regions_ID
        ) and
        /* %JoinPKPK(Viehwirtschaft,:%New," = "," and") */
        Viehwirtschaft.Viehw_ID = :new.Viehw_ID;
end;
create trigger tI_Agrawirtschaft after INSERT on Agrawirtschaft for each row
-- INSERT trigger on Agrawirtschaft
declare numrows INTEGER;
begin
    /* Region R/13 Agrawirtschaft ON CHILD INSERT SET NULL */
    update Agrawirtschaft
      set
        /* %SetFK(Agrawirtschaft,NULL) */
        Agrawirtschaft.Regions_ID = NULL
      where
        not exists (
          select * from Region
            where
              /* %JoinFKPK(:%New,Region," = "," and") */
              :new.Regions_ID = Region.Regions_ID
        ) and
        /* %JoinPKPK(Agrawirtschaft,:%New," = "," and") */
        Agrawirtschaft.Agrar_ID = :new.Agrar_ID;
end;
create trigger tU_Agrawirtschaft after UPDATE on Agrawirtschaft for each row
-- UPDATE trigger on Agrawirtschaft
declare numrows INTEGER;
begin
    /* Region R/13 Agrawirtschaft ON CHILD UPDATE SET NULL */
    update Agrawirtschaft
      set
        /* %SetFK(Agrawirtschaft,NULL) */
        Agrawirtschaft.Regions_ID = NULL
      where
```

```
        not exists (
          select * from Region
            where
              /* %JoinFKPK(:%New,Region," = "," and") */
              :new.Regions_ID = Region.Regions_ID
        ) and
        /* %JoinPKPK(Agrawirtschaft,:%New," = "," and") */
        Agrawirtschaft.Agrar_ID = :new.Agrar_ID;
end;
create trigger tI_Industrie after INSERT on Industrie for each row
-- INSERT trigger on Industrie
declare numrows INTEGER;
begin
    /* Region R/12 Industrie ON CHILD INSERT SET NULL */
    update Industrie
      set
        /* %SetFK(Industrie,NULL) */
        Industrie.Regions_ID = NULL
      where
        not exists (
          select * from Region
            where
              /* %JoinFKPK(:%New,Region," = "," and") */
              :new.Regions_ID = Region.Regions_ID
        ) and
        /* %JoinPKPK(Industrie,:%New," = "," and") */
        Industrie.Industri_ID = :new.Industri_ID;
end;
create trigger tU_Industrie after UPDATE on Industrie for each row
-- UPDATE trigger on Industrie
declare numrows INTEGER;
begin
    /* Region R/12 Industrie ON CHILD UPDATE SET NULL */
    update Industrie
      set
        /* %SetFK(Industrie,NULL) */
        Industrie.Regions_ID = NULL
      where
        not exists (
          select * from Region
            where
              /* %JoinFKPK(:%New,Region," = "," and") */
              :new.Regions_ID = Region.Regions_ID
        ) and
        /* %JoinPKPK(Industrie,:%New," = "," and") */
        Industrie.Industri_ID = :new.Industri_ID;
end;
create trigger tD_Stadt after DELETE on Stadt for each row
-- DELETE trigger on Stadt
declare numrows INTEGER;
begin
    /* Stadt R/33 Adresse_Stadt ON PARENT DELETE RESTRICT */
```

```
    select count(*) into numrows
      from Adresse_Stadt
      where
        /*  %JoinFKPK(Adresse_Stadt,:%Old," = "," and") */
        Adresse_Stadt.Stadt_ID = :old.Stadt_ID;
    if (numrows > 0)
    then
      raise_application_error(
        -20001,
        'Cannot DELETE Stadt because Adresse_Stadt exists.'
      );
    end if;
end;
create trigger tI_Stadt after INSERT on Stadt for each row
-- INSERT trigger on Stadt
declare numrows INTEGER;
begin
    /* Region R/11 Stadt ON CHILD INSERT SET NULL */
    update Stadt
      set
        /* %SetFK(Stadt,NULL) */
        Stadt.Regions_ID = NULL
      where
        not exists (
          select * from Region
            where
              /* %JoinFKPK(:%New,Region," = "," and") */
              :new.Regions_ID = Region.Regions_ID
        ) and
        /* %JoinPKPK(Stadt,:%New," = "," and") */
        Stadt.Stadt_ID = :new.Stadt_ID;
end;
create trigger tU_Stadt after UPDATE on Stadt for each row
-- UPDATE trigger on Stadt
declare numrows INTEGER;
begin
  /* Stadt R/33 Adresse_Stadt ON PARENT UPDATE RESTRICT */
  if
    /* %JoinPKPK(:%Old,:%New," <> "," or ") */
    :old.Stadt_ID <> :new.Stadt_ID
  then
    select count(*) into numrows
      from Adresse_Stadt
      where
        /*  %JoinFKPK(Adresse_Stadt,:%Old," = "," and") */
        Adresse_Stadt.Stadt_ID = :old.Stadt_ID;
    if (numrows > 0)
    then
      raise_application_error(
        -20005,
        'Cannot UPDATE Stadt because Adresse_Stadt exists.'
      );
```

```
    end if;
  end if;
/* Region R/11 Stadt ON CHILD UPDATE SET NULL */
    update Stadt
      set
        /* %SetFK(Stadt,NULL) */
        Stadt.Regions_ID = NULL
      where
        not exists (
          select * from Region
            where
              /* %JoinFKPK(:%New,Region," = "," and") */
              :new.Regions_ID = Region.Regions_ID
        ) and
        /* %JoinPKPK(Stadt,:%New," = "," and") */
        Stadt.Stadt_ID = :new.Stadt_ID;
end;
create trigger tD_Region after DELETE on Region for each row
-- DELETE trigger on Region
declare numrows INTEGER;
begin
    /* Region R/14 Viehwirtschaft ON PARENT DELETE SET NULL */
    update Viehwirtschaft
      set
        /* %SetFK(Viehwirtschaft,NULL) */
        Viehwirtschaft.Regions_ID = NULL
      where
        /* %JoinFKPK(Viehwirtschaft,:%Old," = "," and") */
        Viehwirtschaft.Regions_ID = :old.Regions_ID;
    /* Region R/13 Agrawirtschaft ON PARENT DELETE SET NULL */
    update Agrawirtschaft
      set
        /* %SetFK(Agrawirtschaft,NULL) */
        Agrawirtschaft.Regions_ID = NULL
      where
        /* %JoinFKPK(Agrawirtschaft,:%Old," = "," and") */
        Agrawirtschaft.Regions_ID = :old.Regions_ID;
/* Region R/12 Industrie ON PARENT DELETE SET NULL */
    update Industrie
      set
        /* %SetFK(Industrie,NULL) */
        Industrie.Regions_ID = NULL
      where
        /* %JoinFKPK(Industrie,:%Old," = "," and") */
        Industrie.Regions_ID = :old.Regions_ID;
/* Region R/11 Stadt ON PARENT DELETE SET NULL */
    update Stadt
      set
        /* %SetFK(Stadt,NULL) */
        Stadt.Regions_ID = NULL
      where
        /* %JoinFKPK(Stadt,:%Old," = "," and") */
```

```
        Stadt.Regions_ID = :old.Regions_ID;
/* Region R/3 Fluss_Region ON PARENT DELETE RESTRICT */
    select count(*) into numrows
      from Fluss_Region
      where
        /*  %JoinFKPK(Fluss_Region,:%Old," = "," and") */
        Fluss_Region.Regions_ID = :old.Regions_ID;
    if (numrows > 0)
    then
      raise_application_error(
        -20001,
        'Cannot DELETE Region because Fluss_Region exists.'
      );
    end if;
end;
create trigger tU_Region after UPDATE on Region for each row
-- UPDATE trigger on Region
declare numrows INTEGER;
begin
  /* Region R/14 Viehwirtschaft ON PARENT UPDATE SET NULL */
  if
    /* %JoinPKPK(:%Old,:%New," <> "," or " */
    :old.Regions_ID <> :new.Regions_ID
  then
    update Viehwirtschaft
      set
        /* %SetFK(Viehwirtschaft,NULL) */
        Viehwirtschaft.Regions_ID = NULL
      where
        /* %JoinFKPK(Viehwirtschaft,:%Old," = ","","") */
        Viehwirtschaft.Regions_ID = :old.Regions_ID;
  end if;
  /* Region R/13 Agrawirtschaft ON PARENT UPDATE SET NULL */
  if
    /* %JoinPKPK(:%Old,:%New," <> "," or " */
    :old.Regions_ID <> :new.Regions_ID
  then
    update Agrawirtschaft
      set
        /* %SetFK(Agrawirtschaft,NULL) */
        Agrawirtschaft.Regions_ID = NULL
      where
        /* %JoinFKPK(Agrawirtschaft,:%Old," = ","","") */
        Agrawirtschaft.Regions_ID = :old.Regions_ID;
  end if;
  /* Region R/12 Industrie ON PARENT UPDATE SET NULL */
  if
    /* %JoinPKPK(:%Old,:%New," <> "," or " */
    :old.Regions_ID <> :new.Regions_ID
  then
    update Industrie
      set
```

