An Introduction to Relational Databases

Chapter 1

Foundations of Information Systems (WS 2008/09)
Strategy in this chapter:

- Practice
- Application oriented
- Hands-on experience

Basic concepts of relational databases

- Theory
- Principles and ideas
- Scientific basis

Both sides are important: Learning without applying is rather useless!
Relational databases: General remarks

- At present, the DB-market is completely dominated by systems supporting the relational model of data.

- Leading (commercial) manufacturers of relational DB-products:
  - Oracle
  - Sybase
  - Microsoft (Access, SQL Server)
  - Postgres (Freeware)
  - IBM (DB2, Informix)
  - MySQL (Freeware)

- The notion "relational" is motivated by the mathematical concept of a relation. Relations in mathematics are sets of tuples.

- Relational databases are collections of one or more relations.

- In practice, relations can be visualized as tables, the rows of which are individual records of data with the same (homogeneous) field structure.

- In science, relational databases have a broad range of theoretical foundations.
The idea to organize data in tables is quite old and pretty obvious.

The idea to investigate this representation of data by means of the theory of relations is due to one man, who proposed this view at the end of the 1960s:

Edgar F. Codd

In 1970, he published his seminal paper "A Relational Model of Data for Large Shared Data Banks", in which he fixed all foundations of relational databases with amazing precision and clarity.

Codd died in early 2003.

For this pioneering work Codd received the Turing Award in 1982, the "Nobel price of informatics".
A strong recommendation for your own studies:

Ramon A. Mata-Toledo, Pauline K. Cushman
“Fundamentals of Relational Databases”
(Schaum's Outlines)
McGraw-Hill Professional
ISBN 978-0071361880
249 pp.
€ 15,99 (amazon.de)

There are many good and expensive academic textbooks on (relational) databases. This one is cheap and not really a high-profile book, but it fits perfectly with our lecture, is up-to-date, very readable and covers exactly what you need.

Everybody should have his/her own copy!
A brief primer in data representation formats and data manipulation paradigms

• Before „diving into“ relational databases proper, we will briefly investigate various competitive formats of representing and manipulating data arranged in **tabular form:**
  - plain text files
  - formatted text files
  - spreadsheets
  - relational databases

• In each **representation format**, the data are stored in files. Such files may well be considered as databases – however, there are different degrees of „database-ness“!

• Each format comes along with a special software system (or program) that controls any kind of access to and manipulation of the respective „database“.

• **Data manipulation** in this context means searching for special data in the file and/or changing (adding, deleting, modifying) data.

• Each of these pairs of representation format + manipulation system can be viewed as a particular variant of the equation DBS = DBMS + DB.
Reminder: database (management) systems

DBMS: Database Management System (application-independent services for managing data)

users and application programmes
A little „case study“: The chemical elements „database“

Representing information/data about the 116 chemical elements in different formats:

- as **textfile** (separated by tabs)
- as **textfile** (separated by semicolons)
- as **Word file**
- as **Excel file**
- as **Access** (relational) database
- The simplest format for representing the chemical elements data is the **text file format**.

- Text files (extension .txt under Windows) are conceptually just long strings of printable symbols (such as digits, characters, or even blanks) arranged in lines.

- By hand, spaces in a text file can be arranged in such a way that e.g. a **tabular structure** (rows-columns) appears.

- Thus, **text files** appear as simple **databases**.

- A **text editor** can be used for performing simple manipulations of the file contents, such as pattern matching and substring replacement: The editor takes the role of a primitive **DBMS**!
• Data in text files can be arranged in any form convenient for humans reading that file.

• The text editor is unable to „see“ the particular structuring convention (e.g. columns).

• Computer programs using the „text DB“ don‘t need visual support either, they just need some means of separating individual parts of the data.

• In this text file version of the elements data, the line structure is retained (one element per line), but columns are just separated by a semicolon delimiter.

• We will deal later with different popular structuring conventions (not made for people, but for programs) such as XML and RDF.
MS Office offers a much more powerful text editor, called **MS Word**, supporting e.g. a „true“ tabular format for data (visualizing rows and columns automatically). Text files managed by the Word software are identifiable by their extension **.doc**.

