Geographic Information Technology Training Alliance (GITTA) presents:

Datenbanksysteme: Konzepte und Architekturen

Verantwortliche Personen: Michael Schrattner, Stephan Nebiker, Susanne Bleisch

Inhaltsverzeichnis

1. Datenbanksysteme: Konzepte und Architekturen

Nachdem Sie die Vorteile einer datenbankgestützten Datenverwaltung und die vielseitigen Einsatzmöglichkeiten von Datenbanksystemen kennen gelernt haben, wollen wir uns in dieser Lektion mit einigen grundlegenden Konzepten und typischen Architekturen von Datenbanksystemen vertraut machen.
Dabei interessiert zunächst, wie ein konzeptionelles Schema in eine Datenbankumgebung übertragen werden kann und welche typischen Mittel dafür zur Verfügung stehen. Anschliessend lernen wir eine generische Datenbankarchitektur kennen, die es uns erlauben soll, wichtige Aspekte wie das Zusammenspiel der verschiedenen Schemas sowie Schnittstellen für die Kommunikation mit einem Datenbankverwaltungssystem verstehen zu können.

Lernziele

- Sie kennen den Zusammenhang zwischen Datenschemata, Datenbankmodellen und Datenbankinstanzen und können diese beschreiben.
- Sie können die Drei-Schema-Architektur aufzeichnen und erläutern.
- Sie kennen Bedeutung und Prinzip einer Datenbankschnittstelle und können deren typische Funktionalität aufzählen.

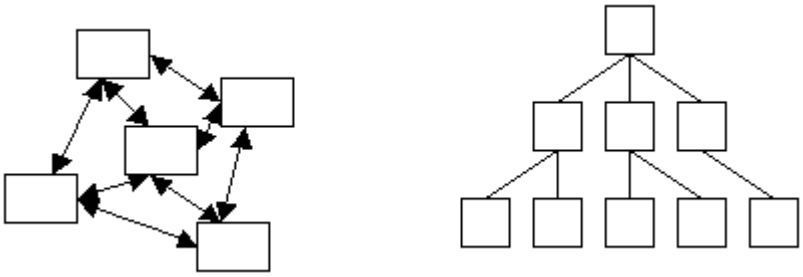
1.1. Datenbankmodelle, Schemas und Instanzen

Mit Datenbanken sollen Sachverhalte und Prozesse aus der Realwelt in computertechnischer Form beschrieben und gespeichert werden. Die dazu erforderliche Abbildung erfolgt wiederum mit Hilfe von Modellen, in diesem Fall den sogenannten Datenbankmodellen.
In dieser Unit werden die gebräuchlichsten vorgestellt. Im Anschluss daran werden Datenbankschemas als formelle Beschreibung einer konkreten Datenbank und Datenbankinstanzen als deren Inhalt bzw. Zustand eingeführt.

1.1.1. Datenbankmodelle

Datenbanksysteme können auf verschiedenen Datenmodellen bzw. Datenbankmodellen basieren. Ein Datenmodell ist eine Ansammlung von Konzepten und Regeln zur Beschreibung der Struktur einer Datenbank. Dabei verstehen wir unter der Struktur einer Datenbank die Datentypen, Bedingungen und Beziehungen zur Beschreibung bzw. Speicherung der Daten.
Die wichtigsten Datenbankmodelle sind:

Netzwerkmodell und Hierarchisches Modell
Sie sind Vorgänger des relationalen Modells. Sie bauen auf individuellen Datensätzen auf und können hierarchische Beziehungen oder auch allgemeinere netzartige Strukturen der Realwelt ausdrücken.



Netzwerk und Hierarchisches Datenmodell

Relationales Modell
Es ist das bekannteste und in heutigen DBMS am weitesten verbreitete Datenbankmodell. Es stellt die Datenbank als eine Sammlung von Tabellen (Relationen) dar, in denen alle Daten angeordnet werden.
Dieses Modul befasst sich vorwiegend mit dem relationalen Datenbankmodell und den darauf basierenden Datenbanksystemen.

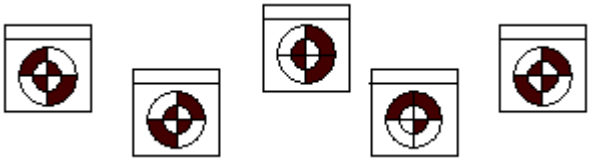
P-ID	Name	Vorname	Ort
1	Müller	Hans	Oberdorf
2	Meier	Jakob	Hinterwil
3	Keiser	Josef	Unterdorf
...	...	...	...

Ort-ID	Fläche	Art	...
5689	45869	Dorf	...
4758	23864	Weiler	...
2145	86541	Stadteil	...
...	...	...	...

Relationales Datenbankmodell

Objektorientiertes Modell

Objektorientierte Modelle definieren eine Datenbank als Sammlung von Objekten mit deren Eigenschaften und Methoden. Eine detailliertere Besprechung von objektorientierten Datenbanken folgt in späteren Modulen.



