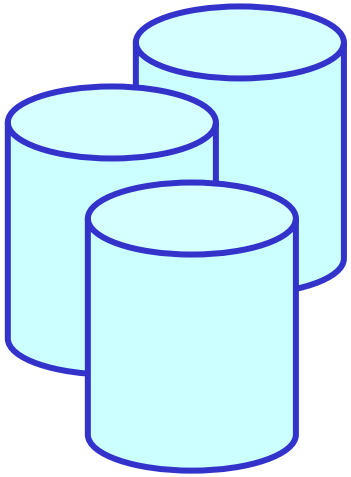
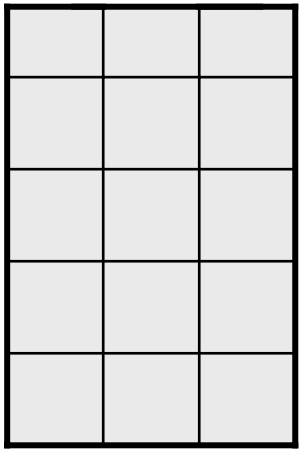
Foundations of Information Systems (WS 2008/09)

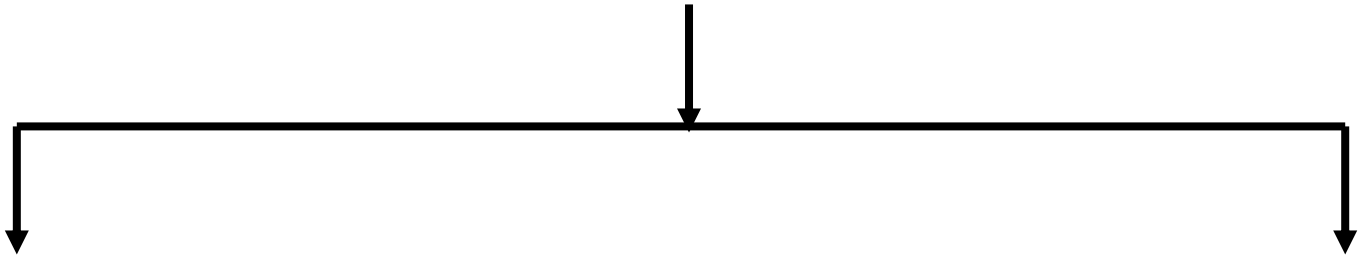


– Chapter 1 –

An Introduction to Relational Databases



Strategy in this chapter:



Microsoft Access for Beginners

- 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 !

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

- Leading (commercial) manufacturers of relational DB-products:

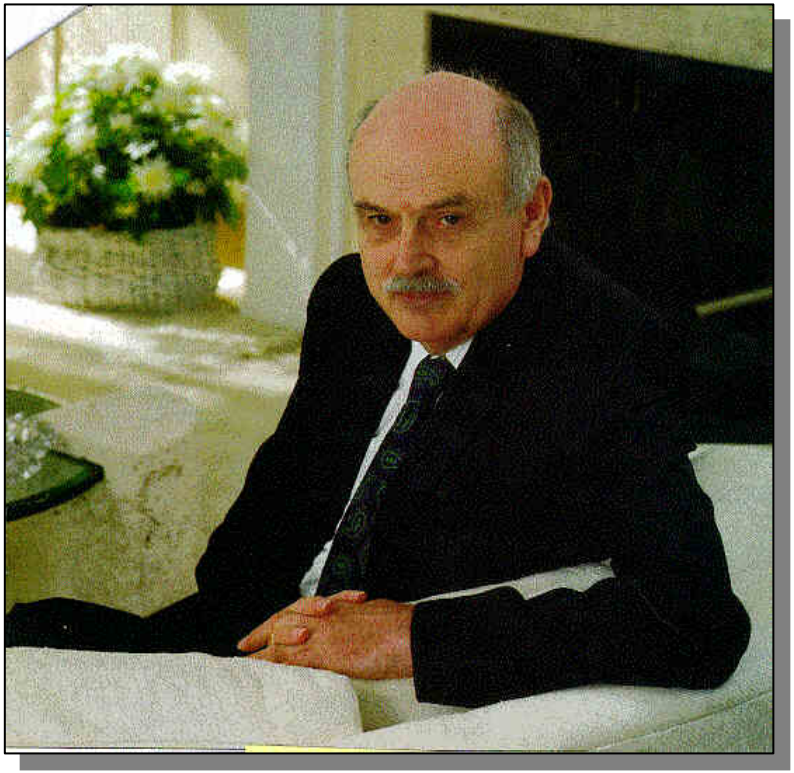
Oracle	Sybase
Microsoft (Access, SQL Server)	Postgres (Freeware)
IBM (DB2, Informix)	MySQL (Freeware)