In addition to the normal functions of a „plain“ text editor, Word offers several „luxury“ variants of text editing.

Word’s ability to search and change data is **not** more powerful than that of „normal“ editors.

Being able to recognize tables, however, makes Word more convenient.
• **Word files are just „illusions“ created by the MS Word system.** If opened with a normal text editor, they turn out to be special text files containing a lot of cryptic special symbols generated (and used) by the Word software.

• The „Word DBMS“ **interprets** these special symbols in order to, e.g., generate the table format not visible in „plain“ text files.

• In addition, Word inserts plenty of other internal code which is needed in order to be able to offer the extra functions not present in a text editor.
• Contained in each Office package on Microsoft PCs, there is an even more powerful tool for managing data, a so-called spreadsheet program called **MS Excel** supporting files with extension `.xls`.

• If processed via Excel, even more details of tabular structure become visible and can be manipulated.

• For searching and changing, the Excel system does not exceed the functionality of Word.

• Excel, however, is specialized in **statistical evaluations** of numerical data.
• Behind the surface, however, there is again a specially formatted text file format, containing plenty of internal control symbols interpreted by the Excel software.

• If opened with a normal text editor, this „hidden“ information becomes visible. It is in principle not different from hand-made separators like semicolons mixed with the „proper“ data parts as seen before.

• What matters is the special software managing these „enhanced text files“, in this case the Excel system.

• Enhanced system functionality requires an enhanced representation format for data.
Each MS Windows computer with MS Office software offers an even more powerful tool for managing data, called **MS Access** – this is the first system computer scientists would call a „real“ DBMS. Files „understood“ by Access have the extension `.mdb`.

Access supports a tabular view of data, too, like Word and Excel, but offers a much, much more powerful set of techniques for searching and changing data.

Access will be considered in more detail in the remainder of this section, dedicated to so-called relational databases.
• Opening an mdb file (alias an Access database) with a text editor reveals the „true nature“ of the representation again: Heavily formatted text file format with excessive use of internal coding interpretable for the MS Access DBMS only.

• Nevertheless, searching e.g. for certain symbols or strings within this file with a text editor returns the same results as searching in the human-friendly tabular text file from the beginning.

• Tricky internal formatting plus intelligent interpreting software is able to generate powerful illusions about databases!
In the remainder of this chapter, we will use the last of these representation formats only:

- **Relational Databases**

In addition, we will forget about text editors, Word and Excel, and explore the power of **Access**, a true **relational DBMS**.

In the companion lecture by Prof. Hofmann-Apitius you will get to know a wide spectrum of additional data representation formats, many of them developed particularly for life science applications („dedicated formats“).

All of these formats are ultimately based on text files as underlying „real“ format. Special structuring information is always interleaved with „plain“ data – as was shown for the general-purpose formats .doc, .xls, and .mdb discussed before.

In addition, most of these dedicated formats comes with its own „gatekeeper“ software, comparable to Word, Excel or Access in that it interprets the special format in a particular, system-specific way.
Our example DBMS: Microsoft Access

### Access

- **Access** is a DBMS for relational databases (data organized in form of tables), developed and distributed since 1992 by Microsoft.
- "Access-Homepage" at Microsoft:
- recent version in MS Office packages: Access 2000
- Access is very well-suited for small to medium DB applicatons in single-user mode.
- useful internet tutorials on Access
  - Michael Brydon‘s tutorial at Simon Fraser University, Canada
    - [http://mis.bus.sfu.ca/tutorials/MSAccess/tutorials.html](http://mis.bus.sfu.ca/tutorials/MSAccess/tutorials.html)
  - Maggie Strapland‘s Access pages at University of Bristol, UK
    - [http://www.bris.ac.uk/is/services/software/packages/access/](http://www.bris.ac.uk/is/services/software/packages/access/)
  - Jakob Lindenmeyer‘s Access tutorial at ETH Zürich, Schweiz
- In addition, there are many, many books on how to use Access, most of them not really that helpful, because there is poor structure and too many details.

Throughout the course, we will use a small, but handy DBMS available on most PCs.
A first example database: European geography

- Our first „real world“ database example is about geography: Facts about countries, cities etc. in Europe!