Schematische Darstellung eines objektorientierten Datenbankmodells

Objektrelationales Modell

Objektorientierte Modelle sind zwar sehr mächtig, aber auch recht komplex. Mit dem relativ neuen objektrelationalen Datenbankmodell wurde das einfache und weit verbreitete relationale Datenbankmodell um einige grundlegende objektorientierte Konzepte erweitert.

ID	XY	DF	ER
56		XXX	
45		YYY	
...	...	...	...

Schematische Darstellung des objektrelationalen Datenbankmodells

1.1.2. Datenbankschemas und Datenbankinstanzen

Unabhängig vom Datenbankmodell ist es wichtig, zwischen der Beschreibung der Datenbank und der Datenbank selber zu unterscheiden. Die Beschreibung einer Datenbank wird **Datenbankschema** oder auch *Metadaten* [1] genannt. Das Datenbankschema wird während des Datenbank-Design-Prozesses festgelegt und ändert später nur sehr selten.

Die eigentlichen Daten einer Datenbank verändern sich im Laufe der Zeit häufig. Der Datenbankzustand zu einem bestimmten Zeitpunkt, gegeben durch die aktuell existierenden Inhalte und Beziehungen und deren Attribute, wird **Datenbankinstanz** genannt.

Die nachfolgende Illustration zeigt auf, dass das Datenbankschema als Schablone oder Bauplan für eine oder mehrere Datenbankinstanzen betrachtet werden kann.

[1] Meta ist ein Präfix, das in den meisten informationstechnologischen Anwendungen „eine zugrunde liegende Definition oder Beschreibung“ meint. Metadaten sind also eine Definition oder Beschreibung von Daten. Man spricht auch von „Daten über Daten“.

Analogie

	Realwelt	Datenbank
Schema	 Plan für ein Norm-Haus	P-ID | Name | Vorname 'Vorlage' für eine Tabelle
Instanzen	 fertig gebaute Norm-Häuser	P-ID | Name | Vorname 102356 | Müller | Hans 102357 | Meier | Jakob Name | Vorname Keiser | Josef 523646 | Weber | Anita mit Daten 'gefüllte' Tabellen

Analogie zwischen Datenbankschema und Bauplan

Beim Entwurf einer Datenbank wird zwischen zwei Abstraktionsstufen und ihren entsprechenden Datenschemas (konzeptionelles Datenschema und logisches Datenschema, siehe untenstehende Definitionen) unterschieden.

Konzeptionelles Datenschema:

Ein konzeptionelles Datenschema („conceptual schema“) ist eine systemunabhängige Datenbeschreibung, d.h. sie ist unabhängig von den eingesetzten Datenbank- und Computersystemen. (ZEHNDER 1998)

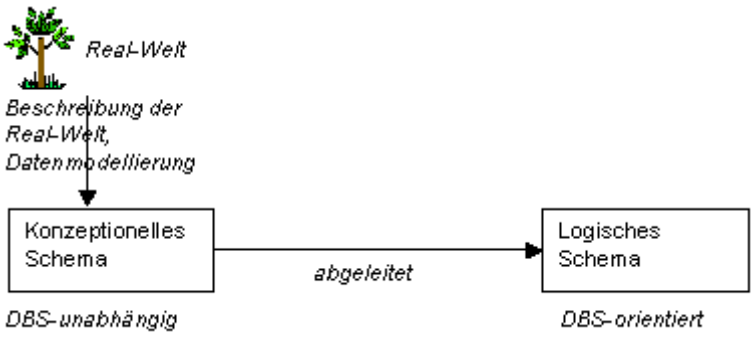
Logisches Datenschema:

Ein logisches Datenschema („logical schema“) beschreibt die Daten in der Datenbeschreibungssprache (DDL = Data Definition Language) eines bestimmten Datenbank-Verwaltungssystems. (ZEHNDER 1998)

Das konzeptionelle Datenschema orientiert sich ausschliesslich an der Datenbankanwendung und somit an der realen Welt, nicht aber an der datentechnischen Infrastruktur (DBMS und Computersysteme), die dafür allenfalls zum Einsatz kommen. *Entitätenblockdiagramme* [4] und Relationen sind Werkzeuge für die Erstellung eines konzeptionellen Schemas.

Beim Datenbankentwurf wird aus dem konzeptionellen Datenschema das logische Schema abgeleitet (siehe Unit **Relationales Datenbankdesign**). Am Ende dieser Ableitung steht das logische Datenschema für eine spezielle Anwendung und ein spezielles DBMS. Ein DB-Entwicklungssystem setzt danach das logische Schema direkt in Anweisungen für das DBMS um.

[4] Eine Entität ist eine Einheit. In einer relationalen Datenbank ist eine Entität als Tabelle dargestellt.



Schematische Darstellung der verschiedenen Schemas

1.1.3. Gegenüberstellung von räumlichen Modellen und Datenbankmodellen

Im Modul „Räumliche Datenmodellierung“ haben Sie Konzepte und numerische Modelle zur Repräsentation raumbezogener Phänomene der Realwelt kennen gelernt.
Die nachfolgende Gegenüberstellung soll Ihnen die Unterscheidung von räumlichen Modellen und Datenbankmodellen erleichtern.