- 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.



Edgar F. Codd

For this pioneering work Codd received the Turing Award in 1982, the „Nobel price of informatics".

- 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:

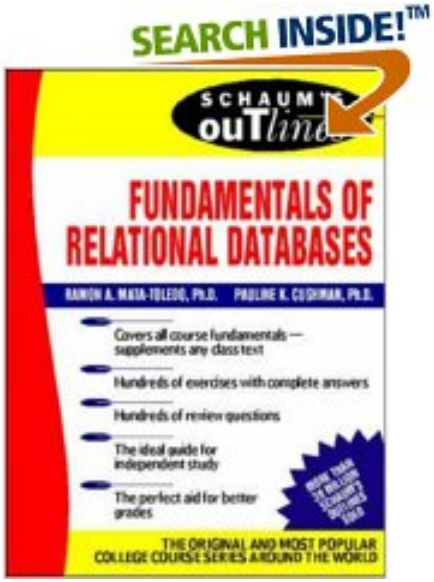
Edgar F. Codd

- 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.



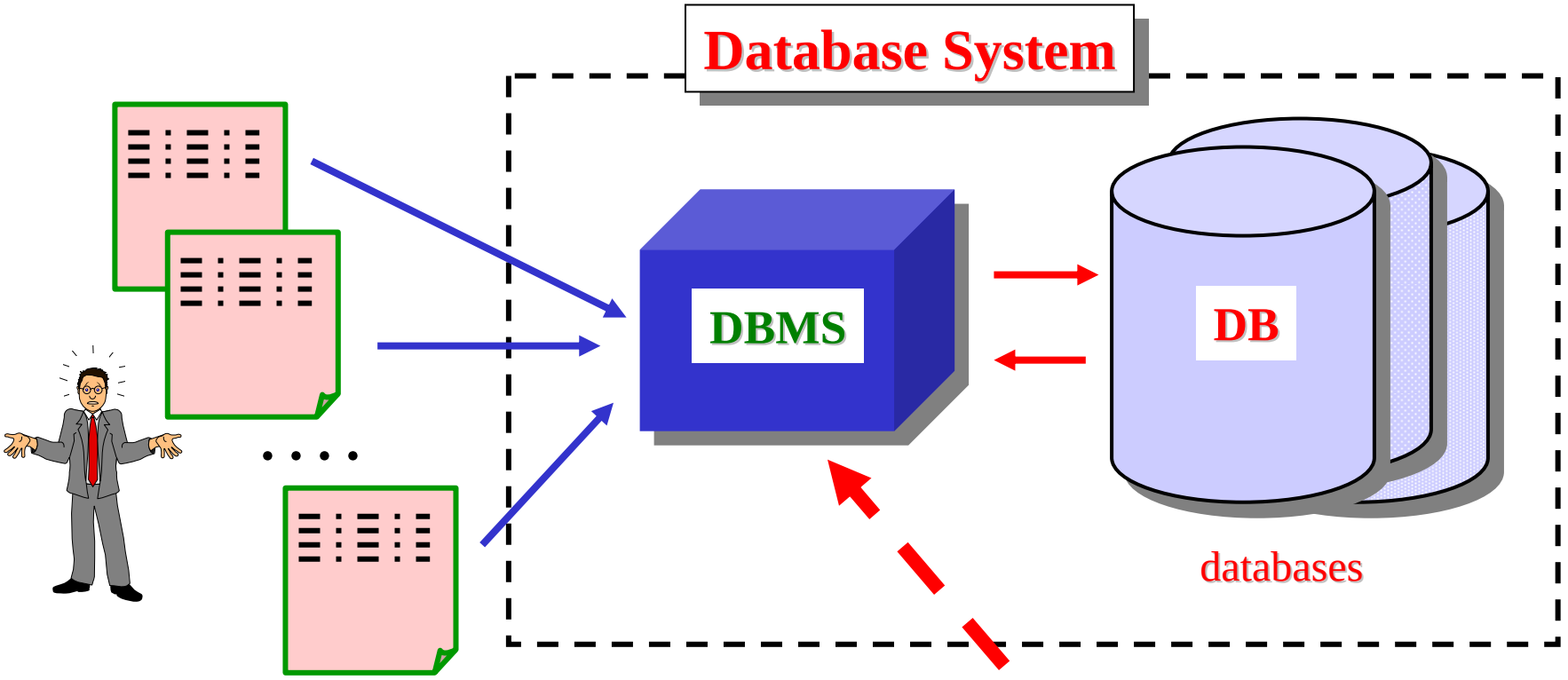
A strong recommendation for your own studies:

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!

- 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.



users and
application programmes

DBMS: **Database Management System**

(application-independent services for managing data)

Group →	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
↓ Period																		
1	1 H																	2 He
2	3 Li	4 Be											5 B	6 C	7 N	8 O	9 F	10 Ne
3	11 Na	12 Mg											13 Al	14 Si	15 P	16 S	17 Cl	18 Ar
4	19 K	20 Ca	21 Sc	22 Ti	23 V	24 Cr	25 Mn	26 Fe	27 Co	28 Ni	29 Cu	30 Zn	31 Ga	32 Ge	33 As	34 Se	35 Br	36 Kr
5	37 Rb	38 Sr	39 Y	40 Zr	41 Nb	42 Mo	43 Tc	44 Ru	45 Rh	46 Pd	47 Ag	48 Cd	49 In	50 Sn	51 Sb	52 Te	53 I	54 Xe
6	55 Cs	56 Ba	*	72 Hf	73 Ta	74 W	75 Re	76 Os	77 Ir	78 Pt	79 Au	80 Hg	81 Tl	82 Pb	83 Bi	84 Po	85 At	86 Rn
7	87 Fr	88 Ra	**	104 Rf	105 Db	106 Sg	107 Bh	108 Hs	109 Mt	110 Ds	111 Rg	112 Uub	113 Uut	114 Uuq	115 Uup	116 Uuh	117 Uus	118 Uuo
* Lanthanides				57 La	58 Ce	59 Pr	60 Nd	61 Pm	62 Sm	63 Eu	64 Gd	65 Tb	66 Dy	67 Ho	68 Er	69 Tm	70 Yb	71 Lu
** Actinides				89 Ac	90 Th	91 Pa	92 U	93 Np	94 Pu	95 Am	96 Cm	97 Bk	98 Cf	99 Es	100 Fm	101 Md	102 No	103 Lr