- A wealth of geo data can be accessed freely on the web in the „World Fact Book“ of the CIA:  

- You will find a database called „europe.mdb“ on the lecture homepage for your own „experiments“. This will continuously grow – you are invited to help!

- At this moment in our lecture, the geo database serves as a first „appetizer“ to (relational) database management.
A relational database about European geography

Europe.mdb is a small database for introductory purposes.

Just now it contains **two tables**, one on **countries** in Europe, the other on **cities**.

Today we will just learn the most basic ideas about **relational databases** – some of you will be already familiar with this.

And we will discuss the important question why a **simple text file** is not sufficient for keeping data.
Tables in Access: some basic terminology

Relational **tables are grids**, the fields of which are consisting of columns and rows.
There is a specific terminology for such tables in Access.
Terminological confusion possible!

Unfortunately, the basic concepts of the relational model are denoted by different terms depending on the context. There are synonymous, but different terminologies in database theory, the standard DB language SQL and MS Access:

<table>
<thead>
<tr>
<th>theory</th>
<th>SQL</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>relation</td>
<td>table</td>
<td>datasheet</td>
</tr>
<tr>
<td>tuple</td>
<td>row</td>
<td>record</td>
</tr>
<tr>
<td>attribute</td>
<td>column</td>
<td>field name</td>
</tr>
<tr>
<td>domain</td>
<td>data type</td>
<td>field data type</td>
</tr>
</tbody>
</table>

Be warned of this „Babylonic confusion“ of terms – we urgently recommend that you always stick to a single system of notions in a consistent manner. It doesn‘t matter which system you use – but never mix them up!
Access table: Datasheet view

Access table „countries“:

Datasheet view

<table>
<thead>
<tr>
<th>country</th>
<th>code</th>
<th>capital</th>
<th>area</th>
<th>population</th>
</tr>
</thead>
<tbody>
<tr>
<td>Austria</td>
<td>A</td>
<td>Vienna</td>
<td>83850</td>
<td>8023244</td>
</tr>
<tr>
<td>Albania</td>
<td>AL</td>
<td>Tirane</td>
<td>28750</td>
<td>3249136</td>
</tr>
<tr>
<td>Andorra</td>
<td>AND</td>
<td>Andorra la Vella</td>
<td>450</td>
<td>72766</td>
</tr>
<tr>
<td>Belgium</td>
<td>B</td>
<td>Brussels</td>
<td>30510</td>
<td>10170241</td>
</tr>
<tr>
<td>Bulgaria</td>
<td>BG</td>
<td>Sofia</td>
<td>110910</td>
<td>8612757</td>
</tr>
<tr>
<td>Bosnia and Herzegovina</td>
<td>BIH</td>
<td>Sarajevo</td>
<td>51233</td>
<td>2656240</td>
</tr>
<tr>
<td>Belarus</td>
<td>BY</td>
<td>Minsk</td>
<td>207600</td>
<td>10415973</td>
</tr>
<tr>
<td>Switzerland</td>
<td>CH</td>
<td>Bern</td>
<td>41290</td>
<td>7207060</td>
</tr>
<tr>
<td>Czech Republic</td>
<td>CZ</td>
<td>Prague</td>
<td>78703</td>
<td>10321120</td>
</tr>
<tr>
<td>Germany</td>
<td>D</td>
<td>Berlin</td>
<td>356910</td>
<td>83536115</td>
</tr>
<tr>
<td>Denmark</td>
<td>DK</td>
<td>Copenhagen</td>
<td>43070</td>
<td>5249632</td>
</tr>
<tr>
<td>Spain</td>
<td>E</td>
<td>Madrid</td>
<td>504750</td>
<td>39181114</td>
</tr>
<tr>
<td>Estonia</td>
<td>EW</td>
<td>Tallinn</td>
<td>45100</td>
<td>1459428</td>
</tr>
<tr>
<td>France</td>
<td>F</td>
<td>Paris</td>
<td>547030</td>
<td>58317450</td>
</tr>
<tr>
<td>Liechtenstein</td>
<td>FL</td>
<td>Vaduz</td>
<td>160</td>
<td>31122</td>
</tr>
<tr>
<td>United Kingdom</td>
<td>GB</td>
<td>London</td>
<td>244820</td>
<td>58489975</td>
</tr>
<tr>
<td>Greece</td>
<td>GR</td>
<td>Athens</td>
<td>131940</td>
<td>10538594</td>
</tr>
<tr>
<td>Hungary</td>
<td>H</td>
<td>Budapest</td>
<td>93030</td>
<td>10002541</td>
</tr>
<tr>
<td>Croatia</td>
<td>HR</td>
<td>Zagreb</td>
<td>56538</td>
<td>5004112</td>
</tr>
<tr>
<td>Italy</td>
<td>I</td>
<td>Rome</td>
<td>301230</td>
<td>57460271</td>
</tr>
</tbody>
</table>