Räumliche Modelle
Räumliche Modelle ermöglichen eine Abbildung (bzw. Modellierung) von raumbezogenen Phänomenen der Realität (Gegebenheiten, Prozesse) auf einer primär konzeptuellen Stufe, d. h. es wird in der Regel noch keine Aussage über die Konkretisierung bzw. die Umsetzung dieser Modelle in die Informatik gemacht. Räumliche Modelle sind oft grafik-orientierte Modelle (z. B. Karten oder Pläne) und können sowohl digital als auch analog (z. B. in einem Gipsmodell) umgesetzt werden.
Beispiele für räumliche Modelle sind:

- das Vektormodell als Spezialfall eines Objektmodells
- das Rastermodell als Spezialfall eines feldbasierten räumlichen Modells
- kombinierte (hybride) Modelle

Datenbankmodelle
Datenbankmodelle gehören in die Kategorie der Informatikmodelle und werden auch als konkrete Modelle bzw. als Implementierungsmodelle bezeichnet. Datenbankmodelle sind sehr allgemein einsetzbar und somit nicht auf raumbezogene Aufgabenstellungen beschränkt.
Beispiele für Datenbankmodelle sind:

- das relationale Datenbankmodell
- das objekt-relationale Datenbankmodell

1.2. DBMS-Architektur und Datenunabhängigkeit

Datenbankverwaltungssysteme sind komplexe Software-Lösungen, die oft über Jahre entwickelt und optimiert wurden. Aus Anwendungs- und Benutzersicht weisen jedoch die meisten Systeme eine ähnliche Basisarchitektur auf. Die Auseinandersetzung mit dieser Basisarchitektur hilft, Querbezüge zur Datenmodellierung zu schaffen und die eingangs des Moduls postulierte „Datenunabhängigkeit“ des Datenbankansatzes zu verstehen.

1.2.1. Drei-Schema-Architektur

Nachdem wir im Abschnitt „**Datenmodelle, Schemas und Instanzen**“ das konzeptionelle und das daraus abgeleitete logische Schema kennen gelernt haben, werden in dieser Unit zum Verständnis der DBMS-Architektur noch zwei weitere Schemas vorgestellt – das externe Schema und das interne Schema.

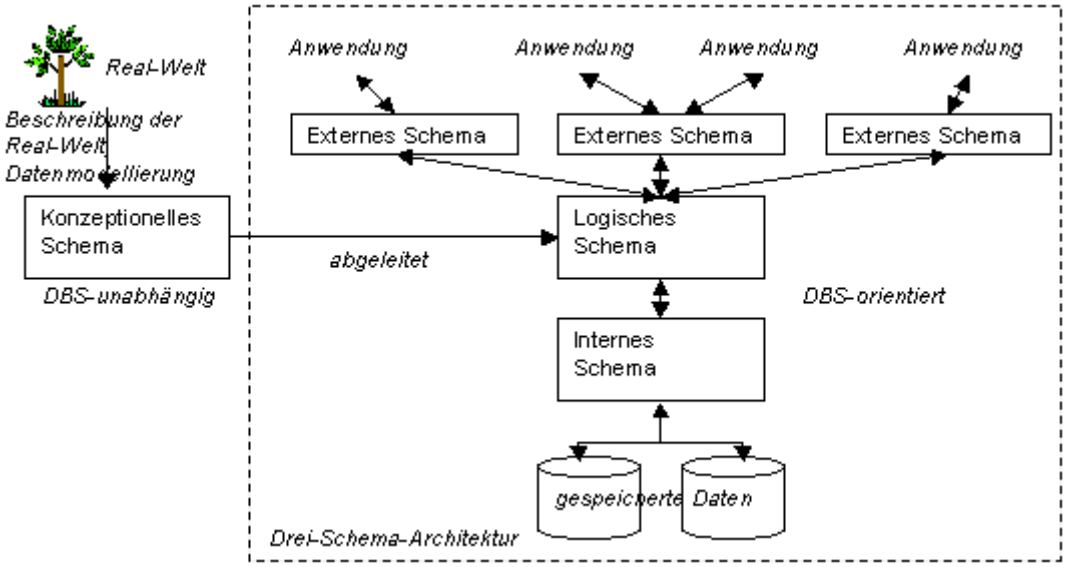
Externes Schema:

Ein externes Datenschema („external schema“) beschreibt die Benutzersichtdaten bestimmter Benutzer(gruppen) sowie die damit verbundenen spezifischen Operationen und Bedingungen. (ZEHNDER 1998)

Internes Schema:

Das interne Datenschema („internal schema“) beschreibt den Inhalt der Datenbasis und die benötigten Dienstleistungsfunktionen, soweit dies für den Einsatz des DBMS nötig ist. (ZEHNDER 1998)

Das interne Schema beschreibt somit die Daten aus computernaher Sicht bzw. aus der Sicht des Systems. Es ergänzt das zugehörige logische Schema um datentechnische Aspekte wie Speichermethoden oder Hilfskonstrukte zur Steigerung der Effizienz.