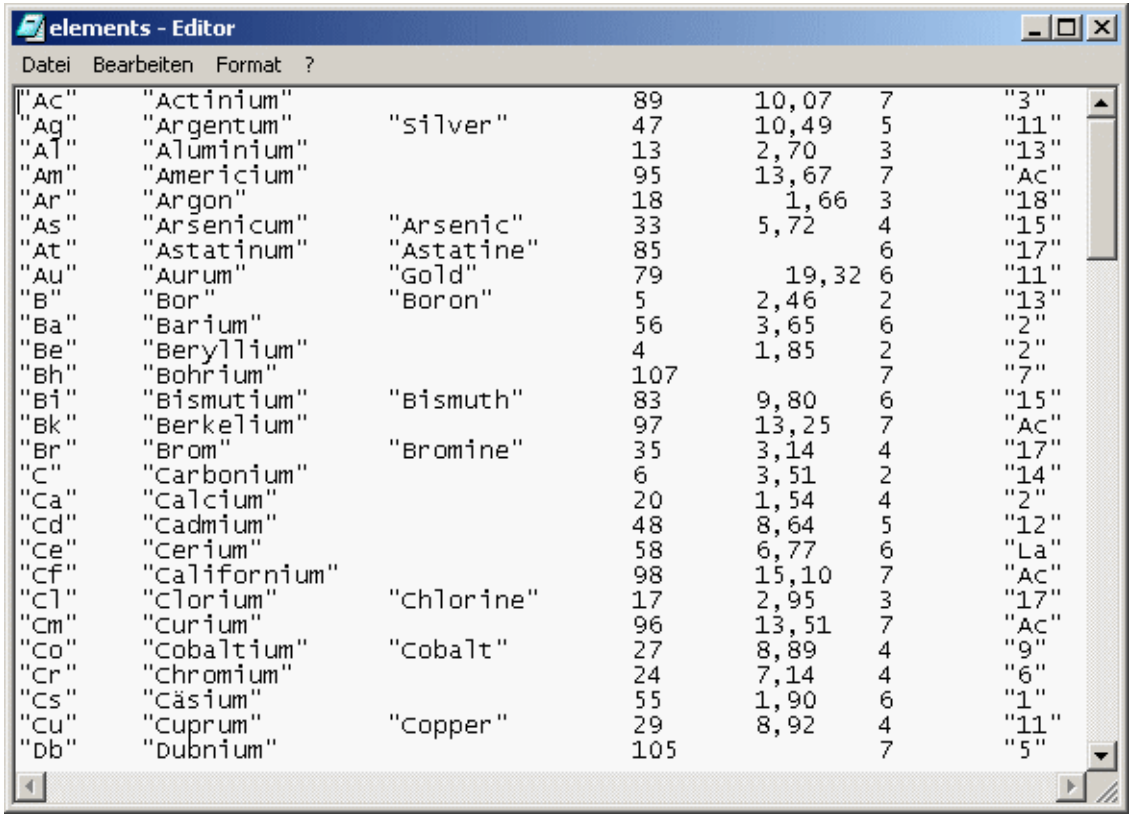
Chemical series of the periodic table

Alkali metals[2]	Alkaline earth metals[2]	Lanthanides[1,2]	Actinides[1,2]	Transition metals[2]
Poor metals	Metalloids	Nonmetals	Halogens[3]	Noble gases[3]

