

Geographic Information Technology Training Alliance (GITTA) presents:

Database Systems: Concepts and Architectures

Responsible persons: Stephan Nebiker, Susanne Bleisch

Table Of Content

- 1. Database Systems: Concepts and Architectures 2
 - 1.1. Database Models, Schemes and Instances 3
 - 1.1.1. Database Models 3
 - 1.1.2. Database Schemes and Database Instances 4
 - 1.1.3. Comparison of Spatial Models and Database Models 6
 - 1.2. DBMS-Architecture and Data Independence 7
 - 1.2.1. Three-Schemes Architecture 7
 - 1.2.2. Data Independence 8
 - 1.3. Database Languages and Database Interfaces 9
 - 1.3.1. Database Languages 9
 - 1.3.2. Database Interfaces 10
 - 1.3.3. User Interfaces 10
 - 1.4. Tasks 15
 - 1.5. Exercise Data Independence 16
 - 1.6. Summary 17
 - 1.7. Recommended Reading 18
 - 1.8. Glossary 19
 - 1.9. Bibliography 20

1. Database Systems: Concepts and Architectures

Knowing about the advantages of database supported data management and the huge possibilities of applying database systems this lesson will explain the basic concepts and typical architectures of database systems.

Firstly, it is discussed how a *conceptual*¹ scheme can be transferred into a database environment and which typical database models can be used for this task. Afterwards, the generic database architecture is explained, which should allow for the understanding the important aspects of the co-operation between the different schemes and the interfaces for the communication with a database management system.

Learning Objectives

- You know the relationship between data schemes, database models and database instances and are able to describe them.
- You are able to sketch and explain the 3-Scheme-Architecture.
- You know the meaning and principles of a database interface and are able to list their typical features.

¹ A concept is an abstract, universal idea, notion, or entity that serves to designate a category or class of entities, events, or relations. Concepts are abstract in that they omit the differences of the things in their extension, treating them as if they were identical. They are universal in that they apply equally to every thing in their extension. Concepts are also the basic elements of propositions, much the same way a word is the basic semantic element of a sentence.

1.1. Database Models, Schemes and Instances

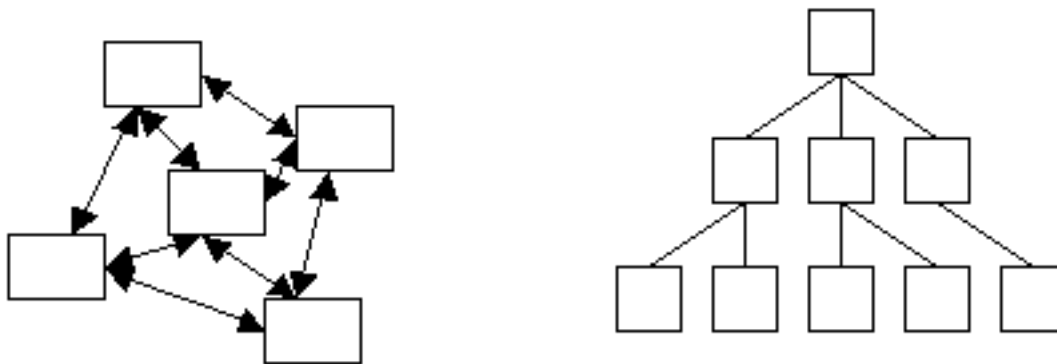
With the help of databases facts and processes from the real world should be described and stored in digital form. The abstraction from the real world to the digital format is done with the help of models, so called database models.

In this unit, the most often used database models are presented. Following, database schemes are introduced as a formal description of a concrete database and their database instances or their database state.

1.1.1. Database Models

Database systems can be based on different data models or database models respectively. A data model is a collection of concepts and rules for the description of the structure of the database. Structure of the database means the data types, the constraints and the relationships for the description or storage of data respectively. The most often used data models are:

Network Model and Hierarchical Model The network model and the hierarchical model are the predecessors of the relational model. They build upon individual data sets and are able to express hierarchical or network like structures of the real world.



Network Model and Hierarchical Model

Relational Model The relational model is the best known and in today's DBMS most often implemented database model. It defines a database as a collection of tables (relations) which contain all data.

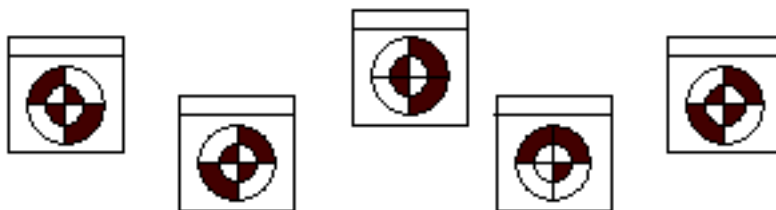
This module deals predominantly with the relational database model and the database systems based on it.