Drei-Schema-Architektur

Die rechte Seite der obigen Abbildung wird auch als Drei-Schema-Architektur bezeichnet. Bstehend aus internem, logischem und externem Schema.

Während das interne Schema die physische Gruppierung der Daten und die Speicherplatzbelegung beschreibt, gibt das logische Schema (abgeleitet aus dem konzeptionellen Schema) den grundlegenden Aufbau der Datenstruktur wieder. Das externe Schema einer konkreten Anwendung beleuchtet im Allgemeinen nur einen Teilbereich des logischen Schemas, der für die Anwendung relevant ist. Daraus folgt, dass jede Datenbank genau ein internes und ein logisches Schema hat, während jeweils mehrere externe Schemas möglich sind.
Das Ziel der Drei-Schema-Architektur ist die Trennung der Benutzeranwendungen von der physischen Datenbank, den gespeicherten Daten. Physisch sind die Daten lediglich auf der internen Ebene vorhanden, die anderen Repräsentationsformen werden jeweils bei Bedarf daraus berechnet bzw. abgeleitet. Das DBMS hat die Aufgabe, die Abbildung zwischen den einzelnen Ebenen zu realisieren.

1.2.2. Datenunabhängigkeit

Mit Kenntnis der Drei-Schema-Architektur lässt sich der Begriff der Datenunabhängigkeit so erläutern, dass eine höhere Ebene dieser Datenarchitektur immun gegenüber Änderungen auf der nächst tieferen Ebene ist.

Physikalische Unabhängigkeit:

Das logische Schema kann demnach unverändert bleiben, wenn sich beispielsweise aus Gründen der Optimierung oder Reorganisation der Speicherort oder die Speicherform einzelner Daten ändern.

Logische Unabhängigkeit:

Ebenso können bei (den meisten) Änderungen des logischen Schemas die externen Schemas unverändert weiterbestehen. Dies ist besonders deshalb erwünscht, weil dadurch Anwendungsprogramme nicht modifiziert oder neu übersetzt werden müssen.

Beispiel zur physikalischen und logischen Unabhängigkeit

Müssen beispielsweise in einem Geomatik-Ingenieurbüro mit unterschiedlichen Abteilungen (Amtliche Vermessung, Leitungskataster, usw.) aus irgendwelchen Gründen die Daten der verwendeten GIS-Datenbank auf einen neuen Datenbank-Server portiert werden, so darf dies keine Auswirkungen auf das logische Schema der Datenbank haben.
Wird wiederum das logische Schema verändert, müssen die unterschiedlichen Anwendergruppen des Geomatik-Ingenieurbüros ohne Einschränkungen mit den Daten der Datenbank weiterarbeiten können, obgleich sie verschiedene Anwender-Sichten auf die Daten haben.

1.3. Datenbanksprachen und Datenbankschnittstellen

Bis jetzt haben wir uns mit Datenmodellen bzw. Datenbankmodellen, Datenbeschreibungen und den Komponenten eines Datenbanksystems vertraut gemacht.
In dieser Unit wollen wir uns anschauen, wie ein Datenmodell in das Datenbanksystem und wie Informationen zu den Anwendern gelangen. Fachlich etwas präziser formuliert, sollen in dieser Unit die folgenden Fragen beantwortet werden:

- Wie erfolgt die Interaktion zwischen einer Anwendung und einem Datenbank-Verwaltungssystem?
- Wie sieht ein Datenbanksystem gegenüber einer Benutzerin aus?
- Wie können Anfragen durchgeführt und Resultate in einer Anwendung dargestellt werden?

1.3.1. Datenbanksprachen

DDL (Data Definition Language)
Wer Daten und Datenstrukturen beschreiben will, benötigt dazu ein geeignetes Beschreibungswerkzeug, eine **Datenbeschreibungssprache**. Diese dient der Definition und der Veränderung des Datenschemas.
Typische DDL-Operationen (mit den entsprechenden Schlüsselwörtern in der relationalen Datenbanksprache **SQL**) sind:

- Erzeugen von Tabellen und Festlegen von Attributen („create table …“)
- Ändern von Tabellen durch Hinzufügen oder Entfernen von Attributen („alter table …“)
- Löschen ganzer Tabellen mitsamt Inhalt (!) („drop table …“)