```
        /* %SetFK(Industrie,NULL) */
        Industrie.Regions_ID = NULL
      where
        /* %JoinFKPK(Industrie,:%Old," = ","") */
        Industrie.Regions_ID = :old.Regions_ID;
  end if;
  /* Region R/11 Stadt ON PARENT UPDATE SET NULL */
  if
    /* %JoinPKPK(:%Old,:%New," <> "," or " */
    :old.Regions_ID <> :new.Regions_ID
  then
    update Stadt
      set
        /* %SetFK(Stadt,NULL) */
        Stadt.Regions_ID = NULL
      where
        /* %JoinFKPK(Stadt,:%Old," = ","") */
        Stadt.Regions_ID = :old.Regions_ID;
  end if;
/* Region R/3 Fluss_Region ON PARENT UPDATE RESTRICT */
  if
    /* %JoinPKPK(:%Old,:%New," <> "," or ") */
    :old.Regions_ID <> :new.Regions_ID
  then
    select count(*) into numrows
      from Fluss_Region
      where
        /*  %JoinFKPK(Fluss_Region,:%Old," = "," and") */
        Fluss_Region.Regions_ID = :old.Regions_ID;
    if (numrows > 0)
    then
      raise_application_error(
        -20005,
        'Cannot UPDATE Region because Fluss_Region exists.'
      );
    end if;
  end if;
end;
create trigger tD_Gewaesser after DELETE on Gewaesser for each row
-- DELETE trigger on Gewaesser
declare numrows INTEGER;
begin
/* Gewaesser R/22 Gewaesser_Fluss ON PARENT DELETE RESTRICT */
    select count(*) into numrows
      from Gewaesser_Fluss
      where
        /*  %JoinFKPK(Gewaesser_Fluss,:%Old," = "," and") */
        Gewaesser_Fluss.Gewaesser_ID = :old.Gewaesser_ID;
    if (numrows > 0)
    then
      raise_application_error(
        -20001,
```

```
        'Cannot DELETE Gewaesser because Gewaesser_Fluss exists.'
      );
    end if;
end;
create trigger tU_Gewaesser after UPDATE on Gewaesser for each row
-- UPDATE trigger on Gewaesser
declare numrows INTEGER;
begin
/* Gewaesser R/22 Gewaesser_Fluss ON PARENT UPDATE RESTRICT */
  if
    /* %JoinPKPK(:%Old,:%New," <> "," or ") */
    :old.Gewaesser_ID <> :new.Gewaesser_ID
  then
    select count(*) into numrows
      from Gewaesser_Fluss
      where
        /*  %JoinFKPK(Gewaesser_Fluss,:%Old," = "," and") */
        Gewaesser_Fluss.Gewaesser_ID = :old.Gewaesser_ID;
    if (numrows > 0)
    then
      raise_application_error(
        -20005,
        'Cannot UPDATE Gewaesser because Gewaesser_Fluss exists.'
      );
    end if;
  end if;
end;
/
```

C 2 SQL - Skript kleines Flussmodell

```
DROP TABLE Lage_am_Fluss CASCADE CONSTRAINTS;
CREATE TABLE Lage_am_Fluss (
       Region_ID            INTEGER NOT NULL,
       Name                 VARCHAR2(30) NOT NULL,
       Fluss_ID             INTEGER NOT NULL,
       Flusskm              NUMBER NULL
);
ALTER TABLE Lage_am_Fluss
       ADD  ( PRIMARY KEY (Region_ID, Name, Fluss_ID) ) ;
DROP TABLE Messwert CASCADE CONSTRAINTS;
CREATE TABLE Messwert (
       Messwert_ID          INTEGER NOT NULL,
       Messpunkt_ID         INTEGER NULL,
       Messzeitpunkt        VARCHAR2(20) NULL,
       Pegelstand_in_m      FLOAT NULL
);
ALTER TABLE Messwert
       ADD  ( PRIMARY KEY (Messwert_ID) ) ;
DROP TABLE Messpunkt CASCADE CONSTRAINTS;
CREATE TABLE Messpunkt (
       Messpunkt_ID         INTEGER NOT NULL,
       Fluss_ID             INTEGER NULL,
       Laengengrad          VARCHAR2(10) NULL,
       Breitengrad          VARCHAR2(10) NULL,
       Einrichtungszeitpunk DATE NULL
);
ALTER TABLE Messpunkt
       ADD  ( PRIMARY KEY (Messpunkt_ID) ) ;

DROP TABLE Fluss_Region CASCADE CONSTRAINTS;
CREATE TABLE Fluss_Region (
       Region_ID            INTEGER NOT NULL,
       Fluss_ID             INTEGER NOT NULL,
       Name                 VARCHAR2(30) NULL
);
DROP TABLE Fluss CASCADE CONSTRAINTS;
CREATE TABLE Fluss (
       Fluss_ID             INTEGER NOT NULL,
       Name                 VARCHAR2(20) NULL,
       Laenge_in_km         FLOAT NULL,
       Quellgebiet          VARCHAR2(30) NULL,
       Muendungsgewaesser   VARCHAR2(20) NULL
);
ALTER TABLE Fluss
       ADD  ( PRIMARY KEY (Fluss_ID) ) ;
DROP TABLE Region CASCADE CONSTRAINTS;
CREATE TABLE Region (
       Region_ID            INTEGER NOT NULL,
```

```
       Name                 VARCHAR2(30) NOT NULL,
       Kategorie            VARCHAR2(20) NULL
                                  CHECK (Kategorie IN ('Ort', 'Industrie',
'Agrawirtschaft', 'Viehwirtschaft')),
       Flaeche_in_qkm       FLOAT NULL
);
ALTER TABLE Region
       ADD  ( PRIMARY KEY (Region_ID, Name) ) ;
ALTER TABLE Lage_am_Fluss
       ADD  ( FOREIGN KEY (Fluss_ID)
                           REFERENCES Fluss ) ;
ALTER TABLE Lage_am_Fluss
       ADD  ( FOREIGN KEY (Region_ID, Name)
                           REFERENCES Region ) ;
ALTER TABLE Messwert
       ADD  ( FOREIGN KEY (Messpunkt_ID)
                           REFERENCES Messpunkt ) ;
ALTER TABLE Messpunkt
       ADD  ( FOREIGN KEY (Fluss_ID)
                           REFERENCES Fluss ) ;
ALTER TABLE Fluss_Region
       ADD  ( FOREIGN KEY (Region_ID, Name)
                           REFERENCES Region ) ;
ALTER TABLE Fluss_Region
       ADD  ( FOREIGN KEY (Fluss_ID)
                           REFERENCES Fluss ) ;
ALTER TABLE Fluss
       ADD  ( FOREIGN KEY (Fluss_ID)
                           REFERENCES Fluss ) ;
```

C 3 SQL - Skript Daten für kleines Flussmodell

```
Insert into Fluss (Fluss_ID, Name, Laenge_in_km, Quellgebiet, Muendungsgewaesser)
        values(1, 'Rhein', 1250, 'Schweiz', 'Nordsee');
Insert into Fluss Fluss (Fluss_ID, Name, Laenge_in_km, Quellgebiet, Muendungsgewaesser)
        values(2, 'Mosel', 250, 'Deutschland', 'Rhein');
Insert into Fluss Fluss (Fluss_ID, Name, Laenge_in_km, Quellgebiet, Muendungsgewaesser)
        values(3, 'Donau', 2500, '', 'Schwarzemeer');

Insert into Region (Region_ID, Name, Kategorie, Flaeche_in_qkm)
         values(1, 'Köln', 'Ort', 67);
Insert into Region (Region_ID, Name, Kategorie, Flaeche_in_qkm)
         values(2, 'Bonn', 'Ort', 50);
Insert into Region (Region_ID, Name, Kategorie, Flaeche_in_qkm)
         values(3, 'Wien', 'Ort', 72);

Insert into Fluss_Region (Region_ID, Fluss_ID)
            values (1, 1);
Insert into Fluss_Region (Region_ID, Fluss_ID)
            values (2, 1);
Insert into Fluss_Region (Region_ID, Fluss_ID)
            values (3, 3);

Insert into Messpunkt (Messpunkt_ID, Fluss_ID, Laengengrad, Breitengrad, EINRICHTUNGSZEITPUNK)
          values (1, 1, '3°O', '4°W', '12.12.2001');

Insert into Messwert (Messwert_ID, Messpunkt_ID, Messzeitpunkt, Pegelstand_in_m)
          values (1, 1, '12.12.01 13.00', 4.10);
Insert into Messwert (Messwert_ID, Messpunkt_ID, Messzeitpunkt, Pegelstand_in_m)
          values (2, 1, '12.12.01 13.10', 4.11);
Insert into Messwert (Messwert_ID, Messpunkt_ID, Messzeitpunkt, Pegelstand_in_m)
          values (3, 1, '12.12.01 13.20', 4.12);
Insert into Messwert (Messwert_ID, Messpunkt_ID, Messzeitpunkt, Pegelstand_in_m)
          values (4, 1, '12.12.01 13.30', 4.13);
Insert into Messwert (Messwert_ID, Messpunkt_ID, Messzeitpunkt, Pegelstand_in_m)
          values (5, 1, '12.12.01 13.40', 4.14);
Insert into Messwert (Messwert_ID, Messpunkt_ID, Messzeitpunkt, Pegelstand_in_m)
          values (6, 1, '12.12.01 13.50', 4.14);
Insert into Messwert (Messwert_ID, Messpunkt_ID, Messzeitpunkt, Pegelstand_in_m)
          values (7, 1, '12.12.01 14.00', 4.13);
Insert into Messwert (Messwert_ID, Messpunkt_ID, Messzeitpunkt, Pegelstand_in_m)
          values (8, 1, '12.12.01 14.10', 4.12);
Insert into Messwert (Messwert_ID, Messpunkt_ID, Messzeitpunkt, Pegelstand_in_m)
          values (9, 1, '12.12.01 14.20', 4.11);
Insert into Messwert (Messwert_ID, Messpunkt_ID, Messzeitpunkt, Pegelstand_in_m)
          values (10, 1, '12.12.01 14.30', 4.10);
Insert into Messwert (Messwert_ID, Messpunkt_ID, Messzeitpunkt, Pegelstand_in_m)
          values (11, 1, '12.12.01 14.40', 4.09);
Insert into Messwert (Messwert_ID, Messpunkt_ID, Messzeitpunkt, Pegelstand_in_m)
          values (12, 1, '12.12.01 14.50', 4.08);
Insert into Messwert (Messwert_ID, Messpunkt_ID, Messzeitpunkt, Pegelstand_in_m)
          values (13, 1, '12.12.01 15.00', 4.07);
```

D Programmcode

Das Programm wurde mit NetBeans 3.3.1 entwickelt, entsprechend werden spezielle API's von Netbeans verwendet.
Deweiteren muss das jar-File Class12.jar, zu finden bei Oracle, in den Pfad aufgenommen werden, damit die Datenbankanbindung einwandfrei funktioniert. Außerdem+ muss die URL der Datenbank angepasst werden.

D 1 dbt_Starter.java