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

City-ID	Area	Type	...
5689	45869	Town	...
4758	23864	Village	...
2145	86541	City	...
...





Relational Database Model

Object-oriented Model Object-oriented models define a database as a collection of objects with features and methods. A detailed discussion of object-oriented databases follows in an advanced module.



Schematic Representation of a Object-oriented Database Model

Object-relational Model Object-oriented models are very powerful but also quite complex. With the relatively new object-relational database model is the wide spread and simple relational database model extended by some basic object-oriented concepts. These allow us to work with the widely know relational database model but also have some advantages of the object-oriented model without its complexity.

ID	XY	DF	ER
56		XXX	
45		YYY	
...

Schematic Representation of the object-relational Database Model



1.1.2. Database Schemes and Database Instances

Independent from the database model it is important to differentiate between the description of the database and the database itself. The description of the database is called **database scheme** or also *metadata* ². The database scheme is defined during the database design process and changes very rarely afterwards.

The actual content of the database, the data, changes often over the years. A database state at a specific time defined through the currently existing content and relationship and their attributes is called a **database instance**. The following illustration shows that a database scheme could be looked at like a template or building plan for one or several database instances.

² Metadata is literally "data about data", is information that describes another set of data. A common example is a library catalog card, which contains data about the contents and location of a book: It is data about the data in the book referred to by the card. Other common contents of metadata include the source or author of the described dataset, how it should be accessed, and its limitations. Another important type of data about data is the links or relationship among data.

Analogy

	Real World	Database												
Scheme	 Plan for a Standard-House	<table border="1"> <thead> <tr> <th>P-ID</th> <th>Name</th> <th>Prenome</th> </tr> </thead> <tbody> <tr> <td> </td> <td> </td> <td> </td> </tr> <tr> <td> </td> <td> </td> <td> </td> </tr> </tbody> </table> Template for a Table	P-ID	Name	Prenome									
P-ID	Name	Prenome												
Instances	 Built Standard-Houses	<table border="1"> <thead> <tr> <th>P-ID</th> <th>Name</th> <th>Prenome</th> </tr> </thead> <tbody> <tr> <td>102356</td> <td>Smith</td> <td>John</td> </tr> <tr> <td>102357</td> <td>Potter</td> <td>Harry</td> </tr> </tbody> </table> <table border="1"> <tbody> <tr> <td>523646</td> <td>Wood</td> <td>Lucinda</td> </tr> </tbody> </table> Data-filled Tables	P-ID	Name	Prenome	102356	Smith	John	102357	Potter	Harry	523646	Wood	Lucinda
P-ID	Name	Prenome												
102356	Smith	John												
102357	Potter	Harry												
523646	Wood	Lucinda												

Analogy Database Schemes and Building Plans

When designing a database it is differentiated between two levels of abstraction and their respective data schemes, the conceptual and the logical data scheme.

Conceptual Data Scheme :

A conceptual data scheme is a system independent data description. That means that it is independent from the database or computer systems used. (Translated) (ZEHNDER 1998)

Logical Data Scheme:

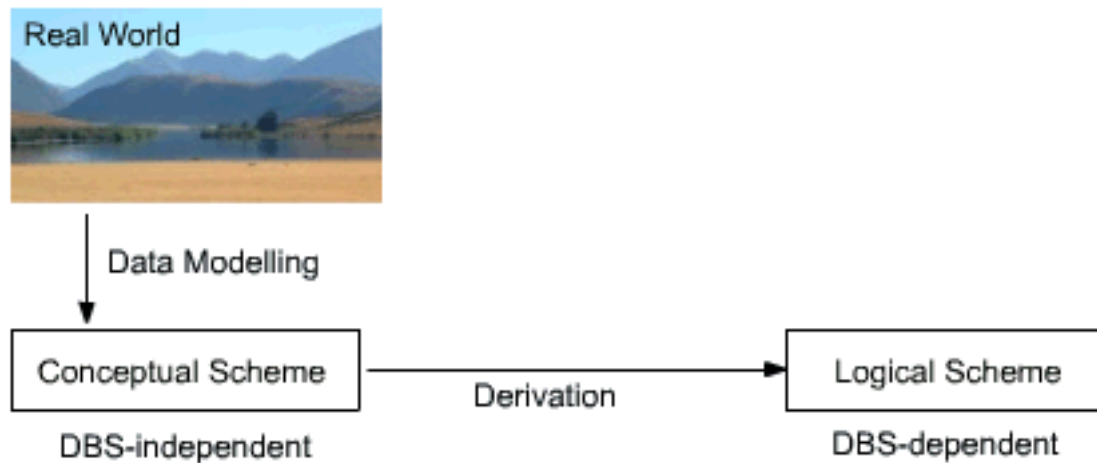
A logical data scheme describes the data in a data definition language DDL of a specific database management system. (Translated) (ZEHNDER 1998)