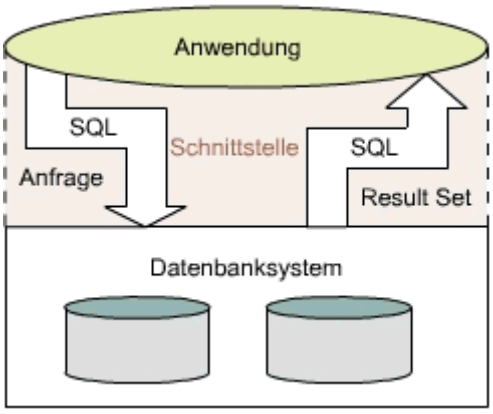
DML (Data Manipulation Language)
Zusätzlich wird eine Sprache für die Beschreibung der Arbeitsmöglichkeiten mit Daten (Speichern, Suchen, Lesen, Ändern), den sogenannten Datenmanipulationen, benötigt. Solche Operationen können mit einer **Datenmanipulationssprache** durchgeführt werden. In solchen Sprachen treten typischerweise die erwähnten Stichworte auf, meist auf Englisch („insert, modifiy, update, delete, select“).
Typische DML-Operationen (mit den entsprechenden Schlüsselwörtern in der relationalen Datenbanksprache **SQL**) sind:

- Einfügen von Daten („insert“)
- Verändern von Einträgen („update“)
- Löschen von Einträgen („delete“)
- Datenabfragen („select“)

Häufig aber sind diese zwei Sprachen für Definition und Manipulation von Datenbanken in einer umfassenden Sprache zusammengefasst. Ein gutes Beispiel ist die relationale Anfragesprache SQL (Structured Query Language), die vertiefter in der Lektion „**Relationale Anfragesprache SQL**“ behandelt wird.

1.3.2. Datenbankschnittstellen

Funktionsprinzip einer Datenbankschnittstelle



Funktionsprinzip einer Datenbankschnittstelle

Die Anwendung stellt mit Hilfe von SQL, einer Anfragesprache, eine Anfrage an das Datenbanksystem. Dort wird die entsprechende Antwort (Result-Set) bereitgestellt und ebenfalls mittels SQL an die Anwendung zurückgegeben. Diese Kommunikation kann interaktiv oder eingebettet stattfinden.

Art und Nutzung der Datenbankschnittstelle
Die zwei wichtigsten Einsatzmöglichkeiten einer Datenbankschnittstelle, wie SQL eine ist, sind nachfolgend kurz aufgeführt:

- interaktiv: SQL kann als interaktive Anfragesprache, direkt von einem Terminal aus eingesetzt werden.
- eingebettet SQL kann in Programmiersprachen („host language“) eingebettet („embedded“) werden, um damit Datenbankanwendungen zu erstellen.

1.3.3. Benutzungsoberflächen

Eine Benutzungsoberfläche ist die „Seite“ der Datenbankschnittstelle die der Benutzer sieht. Diese ist häufig grafisch oder zumindest teilgrafisch aufgebaut und bietet Hilfsmittel welche die Benutzung der Datenbank vereinfachen.

Formular-basierte Oberfläche
Diese Oberflächen bestehen Formularen die den Benutzern angepasst sind. Der Benutzer kann entweder alle Felder ausfüllen und so neue Einträge machen, oder er kann einzelne Felder ausfüllen und auf diese Weise eine Anfrage für die restlichen Felder machen.

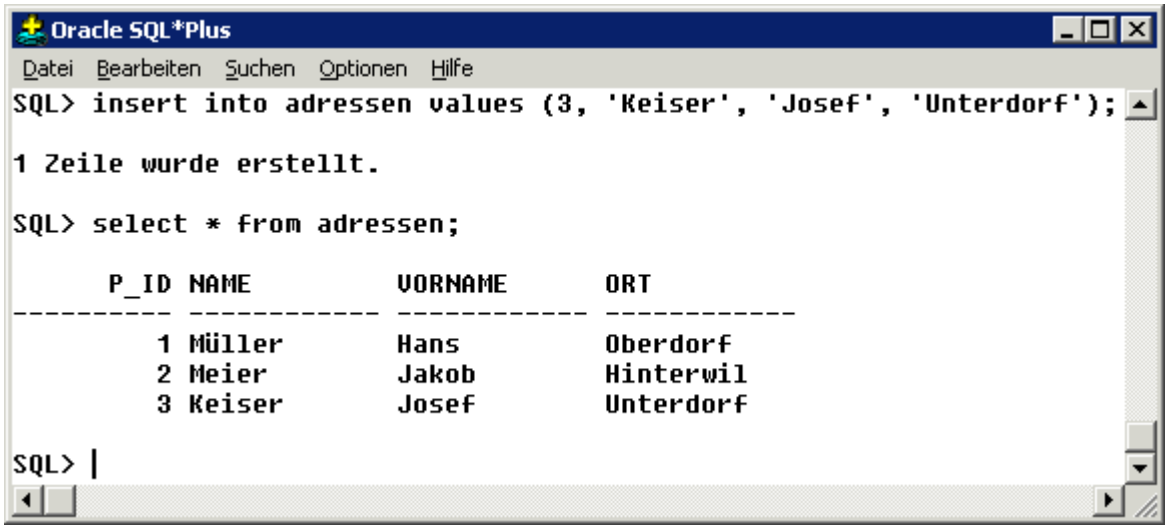
Formular-basierte Benutzungsoberflächen sind stark verbreitet und machen die wichtigste Interaktionsmöglichkeit mit DBMS aus. Formular-basierte Oberflächen sind sehr einfach zu bedienen und haben den grossen Vorteil, dass die Anwender keine Kenntnisse einer Datenbanksprache (z. B. SQL) benötigen.