```
/*
 * dbt_Starter.java
 *
 * DataBankTool V0.02 (11.4.2k2)
 *
 * (c) Reinhard S. Zoellner
 * erstellt im Rahmen der Diplomarbeit
 *
 * Created on 11. April 2002, 10:34
 */

package de.ducesoft.rsz.diparbeit;

/**
 * dbt-Starter ist der Starter fuer das DataBankTool.
 * Hier kann ausgewaehlt werden auf welches DBSytem
 * Oracle oder MySql zugegriffen werden soll.
 *
 * @author  Reinhard S. Zöllner
 */
public class dbt_Starter extends javax.swing.JFrame {
    /** Creates new form DBT_Starter */
    public dbt_Starter() {
        super("DataBankTool V0.02");
                                // durch super werden Methoden der
                                // Elternklasse geerbt, hier insbesondere
                                // das bezeichnen der Fenster.
        initComponents();
    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    private void initComponents() {//GEN-BEGIN:initComponents
        jLabel2 = new javax.swing.JLabel();
```

```
jPanel1 = new javax.swing.JPanel();
jButton1 = new javax.swing.JButton();
jButton2 = new javax.swing.JButton();
jLabel1 = new javax.swing.JLabel();
jButton3 = new javax.swing.JButton();
jLabel3 = new javax.swing.JLabel();

addWindowListener(new java.awt.event.WindowAdapter() {
    public void windowClosing(java.awt.event.WindowEvent evt) {
        exitForm(evt);
    }
});

jLabel2.setVerticalAlignment(javax.swing.SwingConstants.TOP);
jLabel2.setText("DataBankTool V0.02");
jLabel2.setToolTipText("Version von DBT");
jLabel2.setFont(new java.awt.Font("Arial", 1, 18));
getContentPane().add(jLabel2, java.awt.BorderLayout.NORTH);

jPanel1.setLayout(new org.netbeans.lib.awtextra.AbsoluteLayout());

jButton1.setToolTipText("W\u00e4hlt Oracle als DBS aus");
jButton1.setText("Oracle 8i DBS");
jButton1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton1ActionPerformed(evt);
    }
});

jButton1.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseClicked(java.awt.event.MouseEvent evt) {
        oraAuswahl(evt);
    }
});

jPanel1.add(jButton1, new
      org.netbeans.lib.awtextra.AbsoluteConstraints(100, 80, 170, 40));

jButton2.setToolTipText("w\u00e4hlt MySQL als DBS aus");
jButton2.setText("MySQL DBS");
jButton2.setMaximumSize(new java.awt.Dimension(112, 26));
jButton2.setMinimumSize(new java.awt.Dimension(112, 26));
jButton2.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton2ActionPerformed(evt);
    }
});

jButton2.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseClicked(java.awt.event.MouseEvent evt) {
        mySQLAusgewaehlt(evt);
    }
```

```
        });

        jPanel1.add(jButton2, new
             org.netbeans.lib.awtextra.AbsoluteConstraints(100, 130, 170, 40));

        jLabel1.setText("W\u00e4hlen Sie auf welches DB-System zugegriffen
         werden soll.");
        jPanel1.add(jLabel1, new
           org.netbeans.lib.awtextra.AbsoluteConstraints(20, 40, -1, -1));

        jButton3.setToolTipText("Zeigt eine Information an");
        jButton3.setText("Informationen \u00fcber DBT");
        jButton3.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                info(evt);
            }
        });

        jPanel1.add(jButton3, new
             org.netbeans.lib.awtextra.AbsoluteConstraints(100, 180, 170, 40));

        getContentPane().add(jPanel1, java.awt.BorderLayout.CENTER);

        jLabel3.setText("(c) R.S.Z\u00f6llner, DuceSoft 2k2");
        jLabel3.setToolTipText("(c)");
        getContentPane().add(jLabel3, java.awt.BorderLayout.SOUTH);

        pack();
    }//GEN-END:initComponents

    private void info(java.awt.event.MouseEvent evt) {
        //GEN-FIRST:event_info
        // Es ist moeglich mittels IE HTML-Seiten anzuzeigen, dazu werden diese
        // mittels Runtime.getRuntime().exec(....); aufgerufen. siehe hier.
        // Nachteil : der Pfad ist nicht dynamisch.
        try {
           Runtime.getRuntime().exec("c:\\Programme\\internet
                               explorer\\iexplore.exe c:\\dbt\\dbt.html");
       } catch (Exception ex) {
           System.out.println("Fehler : " + ex );
       }
    }//GEN-LAST:event_info

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt)
    {//GEN-FIRST:event_jButton1ActionPerformed
        // Add your handling code here:
    }//GEN-LAST:event_jButton1ActionPerformed

    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt)
    {//GEN-FIRST:event_jButton2ActionPerformed
        // Add your handling code here:
    }//GEN-LAST:event_jButton2ActionPerformed
```

```
    private void mySQLAusgewaehlt(java.awt.event.MouseEvent evt) {
        //GEN-FIRST:event_mySQLAusgewaehlt
        // Derzeit noch nicht implementiert
        // Sorgt dann fuer die ANbindung von MySQL-DBS
    }//GEN-LAST:event_mySQLAusgewaehlt

    private void oraAuswahl(java.awt.event.MouseEvent evt) {
    //GEN-FIRST:event_oraAuswahl
        // zum aufraeumen, bzw. der besseren Uebersicht Fenster schliessen
        sf.setVisible(false);
        // Nachfolgendes Fenster aufrufen
        new anmeldung().show();
    }//GEN-LAST:event_oraAuswahl

    /** Exit the Application */
    private void exitForm(java.awt.event.WindowEvent evt) {
     //GEN-FIRST:event_exitForm
        System.exit(0);
    }//GEN-LAST:event_exitForm

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        // setVisible() wird deswegen genutzt um das fenster zu schliessen
        // wenn es nicht mehr benoetigt wird.
        // Desweiteren ist show() deprecated !
        sf = new dbt_Starter();
        sf.setVisible(true);
    }


    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JButton jButton3;
    private javax.swing.JButton jButton2;
    private javax.swing.JButton jButton1;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JPanel jPanel1;
    // End of variables declaration//GEN-END:variables
    // Variable die fuer eigenen Code benoetigt werden
    private static dbt_Starter sf;      // sf = Start Fenster
    // Ende der Variablen decclaration
}
```

D 2 anmeldung.java

```
/*
 * anmeldung.java
 *
 * DataBankTool V0.02 (11.4.2k2)
 *
 * (c) Reinhard S. Zoellner
 * erstellt im Rahmen der Diplomarbeit
 *
 * Created on 26. Februar 2002, 16:41
 */

package de.ducesoft.rsz.diparbeit;

import java.sql.*;
import java.io.*;

/**
 * In dieser Komponente wird sichergestelt das die Anmeldung
 * an das DBS korrekt ausgefuehrt wird.
 *
 * @author  Reinhard Zoellner
 */
public class anmeldung extends javax.swing.JFrame {

    /** Creates new form anmeldung */
    public anmeldung() {
        super("DataBankTool V0.02");
        initComponents();
    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    private void initComponents() {//GEN-BEGIN:initComponents
        jLabel1 = new javax.swing.JLabel();
        jButton1 = new javax.swing.JButton();
        jLabel2 = new javax.swing.JLabel();
        jLabel3 = new javax.swing.JLabel();
        jTextField1 = new javax.swing.JTextField();
        jPasswordField1 = new javax.swing.JPasswordField();

        getContentPane().setLayout(new
                org.netbeans.lib.awtextra.AbsoluteLayout());

        addWindowListener(new java.awt.event.WindowAdapter() {
            public void windowClosing(java.awt.event.WindowEvent evt) {
                exitForm(evt);
            }
```

```
        });

        jLabel1.setText("Datenbankanmeldung");
        jLabel1.setFont(new java.awt.Font("Dialog", 1, 18));
        getContentPane().add(jLabel1,new
            org.eans.lib.awtextra.AbsoluteConstraints(90, 30, 220, 30));

        jButton1.setText("anmelden");
        jButton1.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                verbinden(evt);
            }
        });

        getContentPane().add(jButton1,new
              org.netbeans.lib.awtextra.AbsoluteConstraints(160, 220, -1, -1));

        jLabel2.setText("Benutzer");
        getContentPane().add(jLabel2,new
              org.netbeans.lib.awtextra.AbsoluteConstraints(40, 110, -1, -1));

        jLabel3.setText("Passwort");
        getContentPane().add(jLabel3,new
               org.netbeans.lib.awtextra.AbsoluteConstraints(40, 140, -1, -1));

        getContentPane().add(jTextField1,new
              org.netbeans.lib.awtextra.AbsoluteConstraints(160, 110, 90, -1));

        getContentPane().add(jPasswordField1,new
              org.netbeans.lib.awtextra.AbsoluteConstraints(160, 140, 90, -1));

        pack();
    }//GEN-END:initComponents

    private void verbinden(java.awt.event.MouseEvent evt)
        {//GENFIRST:event_verbinden
        // Add your handling code here:
        String user = jTextField1.getText().trim();
        String passwd = jPasswordField1.getText().trim();
        oraZugriffsDaten.setMyURL("jdbc:oracle:thin:@localhost:1521:rsz");
        // 2481
        Connection v_connection = null;
        Statement v_statement= null;
        Integer v_int;
        try {
        // Oracle JDBC Treiber laden bzw. den DriverManager bekannt machen
        Class.forName("oracle.jdbc.driver.OracleDriver");
        }
        catch (java.lang.ClassNotFoundException e) {
        System.err.println("Class Not Found Exception : " + e.getMessage());
        }
```