A logical data scheme describes the data in a data definition language DDL of a specific database management system. (Translated) (ZEHNDER 1998)

The conceptual data scheme orients itself exclusively by the database application and therefore by the real world. It does not consider any data technical infrastructure like DBMS or computer systems, which are eventually employed. *Entity relationship diagrams*⁵ and relations are tools for the development of a conceptual scheme.

When designing a database the conceptual data scheme is derived from the logical data scheme (see unit **Relational Database Design**). This derivation results in a logical data scheme for one specific application and one specific DBMS. A DB-Development System converts then the logical scheme directly into instructions for the DBMS.

⁵ An entity is something that has a distinct, separate existence, though it need not be a material existence. In a relational database an entity is represented as a relation.



Schematic Representation of the Different Schemes

1.1.3. Comparison of Spatial Models and Database Models

Knowing about data modelling, the concepts and numerical models for the representation of spatial phenomena should be familiar.

The following comparison should make the differentiation between spatial models and database models a bit easier.

Spatial Models

Spatial models allow the modelling or representation respectively spatial phenomena of the real world like facts and processes. This happens firstly on a conceptual level, which means that, in general, the actual implementation of these models is not considered. Spatial models are often graphic oriented models like maps or plans and can be represented digitally or analogously (e.g. plaster model).

Examples for spatial models:

- The vector model is a special case of the object model.
- The raster model is a special case of the tesseral spatial model.

Database Models

Database models are from the category of informatic models and are therefore exact models or implementation models respectively. Database models can be used quite often and are not restricted to spatial types of problems.

Examples for database models:

- The relational database model.
- The object-relational database model.

In the intermediate module data management, the representation of spatial data models in database models and the representation of spatial data in geodatabase system will be discussed.

1.2. DBMS-Architecture and Data Independence

Database management systems are complex softwares which were often developed and optimised over years. From the view of the user, however, most of them have a quite similar basic architecture. The discussion of this basic architecture shall help to understand the connection with data modelling and the introductionally to this module postulated 'data independence' of the database approach.

1.2.1. Three-Schemes Architecture

Knowing about the conceptual and the derived logical scheme (discussed in unit **Database Models, Schemes and Instances** this unit explains two additional schemes - the external scheme and the internal scheme - which help to understand the DBMS architecture.

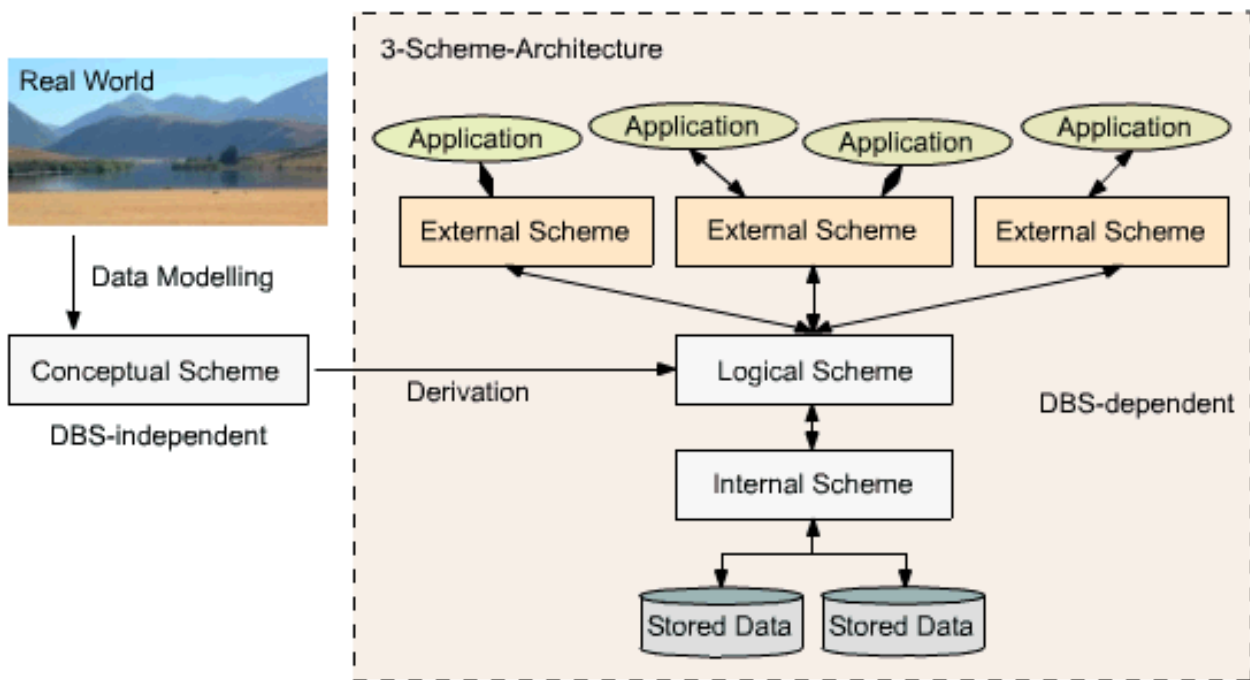
External Scheme:

An external data scheme describes the information about the user view of specific users (single users and user groups) and the specific methods and constraints connected with this information. (Translated) (ZEHNDER 1998)

Internal Scheme:

The internal data scheme describes the content of the data and the required service functionality which is used for the operation of the DBMS. (Translated) (ZEHNDER 1998)

Therefore, the internal scheme describes the data from a view very close to the computer or system in general. It completes the logical scheme with data technical aspects like storage methods or help functions for more efficiency.



Three-Schemes Architecture

The right hand side of the representation above is also called the three-schemes architecture: internal, logical and external scheme.

While the internal scheme describes the physical grouping of the data and the use of the storage space, the logical scheme (derived from the conceptual scheme) describes the basic construction of the data structure. The external scheme of a specific application, generally, only highlights that part of the logical scheme which is relevant for its application. Therefore, a database has exactly one internal and one logical scheme but may have several external schemes for several applications using this database.

The aim of the three-schemes architecture is the separation of the user applications from the physical database, the stored data. Physically the data is only existent on the internal level while other forms of representation are calculated or derived respectively if needed. The DBMS has the task to realise this representation between each of these levels.

1.2.2. Data Independence

With knowledge about the three-schemes architecture the term data independence can be explained as followed: Each higher level of the data architecture is immune to changes of the next lower level of the architecture.

Physical Independence:

Therefore, the logical scheme may stay unchanged even though the storage space or type of some data is changed for reasons of optimisation or reorganisation.

Logical Independence:

Also the external scheme may stay unchanged for most changes of the logical scheme. This is especially desirable as in this case the application software does not need to be modified or newly translated.

1.3. Database Languages and Database Interfaces

So far, we have got to know about data models, data descriptions and the components of a database system. In this unit, it is explained how 'a data models gets into a database system' and 'how the information gets to the users'. More correctly formulated the following questions will be answered:

- How does an application interact with a database management system?
- How does a user look at a database system?
- How can a user query a database system and view the results in his/her application?

1.3.1. Database Languages

DDL For describing data and data structures a suitable description tool, a **data definition language** (DDL), is needed. With this help a data scheme can be defined and also changed later.

Typical DDL operations (with their respective keywords in the structured query language **SQL**):

- Creation of tables and definition of attributes (CREATE TABLE ...)
- Change of tables by adding or deleting attributes (ALTER TABLE ...)
- Deletion of whole table including content (!) (DROP TABLE ...)

DML Additionally a language for the descriptions of the operations with data like store, search, read, change, etc. the so-called data manipulation, is needed. Such operations can be done with a **data manipulation language** (DML). Within such languages keywords like insert, modify, update, delete, select, etc. are common.

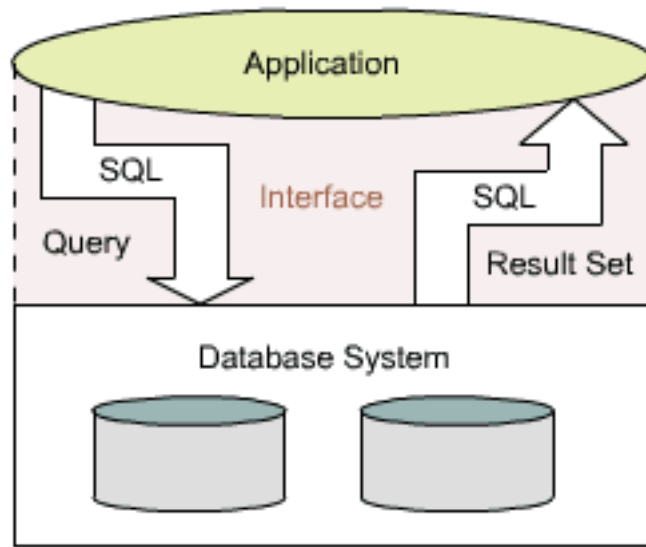
Typical DML operations (with their respective keywords in the structured query language **SQL**):

- Add data (INSERT)
- Change data (UPDATE)
- Delete data (DELETE)
- Query data (SELECT)

Often these two languages for the definition and manipulation of databases are combined in one comprehensive language. A good example is the structured query language SQL which is discussed in detail in lesson [Structured Query Language SQL](#).