Beispiel einer formular-basierten Benutzungsoberfläche

Text-basierte Oberflächen

Um die Datenbank administrieren zu können oder für andere professionelle Nutzer gibt es Möglichkeiten über Ein- und Ausgabefenster direkt in einer Anfragesprache (in Code-Form) mit der Datenbank zu kommunizieren. Wir werden diese Möglichkeit in der Lektion **Relationale Anfragesprache SQL** näher kennen lernen.

Text-basierte Oberflächen sind sehr mächtig und ermöglichen eine sehr umfassende Interaktion mit einem DBMS. Ihre Bedienung erfordert aber aktive Kenntnisse der entsprechenden Datenbanksprache (z. B. SQL).

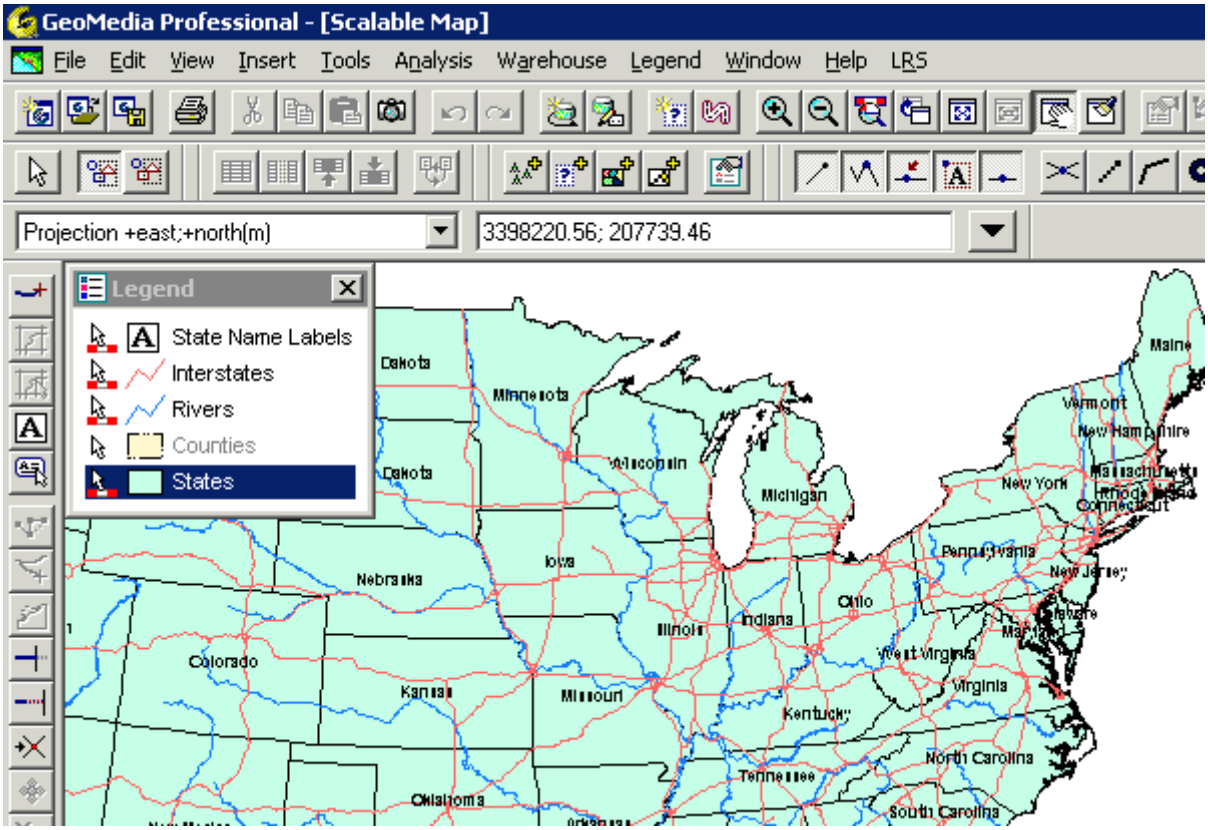


Beispiel einer text-basierten Benutzungsoberfläche

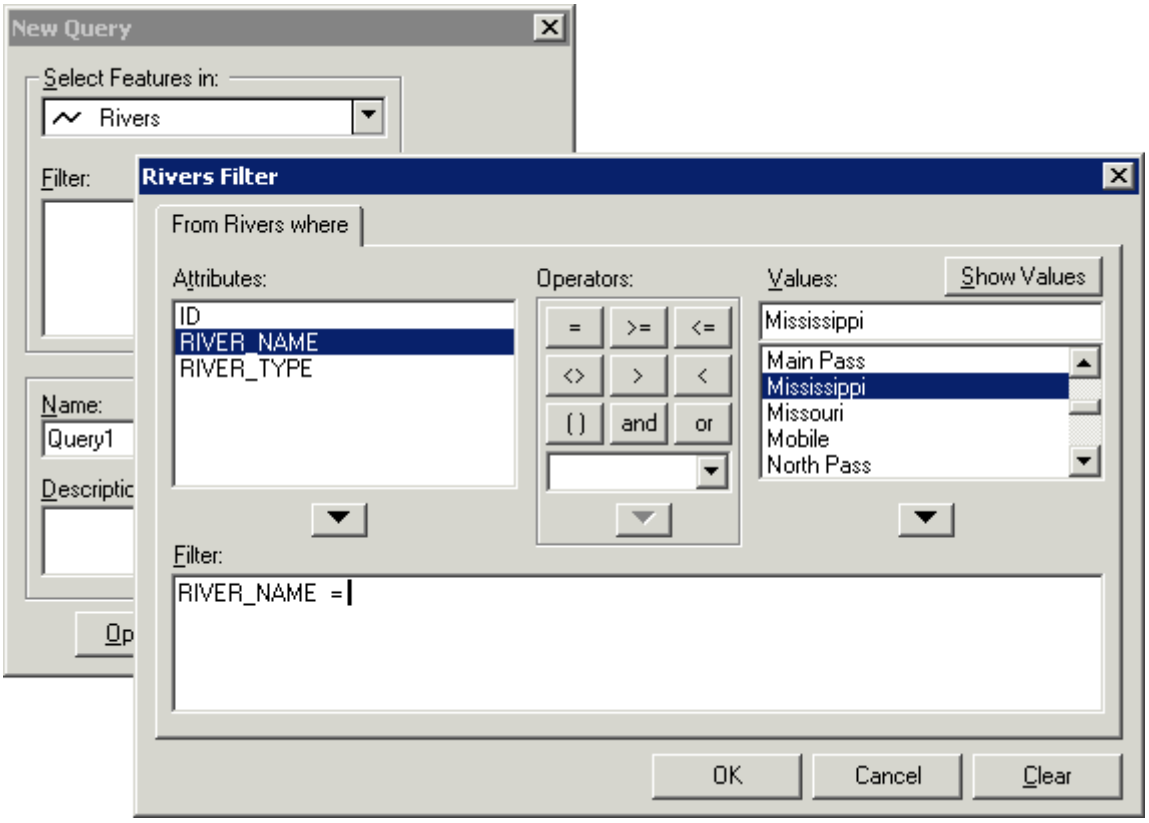
GIS-Oberfläche

Eine GIS-Benutzungsoberfläche integriert in aller Regel auch Elemente einer Datenbankschnittstelle. Die Datenbankinteraktion erfolgt dabei über eine Kombination von verschiedenen Oberflächen:

- grafische Interaktion via Selektionen innerhalb der Kartendarstellung
- Kombination von formular-basierter und text-basierter Interaktion (z.B. in der Form von Query-Wizards, speziellen Masken, die die Erstellung von Datenbankabfragen unterstützen und erleichtern)



Beispiel einer GIS-Oberfläche



Beispiel eines Query-Wizards in einem GIS

1.4. Aufgaben

Begriffszuordnung

Verschieben Sie die untenstehenden Begriffe so in die leeren Kästchen, dass jeweils zwei zusammenpassende Begriffe ein Paar bilden. Klicken Sie den Check-Button an, um ihre Anordnung zu überprüfen. Mit dem Reset-Button können Sie alle Begriffe wieder an die Ausgangsposition zurücksetzen.