```
            try {
            v_connection = DriverManager.getConnection
                           (oraZugriffsDaten.getMyURL(), user, passwd);
            v_statement = v_connection.createStatement();
            v_statement.close();
            v_connection.close();
            }
            catch(SQLException e) {
            String fehler = e.getMessage();
            System.out.println(fehler);
            System.out.println(fehler.substring(6,35));
            if(fehler.substring(0,9).equals("ORA-01017")) {
                new fpofu(new javax.swing.JFrame(), true).show();
                System.exit(0);
            }
            if(fehler.substring(6,35).equals("Benutzer- oder Kennwortangabe")){
                 new fpfu(new javax.swing.JFrame(), true).show();
                 err = 1;
            }
            }
            if (err == 0) {
                oraZugriffsDaten.setMyUser(user);
                oraZugriffsDaten.setMyPasswd(passwd);
                new hauptmenu().show();
            } else {
            err = 0;
            }
        }//GEN-LAST:event_verbinden

        /** Exit the Application */
        private void exitForm(java.awt.event.WindowEvent evt)
            {//GEN-FIRST:event_exitForm
            System.exit(0);
        }//GEN-LAST:event_exitForm

        // Variables declaration - do not modify//GEN-BEGIN:variables
        private javax.swing.JPasswordField jPasswordField1;
        private javax.swing.JTextField jTextField1;
        private javax.swing.JButton jButton1;
        private javax.swing.JLabel jLabel3;
        private javax.swing.JLabel jLabel2;
        private javax.swing.JLabel jLabel1;
        // End of variables declaration//GEN-END:variables
        private int err = 0;
    }
```

D 3 oraZugriffsDaten.java

```
/*
 * oraZugriffsDaten.java
 *
 * DataBankTool V0.02 (11.4.2k2)
 *
 * (c) Reinhard S. Zoellner
 * erstellt im Rahmen der Diplomarbeit
 *
 * Created on 11. April 2002, 17:19
 */

package de.ducesoft.rsz.diparbeit;

/**
 * Verwaltet die Zugriffsdaten fuer die Datenbankzugriffe.
 * D.h. Es werden wenn bekann ist gespeichert :
 *       - Username
 *       - Password
 *       - SID
 *       - URL
 *       - zu nutzende Tabelle
 *
 * @author  Reinhard S. Zöllner
 */
public class oraZugriffsDaten {

    /** Creates a new instance of OraZugiffsDaten */
    public oraZugriffsDaten() {
        myurl = null;
        myuser = null;
        mypasswd = null;
        mysid = null;
        mytabelle = null;
    }

    protected static void setMyURL(String url) {
        myurl = url;
    }

    protected static void setMyUser(String user) {
        myuser = user;
    }

    protected static void setMyPasswd(String passwd) {
        mypasswd = passwd;
    }

    protected static void setMyTabelle(String tabelle) {
```

```
        mytabelle = tabelle;
    }

    protected static String getMyURL() {
        return myurl;
    }

    protected static String getMyUser() {
        return myuser;
    }

    protected static String getMyPasswd() {
        return mypasswd;
    }

    protected static String getMyTabelle() {
        return mytabelle;
    }

    protected static void setMySID(String sid) {
        mysid = sid;
    }

    protected static String getMySID() {
        return mysid;
    }

    private static String myurl;
    private static String myuser;
    private static String mypasswd;
    private static String mytabelle;
    private static String mysid;

}
```

D 4 hauptmenu.java

```
/*
 * hauptmenu.java
 *
 * DataBankTool V0.02 (11.4.2k2)
 *
 * (c) Reinhard S. Zoellner
 * erstellt im Rahmen der Diplomarbeit
 *
 * Created on 28. Februar 2002, 14:55
 */

package de.ducesoft.rsz.diparbeit;

import java.util.*;
import java.sql.*;
import java.io.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;

/**
 * Stellt die Hauptauswahl dar. Man kann waehlen zwischen
 * Daten eingeben, ansehen und Beenden.
 *
 * @author  Reinhard Zoellner
 */
public class hauptmenu extends javax.swing.JFrame implements
       ListSelectionListener {

    /** Creates new form hauptmenu */
    public hauptmenu() {
        super("DataBankTool V0.02");
        initComponents();
    }

    private void ansehen(java.awt.event.MouseEvent evt) {
        //GEN-FIRST:event_ansehen
        String tabelle = oraZugriffsDaten.getMyTabelle();
        if(tabelle == null) {
            err = 1;
            new eta(new javax.swing.JFrame(), true).show();
        }
        if ( err == 0) {
            // Tabell ist korrekt ausgewaehlt
            // Aufruf gewuenschten Daten zum ansehen.
            new lookData().show();
        }
```

```
}//GEN-LAST:event_ansehen
private void aendern(java.awt.event.MouseEvent evt) {
    String tabelle = oraZugriffsDaten.getMyTabelle();
    if(tabelle == null) {
        err = 1;
        new eta(new javax.swing.JFrame(), true).show();
    }
    if ( err == 0) {
        // Tabell ist korrekt ausgewaehlt
        // Aufruf den Aufbau der gewuenschten Tabelle ansehen.
        new changeData().show();
    }
}

private void aufbauansehen(java.awt.event.MouseEvent evt) {
    String tabelle = oraZugriffsDaten.getMyTabelle();
    if(tabelle == null) {
        err = 1;
        new eta(new javax.swing.JFrame(), true).show();
    }
    if ( err == 0) {
        // Tabell ist korrekt ausgewaehlt
        // Aufruf den Aufbau der gewuenschten Tabelle ansehen.
        new taa().show();
    }
}

private void eingeben(java.awt.event.MouseEvent evt) {
//GEN-FIRST:event_eingeben
    String tabelle = oraZugriffsDaten.getMyTabelle();
//    int err = 0;
    if(tabelle == null) {
        err = 1;
        new eta(new javax.swing.JFrame(), true).show();
    }
    if ( err == 0) {
        // Tabell ist korrekt ausgewaehlt
        new editData().show();
    }
}//GEN-LAST:event_eingeben

private void ende(java.awt.event.MouseEvent evt) {
 //GEN-FIRST:event_ende
    System.exit(0);
}//GEN-LAST:event_ende
/*
 * url    := korekte url zum einloggen in ein DBS
 * user   := ueberpruefter User fuer ein DBS
 * passwd := Passwort fuer den User
 */
private void initComponents() {
```

```
jLabel1 = new javax.swing.JLabel();
jLabel2 = new javax.swing.JLabel();
jLabel3 = new javax.swing.JLabel();
jButton1 = new javax.swing.JButton();
jButton2 = new javax.swing.JButton();
jButton3 = new javax.swing.JButton();
jButton4 = new javax.swing.JButton();
jButton5 = new javax.swing.JButton();

getContentPane().setLayout(new
                    org.netbeans.lib.awtextra.AbsoluteLayout());

addWindowListener(new java.awt.event.WindowAdapter() {
    public void windowClosing(java.awt.event.WindowEvent evt) {
        exitForm(evt);
    }
});

jLabel1.setText("Sie sind eingelogt als :");
getContentPane().add(jLabel1,new
      org.netbeans.lib.awtextra.AbsoluteConstraints(10, 10, -1, -1));

jLabel2.setText(oraZugriffsDaten.getMyUser());
getContentPane().add(jLabel2,new
     org.netbeans.lib.awtextra.AbsoluteConstraints(150, 10, -1, -1));

jLabel3.setText("Es existieren volgende Tabellen :");
getContentPane().add(jLabel3,new
       org.netbeans.lib.awtextra.AbsoluteConstraints(10, 60, -1, -1));

jButton1.setText("Daten eingeben");
jButton1.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseClicked(java.awt.event.MouseEvent evt) {
        eingeben(evt);
    }
});
getContentPane().add(jButton1,new
      org.netbeans.lib.awtextra.AbsoluteConstraints(270, 130, -1, -1));

jButton5.setText("Daten aendern");
jButton5.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseClicked(java.awt.event.MouseEvent evt) {
        aendern(evt);
    }
});
getContentPane().add(jButton5,new
       org.netbeans.lib.awtextra.AbsoluteConstraints(270, 90, -1, -1));

jButton2.setText("Daten ansehen");
jButton2.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseClicked(java.awt.event.MouseEvent evt) {
        ansehen(evt);
```

```
        }
    });
    getContentPane().add(jButton2,new
         org.netbeans.lib.awtextra.AbsoluteConstraints(270, 170, -1, -1));

    jButton4.setText("Tabellenaufbau ansehen");
    jButton4.addMouseListener(new java.awt.event.MouseAdapter() {
        public void mouseClicked(java.awt.event.MouseEvent evt) {
            aufbauansehen(evt);
        }
    });
    getContentPane().add(jButton4,new
          org.netbeans.lib.awtextra.AbsoluteConstraints(270, 210, -1, -1));

    jButton3.setText("Beenden");
    jButton3.addMouseListener(new java.awt.event.MouseAdapter() {
        public void mouseClicked(java.awt.event.MouseEvent evt) {
            ende(evt);
        }
    });
    getContentPane().add(jButton3,new
          org.netbeans.lib.awtextra.AbsoluteConstraints(270, 250, -1, -1));

    Vector wordVector = new Vector();
    Connection v_connection = null;
    Statement v_statement= null;
    try {
        // Oracle JDBC Treiber laden bzw. den DriverManager bekannt machen
       Class.forName("oracle.jdbc.driver.OracleDriver");
    }
    catch (java.lang.ClassNotFoundException e) {
       System.err.println("Class Not Found Exception : " + e.getMessage());
    }

    try {
        v_connection = DriverManager.getConnection(
                        oraZugriffsDaten.getMyURL(),
                        oraZugriffsDaten.getMyUser(),
                        oraZugriffsDaten.getMyPasswd());
        v_statement = v_connection.createStatement();

        ResultSet v_resultset = v_statement.executeQuery(
                            "Select Table_Name from USER_TABLES");
        while (v_resultset.next()) {
            String s1 = v_resultset.getString(1);
            wordVector.add(s1);
}
v_statement.close();
v_connection.close();
}
catch(SQLException e) {
}
```