1.3.2. Database Interfaces

Working Principle of a Database Interface



Working Principle of a Database Interface

The application poses with the help of SQL, a query language, a query to the database system. There, the corresponding answer (result set) is prepared and also with the help of SQL given back to the application. This communication can take place interactively or be embedded into another language.

Type and Use of the Database Interface

Following, two important uses of a database interface like SQL are listed:

- Interactive SQL can be used interactively from a terminal.
- Embedded SQL can be embedded into another language (host language) which might be used to create a database application.

1.3.3. User Interfaces

A user interface is the view of a database interface that is seen by the user. User interfaces are often graphical or at least partly graphical (GUI - graphical user interface) constructed and offer tools which make the interaction with the database easier.

- Form-based Interfaces This interface consist of forms which are adapted to the user. He/She can fill in all of the fields and make new entries to the database or only some of the fields to query the other ones. But some operations might be restricted by the application.
Form-based user interfaces are wide spread and are a very important means of interacting with a DBMS. They are easy to use and have the advantage that the user does not need special knowledge about database languages like SQL.

The image shows the SBB Travel Online interface. At the top, there is a red and blue header with the SBB logo and the text "SBB Travel Online". Below the header, on the left, is a sidebar with the text "Swiss Federal Railways" and a "Search" section with a text input field. Below the search field are several blue buttons: "Door-to-Door Timetable", "Ticket Shop", "Info on the offers", "Dialogue", "Travelling in Europe", "Guests from abroad", and "Shopping Cart". The main content area is titled "Timetable, Fare and Ticket order" and contains a form with the following fields and options:

- Departure**: A text input field with a "Search" button.
- Destination**: A text input field with a "Search" button.
- Via**: Three text input fields, each with a "Search" button.
- Travel date**: A date input field with the format (DD.MM.YYYY) and two radio buttons: "Today" (selected) and "Tomorrow".
- Time**: A time input field with the format (HH:MM) and two radio buttons: "Departure" (selected) and "Arrival".

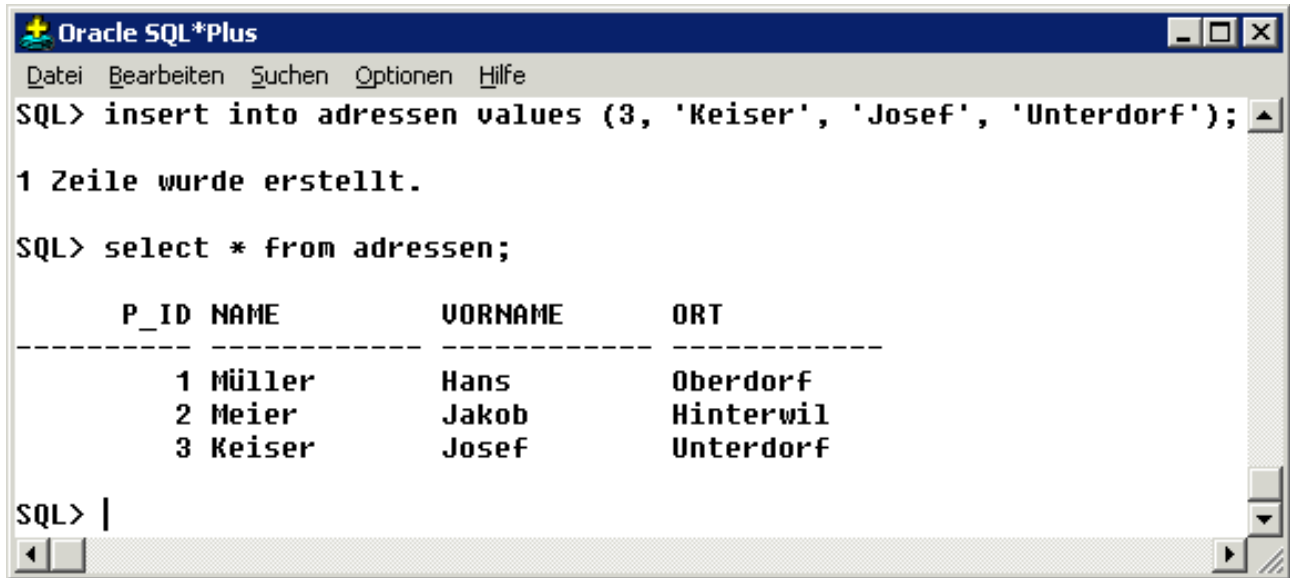
Below the form, there is a checkbox for "With all available carrier" and an "Options" button. At the bottom of the form, there are radio buttons for "Normal" (selected) and "Light", and three buttons: "Start query", "Return trip", and "Reset".