Datensatz: 14 von 44

Datenblattansicht
Access table: Design view

Access table „countries“:

Design view

- field size
- format
- input mask
- caption
- default value
- validation rule
- validation text
- Required
- Allow zero length
- Indexed
- Unicode compression
Some important icons in Access

- In design view:

1. Switch to datasheet view
2. Save table design

- In datasheet view:

1. Switch to design view
12/13. Order data in ascending or descending order
17. Search via "pattern matching" in data sheet
The two different "views" of a table in Access correspond to two fundamental notions of relational databases:

- **Schema** of a relation: definition of name and structure of the relation
- **State** of a relation: all tuples currently contained in the relation
- The structure of each state of a relation is defined by its schema. (States are called instances of the schema.)
- In general, the schema remains fixed during state changes.
- Sometimes, however, there are schema modifications as well, followed by immediate state adaptations: schema evolution
- Plural of schema: schemas (not "schemes")!
Schemas and states

Schema_1

States

Instances of schema_1

Schema_2

Instances of schema_2

Current state
Interaction with databases: principle

There are two basic forms of interaction with a database:

- **Reading access:** Query mode
- **Writing access:** Update mode
There are various ways to read from a DB – only few DBS support all of them!

- **Simple** forms of "retrieval":
  - "Browsing" the records of a table: manually inspecting one record after the other.
  - Looking for some/all records containing a particular string pattern in a particular field of the table: "pattern matching"

- **Complex** forms of "retrieval":
  - Finding all records of a table satisfying a complex search condition, formulated in a special language: "querying"

File systems support browsing and pattern matching, but only database systems allow for querying!
• A fundamental characteristic of each database management system is its support of one or more query languages.

• A query is an expression in this language which . . .
  • . . . is able to express arbitrarily complex search criteria.
  • . . . refers to one or more tables simultaneously.
  • . . . returns one or more records or simply yes/no as an answer.
  • . . . returns records in form of answer tables.

• Access offers two very different query languages representing two completely different query paradigms:
  • Graphically-interactive: "Query-by-Example" (QBE)
  • Textual: "Structured Query Language" (SQL)

• SQL is the most widely distributed query language for relational DBs.

• SQL is standardized and is „understood“ by any commercial DBMS.
Pattern matching is a rather primitive mode of retrieval:

You can just locate fields containing a particular string of symbols, no more.
Retrieve name, capital and area of all countries larger than 100000 km² in descending order of size!

Formulation in QBE-style:

- Graphically represented
- Interactively constructed

In Access, called design view as well.
Retrieve name, capital and area of all countries larger than 100000 km² in descending order of size!

Answers to relational queries are always returned as tables, too.

Thus, they may be „reused“ as input for further queries.

However, these tables are not stored in the DB! They are „virtual“ tables recomputed each time the query is asked.