Hinweis: Die Begriffe werden hier so verwendet, wie sie in dieser Lektion eingeführt wurden. In anderer Fachliteratur werden diese Begriffe möglicherweise leicht anders verwendet.

Dieses Element (Animation, Video etc.) kann nicht dargestellt werden und ist nur in der Onlineversion sichtbar. [link]

Drei-Schema-Architektur

Ziehen Sie die Nummern bei den Begriffen (blau) auf die richtigen weissen Kreise in der Grafik. Mit dem „Check“-Button können Sie ihre Anordnung überprüfen. Mit dem „Reset“-Button können Sie die Nummern an ihre Ausgangsposition zurücksetzen.

Achtung: Für ein korrektes Funktionieren des „Check“-Buttons sollten Sie diesen erst verwenden, wenn Sie alle Nummern den Kreisen in der Grafik zugeordnet haben.

Tipp: Beginnen Sie mit einfachen/bekannten Begriffen.

Dieses Element (Animation, Video etc.) kann nicht dargestellt werden und ist nur in der Onlineversion sichtbar. [link]

1.5. Übung zur Datenunabhängigkeit

Versuchen Sie in eigenen Worten die Architektur eines Datenbankmanagement-Systems zu beschreiben. Was ist Datenunabhängigkeit und wie wird sie erreicht? Ergänzen Sie Ihre Diskussion wenn möglich mit ein oder zwei Beispielen aus Ihrem bisherigen Informatikalltag, die den Nutzen der Datenunabhängigkeit aufzeigen.
Ihre Besprechung, die nicht mehr als 200 bis 400 Wörter umfassen soll, veröffentlichen Sie bitte auf dem Diskussionsforum unter dem Thema „Übung zur Datenunabhängigkeit“. Lesen Sie auch die Mitteilungen ihrer Mitstudierenden und kommentieren Sie diese gegebenenfalls.