```
        jList1 = new javax.swing.JList(wordVector);
        JScrollPane scrollPane = new JScrollPane(jList1);
        JPanel p = new JPanel();
        p.add(scrollPane);
        jList1.addListSelectionListener(this);
        getContentPane().add(p,new
               org.netbeans.lib.awtextra.AbsoluteConstraints(30, 130, -1, -1));
        pack();
    }

    /*
     * valueChanged-Methode aufgrund von ListSelectionListener
     * Interface implementieren.
     */
    public void valueChanged(ListSelectionEvent evt) {
        JList source = (JList)evt.getSource();
        Object[] values = source.getSelectedValues();

        String text = "";
        for (int i = 0; i < values.length; i++) {
            String word = (String)values[i];
            text += word + " ";
        }
        // Tabelle ist ausgewaehlt somit err auf 0 setzen.
        oraZugriffsDaten.setMyTabelle(text);
        err = 0;
    }

    /** Exit the Application */
    private void exitForm(java.awt.event.WindowEvent evt) {
     //GEN-FIRST:event_exitForm
        System.exit(0);
    }//GEN-LAST:event_exitForm
    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JButton jButton3;
    private javax.swing.JButton jButton2;
    private javax.swing.JButton jButton1;
    private javax.swing.JList jList1;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel1;
    // End of variables declaration//GEN-END:variables
    private javax.swing.JButton jButton4;
    private javax.swing.JButton jButton5;
    private String ausgewaehlteTabelle = null;
    private int err = 1;            // Fehlervariable erst Tabelle auswaehlen
}
```

D 5 changeData.java

```
/*
 * changeData.java
 *
 * DataBankTool V0.02 (11.4.2k2)
 *
 * (c) Reinhard S. Zoellner
 * erstellt im Rahmen der Diplomarbeit
 *
 * Created on 11. April 2002, 12:40
 */

package de.ducesoft.rsz.diparbeit;

/**
 *
 * @author  Reinhard S. Zöllner
 */
public class changeData extends javax.swing.JFrame {

    /** Creates new form changeData */
    public changeData() {
        super("DataBankTool V0.02");
        initComponents();
    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    private void initComponents() {//GEN-BEGIN:initComponents
        jLabel1 = new javax.swing.JLabel();
        jScrollPane1 = new javax.swing.JScrollPane();
        jTable1 = new javax.swing.JTable();

        getContentPane().setLayout(new
                              org.netbeans.lib.awtextra.AbsoluteLayout());

        addWindowListener(new java.awt.event.WindowAdapter() {
            public void windowClosing(java.awt.event.WindowEvent evt) {
                exitForm(evt);
            }
        });

        jLabel1.setText(
          "W\u00e4hlen sie den Datensatz aus der ge\u00e4ndert werden soll.");
        getContentPane().add(jLabel1, new
               org.netbeans.lib.awtextra.AbsoluteConstraints(30, 30, -1, -1));
```

```
            jTable1.setModel(new javax.swing.table.DefaultTableModel(
                new Object [][] {
                    {null, null, null, null},
                    {null, null, null, null},
                    {null, null, null, null},
                    {null, null, null, null}
                },
                new String [] {
                    "Title 1", "Title 2", "Title 3", "Title 4"
                }
            ));
            jScrollPane1.setViewportView(jTable1);

            getContentPane().add(jScrollPane1, new
                    org.netbeans.lib.awtextra.AbsoluteConstraints(30, 80, -1, -1));

            pack();
        }//GEN-END:initComponents

        /** Exit the Application */
        private void exitForm(java.awt.event.WindowEvent evt) {
        //GEN-FIRST:event_exitForm
            System.exit(0);
        }//GEN-LAST:event_exitForm

        /**
         * @param args the command line arguments
         */
        public static void main(String args[]) {
            new changeData().show();
        }

        // Variables declaration - do not modify//GEN-BEGIN:variables
        private javax.swing.JScrollPane jScrollPane1;
        private javax.swing.JTable jTable1;
        private javax.swing.JLabel jLabel1;
        // End of variables declaration//GEN-END:variables

    }
```

D 6 druck.java

```
/*
 * druck.java
 *
 * DataBankTool V0.01 (5.4.2k2)
 *
 * (c) Reinhard S. Zoellner
 * erstellt im Rahmen der Diplomarbeit
 *
 * Created on 6. März 2002, 10:38
 */

package de.ducesoft.rsz.diparbeit;

/**
 *
 * @author  Reinhard Zoellner
 */
public class druck extends javax.swing.JFrame {

    /** Creates new form druck */
    public druck() {
        super("DataBankTool V0.01");
        initComponents();
    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    private void initComponents() {//GEN-BEGIN:initComponents
        jLabel1 = new javax.swing.JLabel();
        jButton1 = new javax.swing.JButton();

        getContentPane().setLayout(new
                                   org.netbeans.lib.awtextra.AbsoluteLayout());

        addWindowListener(new java.awt.event.WindowAdapter() {
            public void windowClosing(java.awt.event.WindowEvent evt) {
                exitForm(evt);
            }
        });

        jLabel1.setText("Hier k\u00f6nnte eine Druck-.Methode implementiert
                         werden.");
        getContentPane().add(jLabel1,new
               org.netbeans.lib.awtextra.AbsoluteConstraints(40, 10, 330, 40));
```

```
        jButton1.setText("zur\u00fcck");
        jButton1.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                zurueck(evt);
            }
        });

        getContentPane().add(jButton1,new
               org.netbeans.lib.awtextra.AbsoluteConstraints(150, 50, -1, -1));

        pack();
    }//GEN-END:initComponents

    private void zurueck(java.awt.event.MouseEvent evt) {
     //GEN-FIRST:event_zurueck
        setVisible(false);
        dispose();
    }//GEN-LAST:event_zurueck

    /** Exit the Application */
    private void exitForm(java.awt.event.WindowEvent evt) {
     //GEN-FIRST:event_exitForm
        System.exit(0);
    }//GEN-LAST:event_exitForm
    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JButton jButton1;
    private javax.swing.JLabel jLabel1;
    // End of variables declaration//GEN-END:variables

}
```

D 7 editData.java

```
/*
 * editData.java
 *
 * DataBankTool V0.02 (13.4.2k2)
 *
 * (c) Reinhard S. Zoellner
 * erstellt im Rahmen der Diplomarbeit
 *
 * Created on 6. März 2002, 14:39
 */

package de.ducesoft.rsz.diparbeit;

import java.util.*;
import java.sql.*;
import java.io.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;

/**
 * Daten in der entsprechenden Tabelle eingeben.
 *
 * @author  Reinhard Zoellner
 */
public class editData extends javax.swing.JFrame {

    /** Creates new form editData */
    public editData() {
        super("DataBankTool V0.02");
        initComponents();
    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    private void initComponents() {//GEN-BEGIN:initComponents
        jLabel1 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        jLabel3 = new javax.swing.JLabel();
        jButton1 = new javax.swing.JButton();
        jButton2 = new javax.swing.JButton();
        jScrollPane1 = new javax.swing.JScrollPane();
        jPanel21 = new javax.swing.JPanel();
```

```
getContentPane().setLayout(new
                    org.netbeans.lib.awtextra.AbsoluteLayout());

addWindowListener(new java.awt.event.WindowAdapter() {
    public void windowClosing(java.awt.event.WindowEvent evt) {
        exitForm(evt);
    }
});

jLabel1.setText("Hier k\u00f6nnen Daten s\u00e4tze in die Tabelle
                aufgenommen werden.");
getContentPane().add(jLabel1, new
        org.netbeans.lib.awtextra.AbsoluteConstraints(30, 30, -1, -1));

jLabel2.setText("Derzeit ist folgende Tabelle ge\u00f6ffnet :");
getContentPane().add(jLabel2, new
        org.netbeans.lib.awtextra.AbsoluteConstraints(30, 50, -1, -1));

jLabel3.setText(oraZugriffsDaten.getMyTabelle());
getContentPane().add(jLabel3, new
       org.netbeans.lib.awtextra.AbsoluteConstraints(250, 50, -1, -1));

jButton1.setText("Daten \u00fcbernehmen");
jButton1.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseReleased(java.awt.event.MouseEvent evt) {
        uebernehmen(evt);
    }
});

getContentPane().add(jButton1, new
       org.netbeans.lib.awtextra.AbsoluteConstraints(90, 270, -1, -1));

jButton2.setText("zur\u00fcck");
jButton2.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseClicked(java.awt.event.MouseEvent evt) {
        zurueck(evt);
    }
});

getContentPane().add(jButton2, new
      org.netbeans.lib.awtextra.AbsoluteConstraints(250, 270, -1, -1));

jPanel21.setLayout(new org.netbeans.lib.awtextra.AbsoluteLayout());

Connection v_connection = null;
Statement v_statement= null;

try {
    // Oracle JDBC Treiber laden bzw. den DriverManager bekannt machen
    Class.forName("oracle.jdbc.driver.OracleDriver");
}
catch (java.lang.ClassNotFoundException e) {
```

```
            System.err.println("Class Not Found Exception : " +
                               e.getMessage());
        }

        try {
            v_connection =
                 DriverManager.getConnection(oraZugriffsDaten.getMyURL(),
                                             oraZugriffsDaten.getMyUser(),
                                            oraZugriffsDaten.getMyPasswd());
            v_statement = v_connection.createStatement();

           ResultSet v_resultset = v_statement.executeQuery("Select * from " +
                       oraZugriffsDaten.getMyTabelle());

            ResultSetMetaData rsmd = v_resultset.getMetaData();
            int numberOfColumns = rsmd.getColumnCount();
            setMyNumberOfColumns(numberOfColumns);
            for (int i = 1; i <= numberOfColumns; i++) {
                JLabel bezeichner = new JLabel();
                String name = rsmd.getColumnName(i);
                bezeichner.setText(name);
                System.out.println(name);
                jPanel21.add(bezeichner, new
            org.netbeans.lib.awtextra.AbsoluteConstraints(10,i*20-20, -1, -1));
            }

            v_statement.close();
            v_connection.close();