Example of a Form-based User Interface

Text-based Interfaces

To be able to administrate the database or for other professional users there are possibilities to communicate with the DBMS directly in the query language (in code form) via a input/output window.

We will see this possibility later in the lesson **Structured Query Language SQL**. Text-based interfaces are very powerful tools and allow a comprehensive interaction with a DBMS. However, the use of these is based on active knowledge of the respective database language.



The screenshot shows the Oracle SQL*Plus command-line interface. The window title is "Oracle SQL*Plus". The menu bar includes "Datei", "Bearbeiten", "Suchen", "Optionen", and "Hilfe". The command prompt shows the following sequence of commands and output:

```
SQL> insert into adressen values (3, 'Keiser', 'Josef', 'Unterdorf');  
  
1 Zeile wurde erstellt.  
  
SQL> select * from adressen;
```

P_ID	NAME	VORNAME	ORT
1	Müller	Hans	Oberdorf
2	Meier	Jakob	Hinterwil
3	Keiser	Josef	Unterdorf

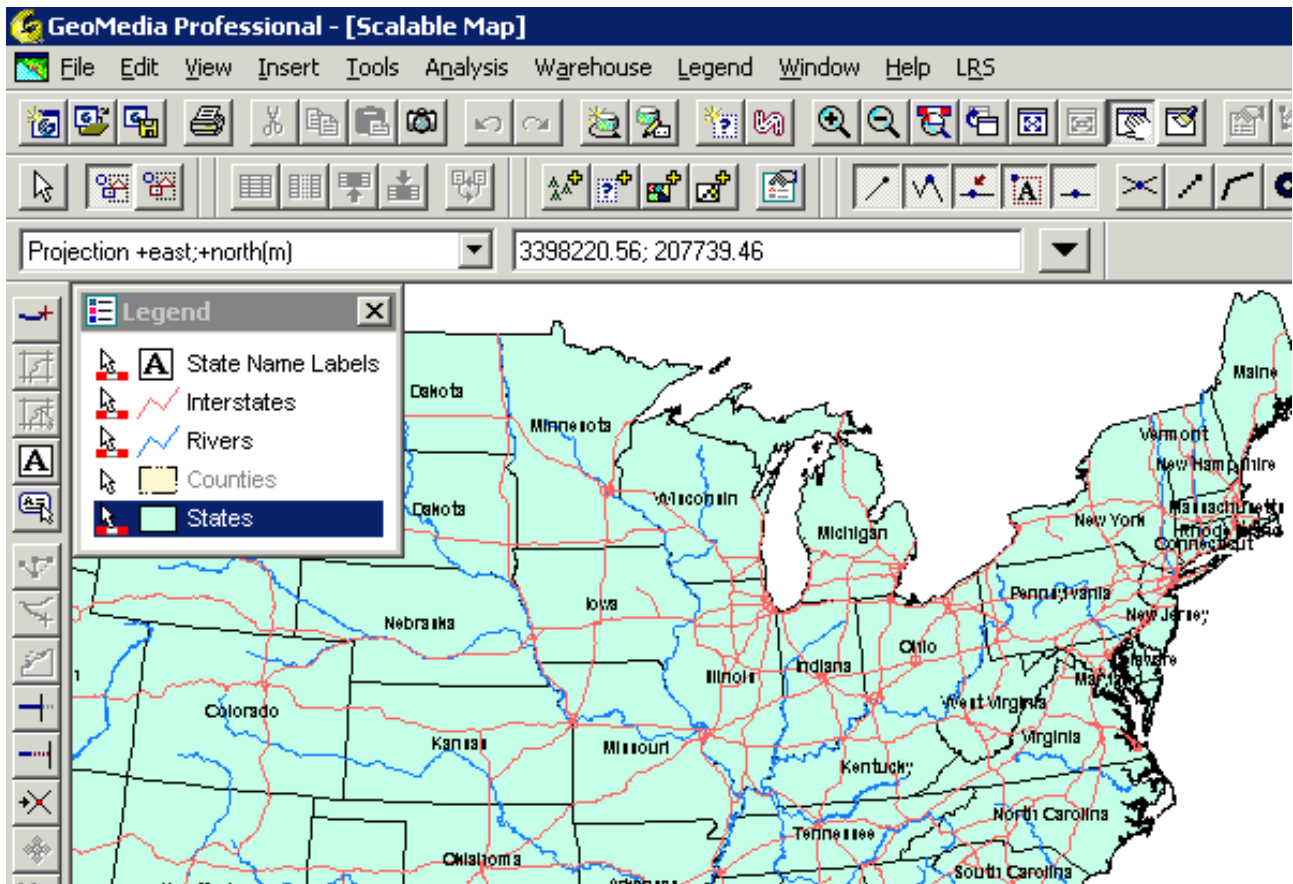
The command prompt ends with "SQL> |".