Datasheet view, too.
SQL-style queries

The same query expressed in textual style as an SQL table expression:
Changes in the query formulation in one view are automatically and immediately passed to the other view: Both representations are fully synchronized!
## Multi-table queries

### Countries Table
<table>
<thead>
<tr>
<th>country</th>
<th>code</th>
<th>capital</th>
<th>area</th>
</tr>
</thead>
<tbody>
<tr>
<td>Austria</td>
<td>A</td>
<td>Vienna</td>
<td>83850</td>
</tr>
<tr>
<td>Albania</td>
<td>AL</td>
<td>Tirane</td>
<td>28750</td>
</tr>
<tr>
<td>Andorra</td>
<td>AND</td>
<td>Andorra la Vella</td>
<td>450</td>
</tr>
<tr>
<td>Belgium</td>
<td>B</td>
<td>Brussels</td>
<td>30510</td>
</tr>
<tr>
<td>Bulgaria</td>
<td>BG</td>
<td>Sofia</td>
<td>110910</td>
</tr>
<tr>
<td>Bosnia and Herzegovina</td>
<td>BH</td>
<td>Sarajevo</td>
<td>51233</td>
</tr>
<tr>
<td>Belarus</td>
<td>BY</td>
<td>Minsk</td>
<td>207600</td>
</tr>
<tr>
<td>Switzerland</td>
<td>CH</td>
<td>Bern</td>
<td>41290</td>
</tr>
<tr>
<td>Czech Republic</td>
<td>CZ</td>
<td>Prague</td>
<td>78703</td>
</tr>
<tr>
<td>Germany</td>
<td>D</td>
<td>Berlin</td>
<td>356910</td>
</tr>
<tr>
<td>Denmark</td>
<td>DK</td>
<td>Copenhagen</td>
<td>43070</td>
</tr>
<tr>
<td>Spain</td>
<td>E</td>
<td>Madrid</td>
<td>504750</td>
</tr>
<tr>
<td>Estonia</td>
<td>EW</td>
<td>Tallinn</td>
<td>45100</td>
</tr>
<tr>
<td>France</td>
<td>F</td>
<td>Paris</td>
<td>547030</td>
</tr>
<tr>
<td>Liechtenstein</td>
<td>FL</td>
<td>Vaduz</td>
<td>160</td>
</tr>
<tr>
<td>United Kingdom</td>
<td>GB</td>
<td>London</td>
<td>244820</td>
</tr>
<tr>
<td>Greece</td>
<td>GR</td>
<td>Athens</td>
<td>131940</td>
</tr>
<tr>
<td>Hungary</td>
<td>H</td>
<td>Budapest</td>
<td>93030</td>
</tr>
<tr>
<td>Croatia</td>
<td>HR</td>
<td>Zagreb</td>
<td>56530</td>
</tr>
<tr>
<td>Italy</td>
<td>I</td>
<td>Rome</td>
<td>301338</td>
</tr>
</tbody>
</table>

### Cities Table
<table>
<thead>
<tr>
<th>city</th>
<th>population</th>
<th>year</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amsterdam</td>
<td>731268</td>
<td>2000</td>
</tr>
<tr>
<td>Andorra la Vella</td>
<td>20787</td>
<td>2001</td>
</tr>
<tr>
<td>Ankara</td>
<td>3203362</td>
<td>2000</td>
</tr>
<tr>
<td>Athens</td>
<td>772072</td>
<td>1991</td>
</tr>
<tr>
<td>Belgrade</td>
<td>1597599</td>
<td>1997</td>
</tr>
<tr>
<td>Berlin</td>
<td>3386667</td>
<td>2000</td>
</tr>
<tr>
<td>Bern</td>
<td>122469</td>
<td>2001</td>
</tr>
<tr>
<td>Bratislava</td>
<td>447345</td>
<td>2000</td>
</tr>
<tr>
<td>Brussels</td>
<td>964405</td>
<td>2001</td>
</tr>
<tr>
<td>Bucharest</td>
<td>1921751</td>
<td>2002</td>
</tr>
<tr>
<td>Budapest</td>
<td>1811522</td>
<td>2000</td>
</tr>
<tr>
<td>Chisinau</td>
<td>778800</td>
<td>2000</td>
</tr>
<tr>
<td>Copenhagen</td>
<td>500531</td>
<td>2002</td>
</tr>
<tr>
<td>Dublin</td>
<td>495101</td>
<td>2002</td>
</tr>
<tr>
<td>Helsinki</td>
<td>559718</td>
<td>2001</td>
</tr>
<tr>
<td>Kiev</td>
<td>2637100</td>
<td>2001</td>
</tr>
<tr>
<td>Lisbon</td>
<td>556797</td>
<td>2001</td>
</tr>
<tr>
<td>London</td>
<td>714850</td>
<td>2001</td>
</tr>
</tbody>
</table>
One of the main advantages of using a query language is the ability to formulate multi-table queries.