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

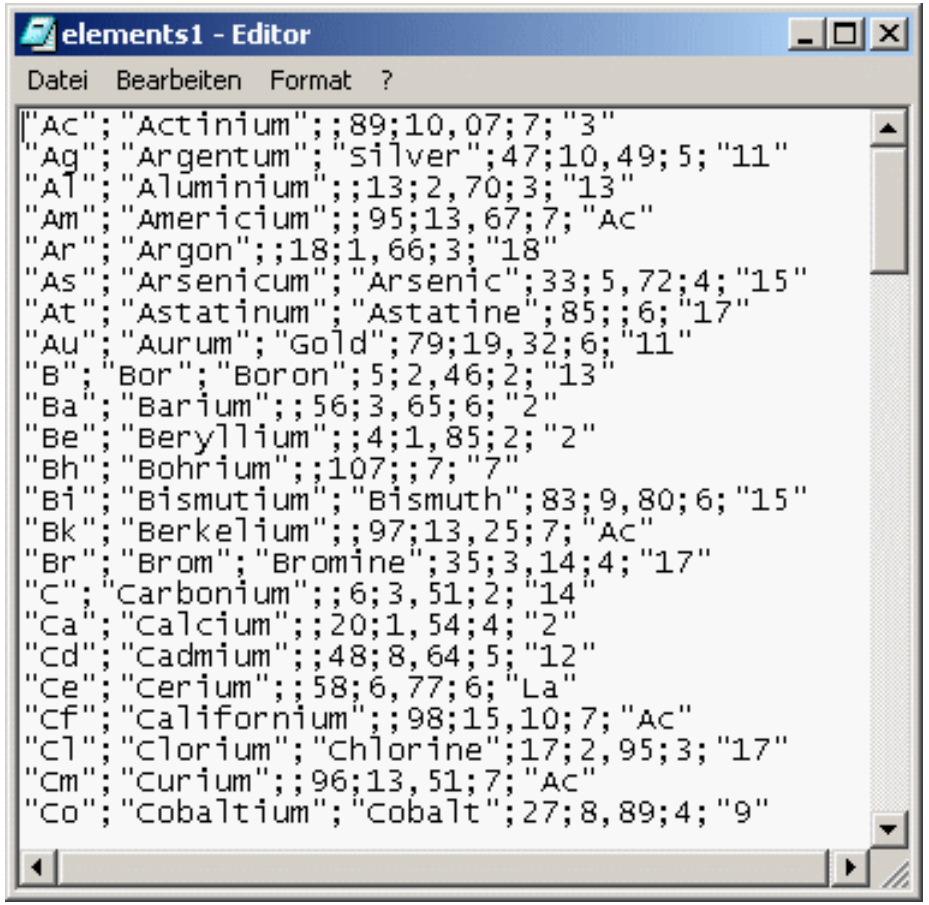
- as textfile (separated by tabs)
- as textfile (separated by semicolons)
- as Word file
- as Excel file
- as Access (relational) database

- 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**!




```
"Ac"    "Actinium"                      89      10,07   7       "3"
"Ag"    "Argentum"      "Silver"        47      10,49   5       "11"
"Al"    "Aluminium"                     13      2,70    3       "13"
"Am"    "Americium"                     95      13,67   7       "Ac"
"Ar"    "Argon"                         18        1,66  3       "18"
"As"    "Arsenicum"     "Arsenic"       33      5,72    4       "15"
"At"    "Astatinum"     "Astatine"      85              6       "17"
"Au"    "Aurum"         "Gold"          79        19,32 6       "11"
"B"     "Bor"           "Boron"         5       2,46    2       "13"
"Ba"    "Barium"                        56      3,65    6       "2"
"Be"    "Beryllium"                     4       1,85    2       "2"
"Bh"    "Bohrium"                       107             7       "7"
"Bi"    "Bismutium"     "Bismuth"       83      9,80    6       "15"
"Bk"    "Berkelium"                     97      13,25   7       "Ac"
"Br"    "Brom"          "Bromine"       35      3,14    4       "17"
"C"     "Carbonium"                     6       3,51    2       "14"
"Ca"    "Calcium"                       20      1,54    4       "2"
"Cd"    "Cadmium"                       48      8,64    5       "12"
"Ce"    "Cerium"                        58      6,77    6       "La"
"Cf"    "Californium"                   98      15,10   7       "Ac"
"Cl"    "Clorium"       "Chlorine"      17      2,95    3       "17"
"Cm"    "Curium"                        96      13,51   7       "Ac"
"Co"    "Cobaltium"     "Cobalt"        27      8,89    4       "9"
"Cr"    "Chromium"                      24      7,14    4       "6"
"Cs"    "Cäsium"                        55      1,90    6       "1"
"Cu"    "Cuprum"        "Copper"        29      8,92    4       "11"
"Db"    "Dubnium"                       105             7       "5"
```

- 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.