        }
        catch(SQLException e) {
        }

        //Leeren Textfelder erzeugen zur Dateneingabe
        eintrag = new JTextField[getMyNumberOfColumns()+1];
        for (int i = 1; i <= getMyNumberOfColumns(); i++) {
            eintrag[i] = new JTextField();
            jPanel21.add(eintrag[i] , new
          org.netbeans.lib.awtextra.AbsoluteConstraints(175,i*20-20, 100, -1));
        }
        jScrollPane1.setViewportView(jPanel21);

        getContentPane().add(jScrollPane1, new
             org.netbeans.lib.awtextra.AbsoluteConstraints(40, 100, 280, 130));

        pack();
    }//GEN-END:initComponents

    private void zurueck(java.awt.event.MouseEvent evt) {
       //GEN-FIRST:event_zurueck
        setVisible(false);
        dispose();
```

```
    }//GEN-LAST:event_zurueck

    private void uebernehmen(java.awt.event.MouseEvent evt) {
      //GEN-FIRST:event_uebernehmen
        Vector wordVector = new Vector();
        Connection v_connection = null;
        Statement v_statement= null;
        String[] datentypen = new String[getMyNumberOfColumns()+1];
        String[] datenName = new String[getMyNumberOfColumns()+1];

        try {
          // Oracle JDBC Treiber laden bzw. den DriverManager bekannt machen
          Class.forName("oracle.jdbc.driver.OracleDriver");
        }
        catch (java.lang.ClassNotFoundException e) {
           System.err.println("Class Not Found Exception : " + e.getMessage());
        }

        try {
             v_connection = DriverManager.getConnection(
                                 oraZugriffsDaten.getMyURL(),
                                 oraZugriffsDaten.getMyUser(),
                                 oraZugriffsDaten.getMyPasswd());
             v_statement = v_connection.createStatement();

             ResultSet v_resultset = v_statement.executeQuery("Select * from "
                                          + oraZugriffsDaten.getMyTabelle());
             ResultSetMetaData rsmd = v_resultset.getMetaData();
             int numberOfColumns = rsmd.getColumnCount();

             for (int i = 1; i <= getMyNumberOfColumns()+1 ; i ++) {
                 datentypen[i] =  rsmd.getColumnTypeName(i);
                 datenName[i] = rsmd.getColumnName(i);
             }

     v_statement.close();
     v_connection.close();

}
catch(SQLException e) {
}

        String valueString = "";
        String dataString = "";
        for (int i = 1; i <= getMyNumberOfColumns(); i++) {

            if( ! eintrag[i].getText().equals("")) {
                // ggf. mit entsprechen den weiteren Datentypen erweitern.
                if (datentypen[i].equals("VARCHAR2")) {
                    valueString = valueString + "'" + eintrag[i].getText() +
                                  "'";
                } else {
```

```
                valueString = valueString + eintrag[i].getText();
            }
            dataString = dataString + datenName[i];

            // Bis auf an den letzten ein Komma anhaengen
            if(i != getMyNumberOfColumns()) {
                valueString = valueString + ",";
                dataString = dataString + " ,";
            }

            // ValuesString und DataString sind zusammengesetzt.
        }
    }

    // ueberpruefen ob letztes Zeichen ein Komma dann loeschen
    if(valueString.endsWith(",")) {
        valueString = valueString.substring(0,valueString.length()-1);
    }
    if(dataString.endsWith(",")) {
        dataString = dataString.substring(0,dataString.length()-1);
    }
    try {
         v_connection = DriverManager.getConnection(
                              oraZugriffsDaten.getMyURL(),
                              oraZugriffsDaten.getMyUser(),
                              oraZugriffsDaten.getMyPasswd());
         v_statement = v_connection.createStatement();

         v_statement.executeUpdate("INSERT INTO " +
                                oraZugriffsDaten.getMyTabelle() +
                                "(" + dataString + ") VALUES (" +
                                valueString + ")");

        v_statement.close();
        v_connection.close();

    }
    catch(SQLException e) {
        System.out.println(e);
    }

    System.out.println(valueString);
    System.out.println(dataString);

}//GEN-LAST:event_uebernehmen

/** Exit the Application */
private void exitForm(java.awt.event.WindowEvent evt) {
  //GEN-FIRST:event_exitForm
    System.exit(0);
}//GEN-LAST:event_exitForm
```

```
    private void setMyNumberOfColumns(int number) {
        myNumberOfColumns = number;
    }

    private int getMyNumberOfColumns() {
        return myNumberOfColumns;
    }

    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JButton jButton2;
    private javax.swing.JButton jButton1;
    private javax.swing.JPanel jPanel21;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel1;
    // End of variables declaration//GEN-END:variables
    private int myNumberOfColumns = 0;
    private JTextField[] eintrag;
}
```

D 8 eta.java

```
/*
 * eta.java
 *
 * DataBankTool V0.01 (5.4.2k2)
 *
 * (c) Reinhard S. Zoellner
 * erstellt im Rahmen der Diplomarbeit
 *
 * Created on 1. März 2002, 10:36
 */

package de.ducesoft.rsz.diparbeit;

/**
 * Dialogfenster : Erst Tabelle auswaehlen.
 *
 * @author  Reinhard Zoellner
 */
public class eta extends javax.swing.JDialog {

    /** Creates new form eta */
    public eta(java.awt.Frame parent, boolean modal) {
        super(parent, modal);
        initComponents();
    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    private void initComponents() {//GEN-BEGIN:initComponents
        jLabel1 = new javax.swing.JLabel();
        jButton1 = new javax.swing.JButton();

        getContentPane().setLayout(new
                                   org.netbeans.lib.awtextra.AbsoluteLayout());

        addWindowListener(new java.awt.event.WindowAdapter() {
            public void windowClosing(java.awt.event.WindowEvent evt) {
                closeDialog(evt);
            }
        });

        jLabel1.setText("Bitte erst eine Tabelle ausw\u00e4hlen !");
        jLabel1.setFont(new java.awt.Font("Dialog", 1, 14));
        getContentPane().add(jLabel1,new
               org.netbeans.lib.awtextra.AbsoluteConstraints(50, 10, 290, 40));
```

```
        jButton1.setText("Zur\u00fcck");
        jButton1.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mousePressed(java.awt.event.MouseEvent evt) {
                zurueck(evt);
            }
        });

        getContentPane().add(jButton1,new
               org.netbeans.lib.awtextra.AbsoluteConstraints(150, 70, -1, -1));

        pack();
    }//GEN-END:initComponents

    private void zurueck(java.awt.event.MouseEvent evt) {
     //GEN-FIRST:event_zurueck
        setVisible(false);
        dispose();
    }//GEN-LAST:event_zurueck

    /** Closes the dialog */
    private void closeDialog(java.awt.event.WindowEvent evt) {
      //GEN-FIRST:event_closeDialog
        setVisible(false);
        dispose();
    }//GEN-LAST:event_closeDialog
    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JButton jButton1;
    private javax.swing.JLabel jLabel1;
    // End of variables declaration//GEN-END:variables

}
```

D 9 fpfu.java

```
/*
 * fpfu.java
 *
 * DataBankTool V0.01 (5.4.2k2)
 *
 * (c) Reinhard S. Zoellner
 * erstellt im Rahmen der Diplomarbeit
 *
 * Created on 28. Februar 2002, 11:24
 */

package de.ducesoft.rsz.diparbeit;

/**
 * Dialogfenster das angezeigt wird wenn das Passwort oder
 * der Username oder beides fehlt.
 *
 * @author  Reinhard Zoellner
 */
public class fpfu extends javax.swing.JDialog {

    /** Creates new form fpfu */
    public fpfu(java.awt.Frame parent, boolean modal) {
        super(parent, modal);
        initComponents();
    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    private void initComponents() {//GEN-BEGIN:initComponents
        jLabel1 = new javax.swing.JLabel();
        jButton1 = new javax.swing.JButton();

        getContentPane().setLayout(new
                                org.netbeans.lib.awtextra.AbsoluteLayout());

        addWindowListener(new java.awt.event.WindowAdapter() {
            public void windowClosing(java.awt.event.WindowEvent evt) {
                closeDialog(evt);
            }
        });

        jLabel1.setText("Fehlender Username oder fehlendes Passwort");
        jLabel1.setFont(new java.awt.Font("Dialog", 1, 14));
        getContentPane().add(jLabel1, new
              org.netbeans.lib.awtextra.AbsoluteConstraints(30, 20, 350, 50));
```

```
        jButton1.setText("Zur\u00fcck");
        jButton1.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                zurueck(evt);
            }
        });

        getContentPane().add(jButton1, new
              org.netbeans.lib.awtextra.AbsoluteConstraints(160, 70, -1, -1));

        pack();
    }//GEN-END:initComponents

    private void zurueck(java.awt.event.MouseEvent evt) {
     //GEN-FIRST:event_zurueck
        setVisible(false);
        dispose();
    }//GEN-LAST:event_zurueck

    /** Closes the dialog */
    private void closeDialog(java.awt.event.WindowEvent evt) {
     //GEN-FIRST:event_closeDialog
        setVisible(false);
        dispose();
    }//GEN-LAST:event_closeDialog
    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JButton jButton1;
    private javax.swing.JLabel jLabel1;
    // End of variables declaration//GEN-END:variables

}
```

D 10 fpofu.java

```
/*
 * fpofu.java
 *
 * DataBankTool V0.01 (5.4.2k2)
 *
 * (c) Reinhard S. Zoellner
 * erstellt im Rahmen der Diplomarbeit
 *
 * Created on 27. Februar 2002, 11:56
 */

package de.ducesoft.rsz.diparbeit;

/**
 * Dialogfenster das angezeigt wird wenn ein falsches
 * Passwort oder ein falscher Username eingegeben wurde.
 *
 * @author  Reinhard Zoellner
 */
public class fpofu extends javax.swing.JDialog {