Tables are „joined“ by marking certain fields with identical type thus forcing tuples with identical values in these join fields to appear in the answer table.
Multi-table queries (3)

Which is the population of the capitals of the European countries?

The answer table combines fields from both input tables and arranges them in a newly defined manner.

<table>
<thead>
<tr>
<th>country</th>
<th>capital</th>
<th>population</th>
</tr>
</thead>
<tbody>
<tr>
<td>Albania</td>
<td>Tirane</td>
<td>427000</td>
</tr>
<tr>
<td>Greece</td>
<td>Athens</td>
<td>772072</td>
</tr>
<tr>
<td>Macedonia</td>
<td>Skopje</td>
<td>444300</td>
</tr>
<tr>
<td>Serbia and Montenegro</td>
<td>Belgrade</td>
<td>1597599</td>
</tr>
<tr>
<td>Andorra</td>
<td>Andorra la Vella</td>
<td>20787</td>
</tr>
<tr>
<td>France</td>
<td>Paris</td>
<td>2115757</td>
</tr>
<tr>
<td>Spain</td>
<td>Madrid</td>
<td>2938723</td>
</tr>
<tr>
<td>Austria</td>
<td>Vienna</td>
<td>1550123</td>
</tr>
<tr>
<td>Czech Republic</td>
<td>Prague</td>
<td>1178576</td>
</tr>
<tr>
<td>Germany</td>
<td>Berlin</td>
<td>3386667</td>
</tr>
<tr>
<td>Hungary</td>
<td>Budapest</td>
<td>1811522</td>
</tr>
<tr>
<td>Italy</td>
<td>Rome</td>
<td>2655970</td>
</tr>
<tr>
<td>Liechtenstein</td>
<td>Vaduz</td>
<td>4949</td>
</tr>
<tr>
<td>Slovakia</td>
<td>Bratislava</td>
<td>447315</td>
</tr>
</tbody>
</table>
You may even **store a query** likely to be asked again and again.

Storing a query means to store its **design**, not its answer table!
Using a stored query

Stored queries may serve as input for subsequent queries in the same way as tables are.

Further refinement of the query
Access icons for query management

- **in design view:**
  1. Switch to answer table (datasheet view)
  2. Store query design
  12. Compute answer table (and switch to datasheet view)

- **in datasheet view (answer table):**
  1. Switch to design view
  17. Search via "pattern matching" in data sheet

3rd alternative in menu behind this icon: **SQL view**

Queries, not tables!!
**Aggregation and grouping**

- Important basic functionality of DB query languages:
  Computation of **numerical summary values** referring to certain fields in a table (e.g., cardinality, sum, average, largest/smallest value)

- Corresponding arithmetic functions: **Aggregate functions**

- In Access design mode for queries: By clicking on the **function symbol \( \Sigma \)**, aggregation mode is activated and aggregate functions can be selected.

- Application of aggregate functions usually requires subdividing the resp. table into groups according to values of a particular field prior to applying the function to each of these groups:

- **Example**: Maximum of the values in field B per value in field A

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>a1</td>
<td>3</td>
</tr>
<tr>
<td>a1</td>
<td>5</td>
</tr>
<tr>
<td>a2</td>
<td>6</td>
</tr>
<tr>
<td>a2</td>
<td>9</td>
</tr>
<tr>
<td>a2</td>
<td>1</td>
</tr>
</tbody>
</table>

- **Aggregation**: maximum: 5
- **Grouping**: maximum: 9
In order to design a query containing an **aggregate function**:

- Click on symbol Σ:
  - additional line „function:“ appears in definition area
- Choose field for **grouping**:
  - per country
- Choose **aggregate function**:
  - max(imum) on field population in table „cities“

**Important:**

It is **impossible** in Access to include other fields not either grouped, restricted by a condition or aggregated upon into a query containing aggregation! (e.g., city name cannot be included just for display purposes)
What is the population of the biggest city in each country?