Example of a Text-base User Interface

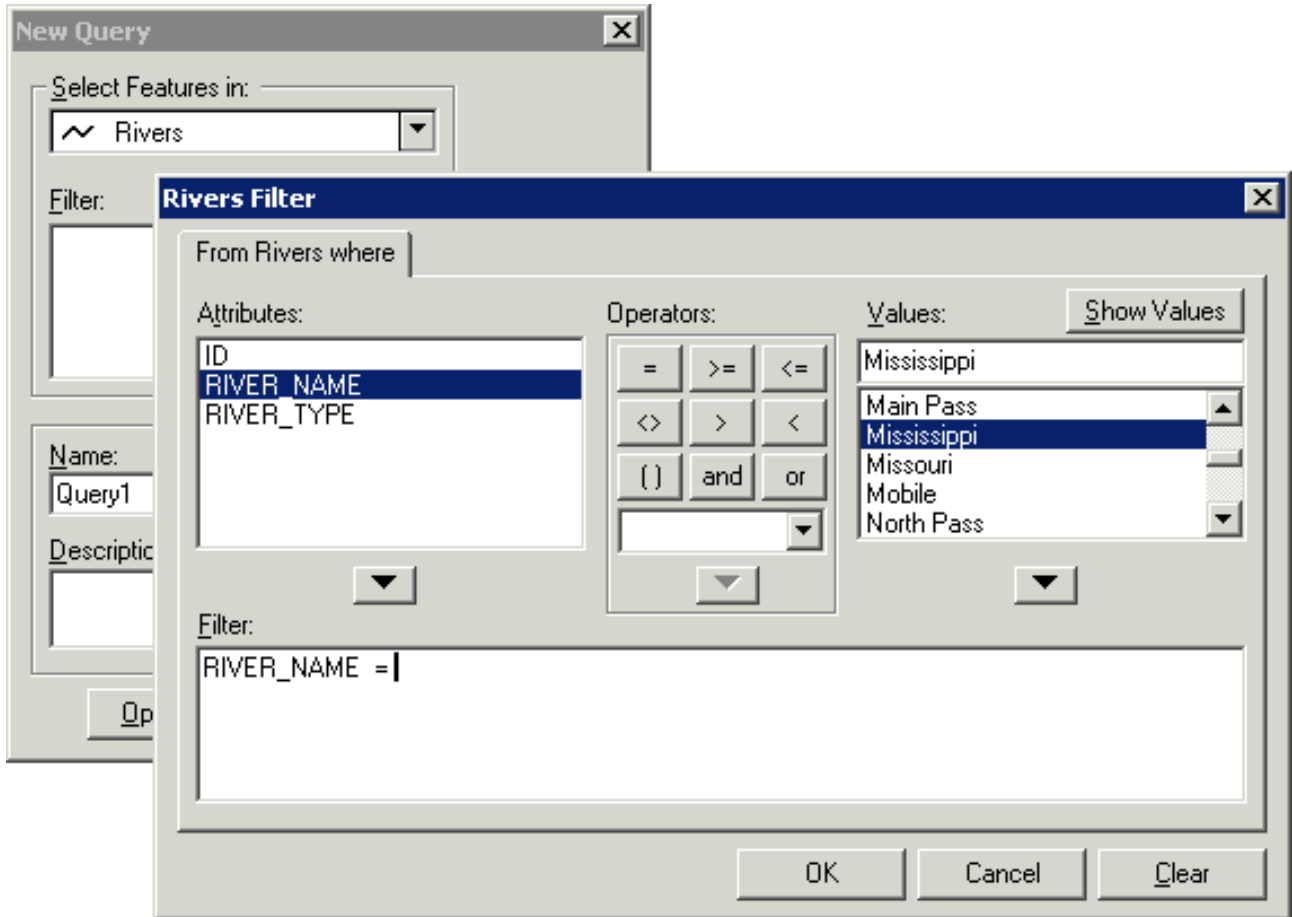
GIS Interface

A GIS user interface often integrates features of a database interface. The database interaction takes place through the combination of different interfaces:

- Graphical interaction via a selection on the map
- Combination of form-based and text-based interaction (e.g. special Query-Wizards for the easier creation of database queries)



Example of a GIS Interface (GeoMedia, Intergraph)



Example of a Query-Wizard within a GIS

1.4. Tasks

Term matching

Drag the keywords below and drop them into the empty rectangles so that each time a pair of matching terms are formed. Click the check button to see if you did it correctly. The reset button sets all the terms back to their original position.