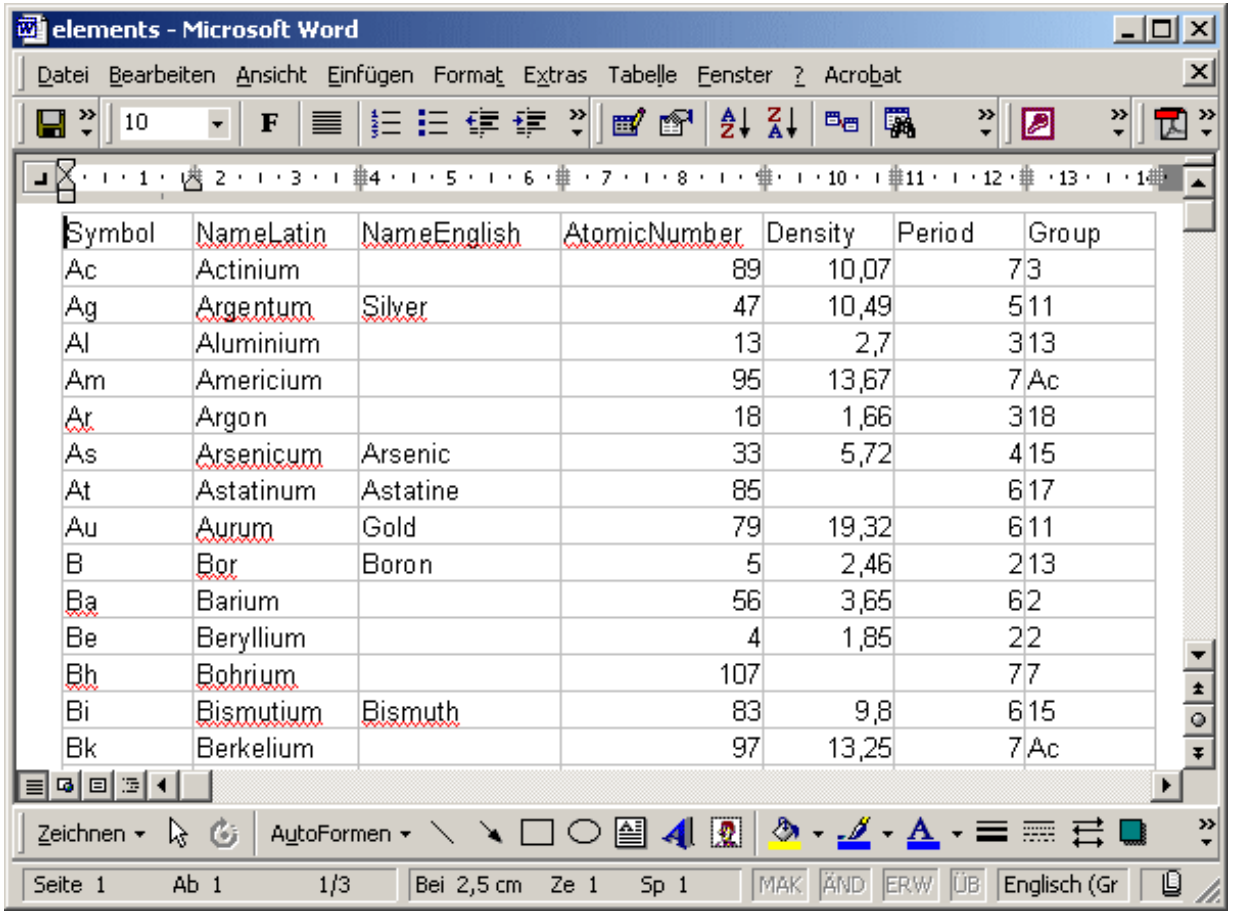


- MS Office offers a much more powerful text editor, called **MS Word**, supporting e.g. a „true“ tabular format for data (visualizing rows and columns automatically). Text files managed by the Word software are identifiable by their extension **.doc**.

- 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.