Answer table to the query designed on the previous slide

(Note that we did not ask „Which is the biggest city in each country?“ as this would violate the exclusion of „display-only fields“ from aggregate queries)
How to include a non-grouping and non-aggregating field?

Find name and population of the biggest city in each country?

Reuse result of the previous query just determining the size of the biggest city – thus reformulating the above query in a complicated (but aggregate-free) manner:

Find name and population of that city in each country, which is equal in size to the biggest city in that country?
Result of the „biggest_city“ query:

City name appears, too!

The somehow „exotic“ technique used for being able to express this query in QBE style is useful for more complex SQL queries with aggregates as well!
State changes: general principles

- "Write" access to a database . . .
  - . . . always results in a state change of the DB.
  - . . . always takes place under control of the DBMS.

- There are three basic forms of write access:
  - insertions of new records into a table
  - deletions of existing records from a table
  - modifications of the value of a particular field in a record of a table

- Insertions and modifications are accepted by the DBMS only if the data types of the resp. fields declared in the schema of the table fit with the values in the new/modified records.

- Caution! The English notion "update" is used in this context with two different meanings – be sure you understand which of them is actually meant:
  - as a synonym for modification
  - as a generalization comprising all three kinds of write access

- In Access, individual updates can be performed directly in the datasheet view by manipulating the individual fields and records.
Updating by direct manipulation in Access

Record to be inserted into table cities (during insert)

Update mode indicated by write icon
Action queries

- Many records in a table can be updated *simultaneously* if they are identified by means of a query.
- For doing so, a special type of query is evaluated, called an action query.
- There are four types of action:
  - append
  - delete
  - modify
  - make table

- Example (modify query):
  Increase the population of every city by 100, if the year value is older than 2000!

- Result:
  - Candidate cities are identified by evaluating criteria conditions.
  - Modification action is applied to all qualifying records!
More on designing tables

Access table „countries“:  
**Design view**

In the following, we will focus more closely on some of the options for designing tables and their fields:

<table>
<thead>
<tr>
<th>field name</th>
<th>field data type</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Country</td>
<td>Text</td>
<td></td>
</tr>
<tr>
<td>Code</td>
<td>Text</td>
<td></td>
</tr>
<tr>
<td>Capital</td>
<td>Text</td>
<td></td>
</tr>
<tr>
<td>Area</td>
<td>Zahl</td>
<td></td>
</tr>
<tr>
<td>Population</td>
<td>Zahl</td>
<td></td>
</tr>
<tr>
<td>Year</td>
<td>Zahl</td>
<td></td>
</tr>
</tbody>
</table>

- **Field size**
- **Format**
- **Input mask**
- **Caption**
- **Default value**
- **Validation rule**
- **Validation text**
- **Required**
- **Allow zero length**
- **Indexed**
- **Unicode compression**
For each field in a table, a field data type has to be declared, e.g. text, number, date, yes/no.

For many of these types more detailed variations concerning the size of memory required for the resp. field can be declared in addition.

In the example:
Field 'code' is of type text, but country codes have at most three characters.

For type 'number': Various subtypes can be chosen from a menu accessible by clicking into the field size entry (e.g., integer, long integer, byte)
• For each field, a **default value** can be defined.
• This value is automatically inserted into every new record in the resp. field in case no explicit value is given during insertion.
• **In the example:**
  Value '0' is defined for numerical field 'population'!
• If no default value is given, fields can remain empty, unless the field is declared as 'Required',
• Empty fields can be imagined to contain a special „invisible“ value (not contained in any data type) called a **null value**. Null values cannot be identified with any other value and are not counted by aggregate functions.
• In most tables, there is one or more fields the values of which uniquely identify a particular record of the table.
• Such special fields (or combinations of fields) are called keys of the table.
• One such key ought to be designated at design time as the primary key of the table.
• In the example, the field 'code' is a key of table "countries", as each country is described in exactly one record identified by a unique country code.
• Marking the field and then clicking on the key symbol in the DB symbol list designates a primary key.
Violating a key constraint

If a new record is to be inserted into a table which has the same primary key value as an already existing record, the insertion is rejected by the DBMS!