1.6. Zusammenfassung

In dieser Lektion wurden die wichtigsten Datenbankmodelle (relational, objekt-orientiert und objekt-relational) und die darauf basierenden Datenbanktechnologien vorgestellt. Ein solches Datenbankmodell stellt die Grundregeln zur Verfügung, um ein konzeptionelles Schema in eine bestimmte Datenbankumgebung zu übertragen und es mittels eines logischen Schemas zu beschreiben. Diese Datenbeschreibung bildet die Grundlage, um anschliessend entsprechende Datenbanken (Instanzen) mit den eigentlichen Daten zu erzeugen und zu verwalten. Die logische Modellierung auf der Basis des relationalen Datenmodells lernen wir in der Lektion **Logische Modellierung** kennen.

In der Folge haben wir eine generische Datenbankarchitektur kennen gelernt, die auf einem 3-Schema-Konzept basiert. Diese Drei-Schema-Architektur erweitert das logische Schema um ein internes und ein externes Schema. Die Dreiteilung wiederum bildet eine wichtige Voraussetzung für die bereits früher postulierte Datenunabhängigkeit und für die Realisierung von Schnittstellen für die Kommunikation zwischen Anwendung und Datenbankverwaltungssystem. Ein Grossteil der Systeme bietet eine Schnittstelle auf der Basis der relationalen Datenbanksprache SQL (Structured Query Language). Die Grundelemente und den Einsatz von SQL lernen Sie in der Lektion **„Die relationale Anfragesprache SQL“** kennen.

1.7. Literaturempfehlungen

- **ZEHNDER, C. A.**, 1998. *Informationssysteme und Datenbanken*. Zürich: vdf Hochschulverlag AG.
  Einführung ins Thema Informationssysteme und Datenbanken inklusive Datenmodellierung, auf Deutsch

1.8. Glossar

Entität:

Eine Entität ist eine Einheit. In einer relationalen Datenbank ist eine Entität als Tabelle dargestellt.

Externes Schema:

Ein externes Datenschema („external schema“) beschreibt die Benutzersichtdaten bestimmter Benutzer(gruppen) sowie die damit verbundenen spezifischen Operationen und Bedingungen. (ZEHNDER 1998)

Internes Schema:

Das interne Datenschema („internal schema“) beschreibt den Inhalt der Datenbasis und die benötigten Dienstleistungsfunktionen, soweit dies für den Einsatz des DBMS nötig ist. (ZEHNDER 1998)

Das interne Schema beschreibt somit die Daten aus computernaher Sicht bzw. aus der Sicht des Systems. Es ergänzt das zugehörige logische Schema um datentechnische Aspekte wie Speichermethoden oder Hilfskonstrukte zur Steigerung der Effizienz.

Konzeptionelles Datenschema:

Ein konzeptionelles Datenschema („conceptual schema“) ist eine systemunabhängige Datenbeschreibung, d.h. sie ist unabhängig von den eingesetzten Datenbank- und Computersystemen. (ZEHNDER 1998)

Logisches Datenschema:

Ein logisches Datenschema („logical schema“) beschreibt die Daten in der Datenbeschreibungssprache (DDL = Data Definition Language) eines bestimmten Datenbank-Verwaltungssystems. (ZEHNDER 1998)

Logische Unabhängigkeit:

Ebenso können bei (den meisten) Änderungen des logischen Schemas die externen Schemas unverändert weiterbestehen. Dies ist besonders deshalb erwünscht, weil dadurch Anwendungsprogramme nicht modifiziert oder neu übersetzt werden müssen.

Metadaten:

Meta ist ein Präfix, das in den meisten informationstechnologischen Anwendungen „eine zugrunde liegende Definition oder Beschreibung“ meint. Metadaten sind also eine Definition oder Beschreibung von Daten. Man spricht auch von „Daten über Daten“.

Physikalische Unabhängigkeit:

Das logische Schema kann demnach unverändert bleiben, wenn sich beispielsweise aus Gründen der Optimierung oder Reorganisation der Speicherort oder die Speicherform einzelner Daten ändern.

1.9. Bibliographie

- **BARTELME, N.**, 2000. *Geoinformatik - Modelle, Strukturen, Funktionen*. 3rd. Berlin: Springer.
- **DITTRICH, K.**, 1999. Datenbanksysteme. *In:* **RECHENBERG, P., POMBERGER, G.**, ed. *Informatik-Handbuch*. Wien: Carl Hanser Verlag, 875-908.
- **ELMASRI, R.; NAVATHE, S.B.**, 1994. *Fundamentals of Database Systems*. 2nd. Redwood City, California: Addison-Wesley.
- **NEBIKER, S.**, 2002. *Modul DBMS*. Muttenz: FHBB, VGI.
- **ZEHNDER, C. A.**, 1998. *Informationssysteme und Datenbanken*. Zürich: vdf Hochschulverlag AG.