    /** Creates new form fpofu */
    public fpofu(java.awt.Frame parent, boolean modal) {
        super(parent, modal);
        initComponents();
    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    private void initComponents() {//GEN-BEGIN:initComponents
        jLabel1 = new javax.swing.JLabel();
        jButton1 = new javax.swing.JButton();

        getContentPane().setLayout(new
                              org.netbeans.lib.awtextra.AbsoluteLayout());

        addWindowListener(new java.awt.event.WindowAdapter() {
            public void windowClosing(java.awt.event.WindowEvent evt) {
                closeDialog(evt);
            }
        });

        jLabel1.setText("Falscher Username oder Falsches Passwort");
        jLabel1.setFont(new java.awt.Font("Dialog", 1, 14));
        getContentPane().add(jLabel1,new
              org.netbeans.lib.awtextra.AbsoluteConstraints(40, 0, 320, 100));
```

```
        jButton1.setText("Beenden");
        jButton1.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                ende(evt);
            }
        });

        getContentPane().add(jButton1,new
               org.netbeans.lib.awtextra.AbsoluteConstraints(160, 80, -1, -1));

        pack();
    }//GEN-END:initComponents

    private void ende(java.awt.event.MouseEvent evt) {
     //GEN-FIRST:event_ende
       System.exit(0);
    }//GEN-LAST:event_ende

    /** Closes the dialog */
    private void closeDialog(java.awt.event.WindowEvent evt) {
     //GEN-FIRST:event_closeDialog
        setVisible(false);
        dispose();
    }//GEN-LAST:event_closeDialog
    // Variables declaration - do not modify
    //GEN-BEGIN:variables
    private javax.swing.JButton jButton1;
    private javax.swing.JLabel jLabel1;
    // End of variables declaration//GEN-END:variables
}
```

D 11 lookData.java

```
/*
 * lookData.java
 *
 * DataBankTool V0.02 (13.4.2k2)
 *
 * (c) Reinhard S. Zoellner
 * erstellt im Rahmen der Diplomarbeit
 *
 * Created on 4. März 2002, 12:26
 */

package de.ducesoft.rsz.diparbeit;

import java.util.*;
import java.sql.*;
import java.io.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;

/**
 * Die Datensaetze einer Tabelle anzeigen.
 *
 * @author  Reinhard Zoellner
 */

public class lookData extends javax.swing.JFrame implements
                                        ListSelectionListener {

    /** Creates new form lookData */
    public lookData() {
        super("DataBankTool V0.02");
        initComponents();
    }

    private void initComponents() {
        jLabel1 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        jLabel3 = new javax.swing.JLabel();
        jButton1 = new javax.swing.JButton();
        jButton2 = new javax.swing.JButton();
        jList1 = new javax.swing.JList();
        jLabel4 = new javax.swing.JLabel();

        getContentPane().setLayout(new
                                org.netbeans.lib.awtextra.AbsoluteLayout());

        addWindowListener(new java.awt.event.WindowAdapter() {
```

```
    public void windowClosing(java.awt.event.WindowEvent evt) {
        exitForm(evt);
    }
});

jLabel1.setText("In der Tabelle : ");
getContentPane().add(jLabel1,new
        org.netbeans.lib.awtextra.AbsoluteConstraints(30, 30, -1, -1));

jLabel2.setText("sind volgende Eintr\u00e4ge enthalten.");
getContentPane().add(jLabel2,new
        org.netbeans.lib.awtextra.AbsoluteConstraints(30, 60, -1, -1));

jLabel3.setText("hier könnte eine Tabellenüberschrift stehen");
getContentPane().add(jLabel3,new
      org.netbeans.lib.awtextra.AbsoluteConstraints(30, 100, 380, -1));

jButton1.setText("zur\u00fcck");
jButton1.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseClicked(java.awt.event.MouseEvent evt) {
        zurueck(evt);
    }
});

getContentPane().add(jButton1,new
       org.netbeans.lib.awtextra.AbsoluteConstraints(70, 380, -1, -1));

jButton2.setText("drucken");
jButton2.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseClicked(java.awt.event.MouseEvent evt) {
        drucken(evt);
    }
});

getContentPane().add(jButton2,new
      org.netbeans.lib.awtextra.AbsoluteConstraints(280, 380, -1, -1));

Vector wordVector = new Vector();
Connection v_connection = null;
Statement v_statement= null;
try {
    // Oracle JDBC Treiber laden bzw. den DriverManager bekannt machen
    Class.forName("oracle.jdbc.driver.OracleDriver");
}
catch (java.lang.ClassNotFoundException e) {
       System.err.println("Class Not Found Exception : "
                          + e.getMessage());
}

try {
     v_connection = DriverManager.getConnection(
                             oraZugriffsDaten.getMyURL(),
```

```
                                oraZugriffsDaten.getMyUser(),
                                oraZugriffsDaten.getMyPasswd());
          v_statement = v_connection.createStatement();

         ResultSet v_resultset = v_statement.executeQuery(
                        "Select * from "+ oraZugriffsDaten.getMyTabelle());
         ResultSetMetaData rsmd = v_resultset.getMetaData();
         setNumberOfColumns(rsmd.getColumnCount());

         String[] elz = new String[getNumberOfColumns()+1];
         // einzufuegende Leerzeichen
          int [] laengsterString = new int[getNumberOfColumns()+1];
          for( int i = 1; i <= getNumberOfColumns(); i++) {
              laengsterString[i] = 0;
          }

          while (v_resultset.next()) {
              String[] s = new String[getNumberOfColumns()+1];
              for (int i = 1; i <= getNumberOfColumns(); i++) {
                 s[i]= v_resultset.getString(i);
              }

             for (int i= 1; i<= getNumberOfColumns(); i++) {
                 s1 = s1 + " " + s[i] ;
             }
             wordVector.add(s1);
          }
          v_statement.close();
          v_connection.close();
     }
     catch(SQLException e) {
     }

     jList1 = new javax.swing.JList(wordVector);
     JScrollPane scrollPane = new JScrollPane(jList1);
     JPanel p = new JPanel();
     p.add(scrollPane);
     jList1.addListSelectionListener(this);
     getContentPane().add(p,new
            org.netbeans.lib.awtextra.AbsoluteConstraints(30, 130, -1, -1));

     jLabel4.setText(oraZugriffsDaten.getMyTabelle());
     getContentPane().add(jLabel4,new
            org.netbeans.lib.awtextra.AbsoluteConstraints(120, 30, -1, -1));

     pack();
}
private void drucken(java.awt.event.MouseEvent evt) {
 //GEN-FIRST:event_drucken
     new druck().show();
}//GEN-LAST:event_drucken
```

```
private void zurueck(java.awt.event.MouseEvent evt) {
 //GEN-FIRST:event_zurueck
    setVisible(false);
    dispose();
}//GEN-LAST:event_zurueck

/** Exit the Application */
private void exitForm(java.awt.event.WindowEvent evt) {
 //GEN-FIRST:event_exitForm
    System.exit(0);
}//GEN-LAST:event_exitForm

/*
 * valueChanged-Methode aufgrund von ListSelectionListener
 * Interface implementieren.
 */
public void valueChanged(ListSelectionEvent evt) {
    JList source = (JList)evt.getSource();
    Object[] values = source.getSelectedValues();

    String text = "";
    for (int i = 0; i < values.length; i++) {
        String word = (String)values[i];
        text += word + " ";
    }
}

private void setNumberOfColumns(int columns) {
    numberOfColumns = columns;
}

private int getNumberOfColumns() {
    return numberOfColumns;
}

private void setTabellenueberschrift (String s) {
    tabellenueberschrift = s;
}

private String getTabellenueberschrift () {
    return tabellenueberschrift;
}
// Variables declaration - do not modify//GEN-BEGIN:variables
private javax.swing.JButton jButton2;
private javax.swing.JButton jButton1;
private javax.swing.JList jList1;
private javax.swing.JLabel jLabel4;
private javax.swing.JLabel jLabel3;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel1;
// End of variables declaration//GEN-END:variables
private int numberOfColumns = 0;
```

```
    private String tabellenueberschrift = null;
}
```

D 12 taa.java

```
/*
 * taa.java
 *
 * DataBankTool V0.02 (13.4.2k2)
 *
 * (c) Reinhard S. Zoellner
 * erstellt im Rahmen der Diplomarbeit
 *
 * Created on 1. März 2002, 19:41
 */

package de.ducesoft.rsz.diparbeit;

import java.util.*;
import java.sql.*;
import java.io.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;

/**
 * taa := Tabellenaufbau ansehen
 *
 * @author  Reinhard Zoellner
 */
public class taa extends javax.swing.JFrame implements ListSelectionListener {

    /** Creates new form taa */
    public taa() {
        super("DataBankTool V0.02");
        initComponents();
    }

   /*
    * tabelle := Tabellenbezeichnung der angezeigten Tabelle.
    */
    private void initComponents() {
        jLabel1 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        jLabel3 = new javax.swing.JLabel();
        jLabel4 = new javax.swing.JLabel();
        jList1 = new javax.swing.JList();
        jButton1 = new javax.swing.JButton();
        jButton2 = new javax.swing.JButton();
        getContentPane().setLayout(new
                                org.netbeans.lib.awtextra.AbsoluteLayout());
```