This kind of control is a case of integrity checking.
Validation rules

- Another means of controlling the contents of a DB table is the concept of a validation rule.
- Such a rule can be associated with each field of a table (here with 'area' in "countries").
- A validation rule is a logical condition defining one or more properties of each proper value in this field.
- In the example, the area value is restricted to positive integers smaller than 10000000.
- In case of a violation of a rule while inserting a new record or modifying an existing value, the modification is rejected and a predefined validation text is displayed.
- The syntax of validation rules will be discussed in more detail during the exercises.
Violating a validation rule

On violation of a validation rule, the predefined validation text is displayed, and the modification is rejected!
Integrity constraints and integrity checking

- Primary key definitions and validation rules are special examples of a very important general concept in database design:

  Integrity constraints

- In general, an integrity constraint (constraint for short) is a logical condition to be satisfied by each state of the database at all times, i.e., integrity constraints are required to be invariantly true during the lifetime of the database.

  Integrity checking

- In SQL, we will find a rather powerful language for expressing nearly arbitrary such conditions. In QBE style, Access supports only few of the most important special cases.

- Integrity constraint violations – likely to happen during DB modifications – are controlled automatically by the DBMS. Each insertion, deletion or update of a table is checked for possibly violating any constraint prior to the execution of the resp. modification:

  Integrity checking

- If integrity violations are detected, the DBMS either refuses to perform the desired modification or „repairs“ the semantic mistake causing the violation automatically, if possible.

- Key and validation rule violations cannot be „repaired“!
A third type of integrity constraint can be established, if relationships between tables have been declared before.

By clicking on the relationship icon in the main icon list of an Access database, you can switch to relationship design mode, which looks like this...
• In our example database on European geography, two relationships have been declared between the two tables.
• As both tables are involved both as referencing as well as as referenced table, one of the has to occur twice.
• Relationships are established between fields of the table which have identical data types.
• One of them – the referenced field – has to be a key of the referenced table. It is indicated by a ‘1’ in the graphical form.
• The other field (usually marked by the infinity symbol indicating arbitrarily many occurrences) is any field in the referencing table.
• Each relationship between two field in two tables can be associated with a **referential integrity constraint**.
• Clicking on the relationship line causes a window to open.
• Here, referential integrity can be activated.
• In addition, two kinds of „repair activities“ can be chosen for cases of integrity violation:
  • Changes to the referenced field are propagated to the referencing field
  • Analoguously for deletions
Repairing a referential integrity violation

Before the modification . . .

... referential integrity holds.
Repairing a referential integrity violation (2)

Modification in „master table“:
Tirane  ⇒ Tirana

Modification is propagated to dependent table:
**Referential action**
Exploiting relationships while browsing a table

- Relationships can be exploited while browsing the datasheet view of a table.
- An extra field (automatically generated) next to the primary key of a referenced table contains a +/- icon.
- This icon can be activated in order to open a subtable containing all records referring to this particular key value.
- In the example: „Countries“ record referencing the resp. city via the link on field ,capital“
Relational databases in Access: Summary

- **Goal** of this chapter:
  - introduction to practical use of a relational DB by means of MS Access
  - illustration of the most important concepts and notions via examples

- **Summary** of the notions/concepts mentioned:

<table>
<thead>
<tr>
<th>data model</th>
<th>DB query</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB schema</td>
<td>query language</td>
</tr>
<tr>
<td>DB state</td>
<td>subquery</td>
</tr>
<tr>
<td>relation/table/datasheet</td>
<td>action query</td>
</tr>
<tr>
<td>attribute/column/field</td>
<td>integrity constraint</td>
</tr>
<tr>
<td>tuple/row/record</td>
<td>validation rule</td>
</tr>
<tr>
<td>domain/(field) data type</td>
<td>primary key</td>
</tr>
<tr>
<td>null value</td>
<td>foreign key</td>
</tr>
<tr>
<td>default value</td>
<td>referential integrity</td>
</tr>
<tr>
<td>relationship</td>
<td></td>
</tr>
</tbody>
</table>

- **In chapter 2**: More detailed introduction to the other style of query formulation supported by Access via **SQL** (Structured Query Language)