Hint: The terms in this exercise are used in the way they were introduced in this lesson. Literature is likely to use these terms slightly differently.

**Only pictures can be viewed in this version! For Flash, animations, movies etc. see online version.
Only screenshots of animations will be displayed. [link]**

3-Scheme-Architecture

Drag the blue numbers near the keywords onto the correct white circles in the graphic. Click the check button to see if you did it correctly. The reset button sets all the numbers back to their original position.

Hint: The check button only works correctly after covering all of the empty circles in the graphic with a number.

Tip: Start with easy/known keywords.

**Only pictures can be viewed in this version! For Flash, animations, movies etc. see online version.
Only screenshots of animations will be displayed. [link]**

1.5. Exercise Data Independence

Describe in your own words the architecture of a database management system and try to answer the following question. What is data independence and how can it be ensured? If possible, add one or two examples from your knowledge which shows the advantage of data independence.

Your discussion should not be longer than 200-400 words. Post it to the discussion board under the topic 'Exercise Data Independence'. Read and discuss the postings of fellow students too.

1.6. Summary

In this lesson, the most often used database models (relational, object oriented and object relational) and the database technologies based on them were discussed. Such a database model offers some basic rules how a conceptual schema can be transferred in a specific database environment and described by means of a logical scheme. The data description is the base for the following creation and management of the database instances with the actual data. Logical modelling on the base of a relational data model will be discussed in the later lesson [Logical Modelling](#).

Following the generic database architecture based on the 3-scheme-architecture was discussed. This 3-scheme-architecture extends the logical scheme with an internal and an external scheme. The concept of dividing it into three parts is an important prerequisite for the earlier mentioned data independence and the realisation of interfaces for the communication between applications and the database management system. Most of the known database systems offer an interface on the bases of the structured query language SQL. The basic elements and the use of SQL will be discussed in the later lesson [Structured Query Language SQL](#).

1.7. Recommended Reading

- **ZEHNDER, C.A.**, 1998. *Informationssysteme und Datenbanken*. Zürich: vdf Hochschulverlag AG.
Introduction into information systems and databases, including data modelling, in German

1.8. Glossary

Conceptual Data Scheme:

A conceptual data scheme is a system independent data description. That means that it is independent from the database or computer systems used. (Translated) (ZEHNDER 1998)

Conceptual:

A concept is an abstract, universal idea, notion, or entity that serves to designate a category or class of entities, events, or relations. Concepts are abstract in that they omit the differences of the things in their extension, treating them as if they were identical. They are universal in that they apply equally to every thing in their extension. Concepts are also the basic elements of propositions, much the same way a word is the basic semantic element of a sentence.

Entity:

An entity is something that has a distinct, separate existence, though it need not be a material existence. In a relational database an entity is represented as a relation.

External Scheme:

An external data scheme describes the information about the user view of specific users (single users and user groups) and the specific methods and constraints connected with this information. (Translated) (ZEHNDER 1998)

Internal Scheme:

The internal data scheme describes the content of the data and the required service functionality which is used for the operation of the DBMS. (Translated) (ZEHNDER 1998)

Therefore, the internal scheme describes the data from a view very close to the computer or system in general. It completes the logical scheme with data technical aspects like storage methods or help functions for more efficiency.

Logical Data Scheme:

A logical data scheme describes the data in a data definition language DDL of a specific database management system. (Translated) (ZEHNDER 1998)

Logical Independence:

Also the external scheme may stay unchanged for most changes of the logical scheme. This is especially desirable as in this case the application software does not need to be modified or newly translated.

Metadata:

Metadata is literally "data about data", is information that describes another set of data. A common example is a library catalog card, which contains data about the contents and location of a book: It is data about the data in the book referred to by the card. Other common contents of metadata include the source or author of the described dataset, how it should be accessed, and its limitations. Another important type of data about data is the links or relationship among data.

Physical Independence:

Therefore, the logical scheme may stay unchanged even though the storage space or type of some data is changed for reasons of optimisation or reorganisation.

1.9. Bibliography

- **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.