```
addWindowListener(new java.awt.event.WindowAdapter() {
    public void windowClosing(java.awt.event.WindowEvent evt) {
        exitForm(evt);
    }
});

jLabel1.setText("Tabellneaufbau von ");
jLabel1.setFont(new java.awt.Font("Dialog", 1, 14));
getContentPane().add(jLabel1,new
       org.netbeans.lib.awtextra.AbsoluteConstraints(40, 20, 150, 30));

jLabel2.setText(oraZugriffsDaten.getMyTabelle());
jLabel2.setFont(new java.awt.Font("Dialog", 1, 14));
getContentPane().add(jLabel2,new
      org.netbeans.lib.awtextra.AbsoluteConstraints(190, 20, 150, 30));

jLabel3.setText("ansehen.");
jLabel3.setFont(new java.awt.Font("Dialog", 1, 14));
getContentPane().add(jLabel3,new
        org.netbeans.lib.awtextra.AbsoluteConstraints(40, 50, 70, 30));

jLabel4.setText(" Zeil        Type    Casesensitiv   besreibbar");
jLabel4.setFont(new java.awt.Font("Lucida Sans Typewriter", 0, 12));
getContentPane().add(jLabel4,new
      org.netbeans.lib.awtextra.AbsoluteConstraints(50, 130, 400, 30));

jList1.setBorder(new
           javax.swing.border.LineBorder(new java.awt.Color(0, 0, 0)));
jList1.setMinimumSize(new java.awt.Dimension(200, 100));

Vector wordVector = new Vector();
Connection v_connection = null;
Statement v_statement= null;
try {
    // Oracle JDBC Treiber laden bzw. den DriverManager bekannt machen
    Class.forName("oracle.jdbc.driver.OracleDriver");
}
catch (java.lang.ClassNotFoundException e) {
    System.err.println("Class Not Found Exception : "
                       + e.getMessage());
}

try {
      v_connection = DriverManager.getConnection(
                              oraZugriffsDaten.getMyURL(),
                              oraZugriffsDaten.getMyUser(),
                              oraZugriffsDaten.getMyPasswd());
     v_statement = v_connection.createStatement();

     ResultSet v_resultset = v_statement.executeQuery(
                  "Select * from " + oraZugriffsDaten.getMyTabelle());
```

```
ResultSetMetaData rsmd = v_resultset.getMetaData();
int numberOfColumns = rsmd.getColumnCount();
int laengsterString = 0;
String colName = null;

// Leerzeichen zum formatieren der Tabelle
String leerzeichen = null;
String zusl = " ";
String tofl = " ";

for (int i = 1 ; i <= numberOfColumns; i++) {
    colName = rsmd.getColumnName(i);
    int j = colName.length();
    if ( j > laengsterString ) {
        laengsterString = j;
    }
}
for (int i = 1 ; i <= numberOfColumns; i++) {
    colName = rsmd.getColumnName(i);
    int j = colName.length();
    String dbtype = rsmd.getColumnTypeName(i);
    boolean caseSen = rsmd.isCaseSensitive(i);
    boolean writable = rsmd.isWritable(i);
    leerzeichen = " ";

// Formatierung der Tabelle implements Fenster mittels Leezeichen
// 1. verschiedene Column-Laengen auf gleiche laenge bringen
    for (int k = j; k < laengsterString; k++) {
       leerzeichen = leerzeichen + " ";
    }

    // 2. Bei verschiedenen Datentypen entstehen entsprechend mehr
    //    oder weniger Leerzeichen
    if( dbtype.equals("VARCHAR2")) {
        zusl = " ";
    }
    if( dbtype.equals("NUMBER")) {
        zusl = "   ";
    }
    if( dbtype.equals("DATE")) {
        zusl = "     ";
    }
    if( dbtype.equals("BLOB")) {
        zusl = "     ";
    }
    if( dbtype.equals("CLOB")) {
        zusl = "     ";
    }
    if( dbtype.equals("CHAR")) {
        zusl = "     ";
    }
    if( dbtype.equals("LONG")) {
```

```
                zusl = "     ";
            }

          // 3. wenn caseSensitiv true bzw. false ist muessen entsprechend
           //    die Leerzeichen eingesetzt werden.
            if( caseSen == true ) {
                tofl = "  ";
            }

            // Den String fuer die Ausgabe zusammensetzen.
            String vectoreintrag = colName + leerzeichen +
                                   dbtype + zusl +
                                   caseSen + tofl + writable;
            // Den String in den Vektor uernehmen.
            wordVector.add(vectoreintrag);

         }
        v_statement.close();
        v_connection.close();

    }
    catch(SQLException e) {
    }

    jList1 = new javax.swing.JList(wordVector);
    JScrollPane scrollPane = new JScrollPane(jList1);
    JPanel p = new JPanel();
    jList1.setFont(new java.awt.Font("Lucida Sans Typewriter", 0, 12));
    p.add(scrollPane);
    jList1.addListSelectionListener(this);
    getContentPane().add(p,new
          org.netbeans.lib.awtextra.AbsoluteConstraints(50, 150, -1, -1));

    jButton1.setText("Zur\u00fcck");
    jButton1.addMouseListener(new java.awt.event.MouseAdapter() {
        public void mouseClicked(java.awt.event.MouseEvent evt) {
            zurueck(evt);
        }
    });

    getContentPane().add(jButton1,new
            org.netbeans.lib.awtextra.AbsoluteConstraints(60, 90, -1, -1));

    jButton2.setText("Drucken");
    jButton2.addMouseListener(new java.awt.event.MouseAdapter() {
        public void mouseClicked(java.awt.event.MouseEvent evt) {
            drucken(evt);
        }
    });

    getContentPane().add(jButton2,new
           org.netbeans.lib.awtextra.AbsoluteConstraints(210, 90, -1, -1));
```

```
        pack();
    }

    /*
     * valueChanged-Methode aufgrund von ListSelectionListener
     * Interface implementieren.
     */
    public void valueChanged(ListSelectionEvent evt) {
        JList source = (JList)evt.getSource();
        Object[] values = source.getSelectedValues();

        String text = "";
        for (int i = 0; i < values.length; i++) {
            String word = (String)values[i];
            text += word + " ";
        }
    }

    private void drucken(java.awt.event.MouseEvent evt) {
     //GEN-FIRST:event_drucken
        new druck().show();
    }//GEN-LAST:event_drucken

    private void zurueck(java.awt.event.MouseEvent evt) {
     //GEN-FIRST:event_zurueck
        setVisible(false);
        dispose();
    }//GEN-LAST:event_zurueck

    /** Exit the Application */
    private void exitForm(java.awt.event.WindowEvent evt) {
     //GEN-FIRST:event_exitForm
        System.exit(0);
    }//GEN-LAST:event_exitForm

    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JButton jButton2;
    private javax.swing.JButton jButton1;
    private javax.swing.JList jList1;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel4;
}
```

E Pflichtenheft

1. Zielbestimmungen

1.1 Mußkriterien

Erstellung eines einfachen Datenmodells zur Modellierung von Wassereinzugsgebieten. Konkretisierung am BSP.: Chile.

Umsetzung und Realisierung des Modells und Integration der einzelnen Komponenten, Datenbank und GIS-Produkt und Kommunikation dieser, sowie erster Testeinsatz.

Beschreibung und Dokumentation des Vorgehens, Datenmodells, Implementierung, Testeinsatz und Architekturmodell.

1.2 Wunschkriterien

Umsetzung von Rollen, Administrator, zwei Unteradministratoren und User.

Zugriff auf Server soll über Browser erfolgen, so dass keine weitere Software auf dem Client installiert werden muß.

Oracle Datenbankinstallation auf Server im Institut für Tropentechnologie, zuerst Nutzung der Oracle-Datenbank des Fachbereiches Informatik in Gummersbach.

Schnittstellenumsetzung : Import und Export von Daten mittels OGC.

ArcSDE / ArcIMS sollten genutzt werden.

Wünschenswert wäre der Zugriff auf mehrere Datenbanken.

1.3 Abgrenzungskriterien

Es ist nicht beabsichtigt, ein ausführliches Datenmodell, das alle Aspekte eines Flusseinzugsgebietes widerspiegelt zu erstellen.

Replikationsprobleme werden nicht berücksichtigt.

Bei Nutzung einer eigenen Datenbank wird die Administration durch das Institut für Tropentechnologie übernommen.

2. Produkteinsatz

2.1 Anwendungsbereich

GIS im Wassermanagement.

2.2 Zielgruppe

Wasserexperten, Studenten, Assistenten und Professoren im Institut für Tropentechnologie.

2.3 Betriebsbedingungen

Das System zur Eingabe und Manipulation von Daten soll im 24-7-365 Tage Betrieb laufen. Über Zugriffszeiten wird keine Aussage gemacht.

3. Produktumgebung

3.1 Software

- ?? Win 2000
- ?? ArcInfo
- ?? ArcSDE
- ?? Oracle
- ?? TOAD
- ?? ERWIN
- ?? Ggf. SureSafe (Versionsmanagement)
- ?? NetBeans 3.3.1

3.2 Hardware

- ?? Server für ArcInfo
- ?? Server für Oracle und entsprechende Middleware
- ?? Client (Normaler PC)

3.3 Produktschnittstellen

- ?? Oracle ✍ ArcInfo
- ?? ArcIMS ✍ Browser
- ?? ArcSDE ✍ Oracle

4. Produktfunktionen

Problembeschreibung Institut für Tropentechnologie.

5. Produktdaten

Siehe ggf. späteres ausführliches Daten modell.

6. Produktleistungen

-

7. Benutzeroberfläche

Browser (mit entsprechenden Eingabemasken) / ESRI

8. Qualitätsbestimmungen

-

9. Globale Testfälle

In der ersten Umsetzung fiktive Daten.

10. Entwicklungsumgebung

?? NetBeans 3.3.1

11. **Ergänzungen**

Eigentlich geht es darum festzustellen, welche Möglichkeiten wie genutzt werden können, so dass im späteren Projekt dann entschieden werden kann, welche Umsetzung realisiert werden soll.

Notizen

Erklärung

Ich versichere, die von mir vorgelegte Arbeit selbständig verfasst zu haben. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder nicht veröffentlichten Arbeiten anderer entnommen sind, habe ich als entnommen kenntlich gemacht. Sämtliche Quellen und Hilfsmittel, die ich für die Arbeit benutzt habe, sind angegenben. Die Arbeit hat mit gleichem Inhalt bzw. in wesentlichen Teilen noch keiner anderen Prüfungsbehörde vgorgelegen.

Köln, den

Reinhard S. Zöllner