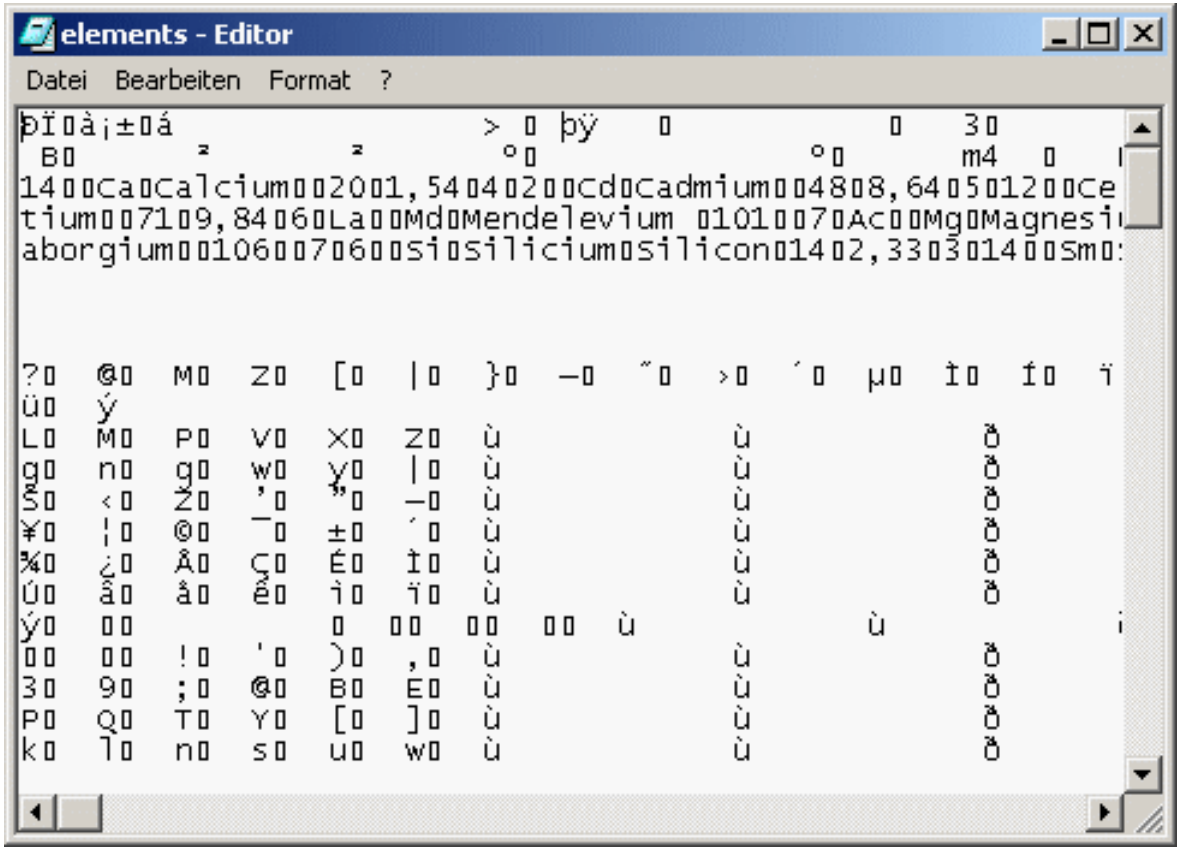


Symbol	NameLatin	NameEnglish	AtomicNumber	Density	Period	Group
Ac	Actinium		89	10,07	7	3
Ag	Argentum	Silver	47	10,49	5	11
Al	Aluminium		13	2,7	3	13
Am	Americium		95	13,67	7	Ac
Ar	Argon		18	1,66	3	18
As	Arsenicum	Arsenic	33	5,72	4	15
At	Astatinum	Astatine	85		6	17
Au	Aurum	Gold	79	19,32	6	11
B	Bor	Boron	5	2,46	2	13
Ba	Barium		56	3,65	6	2
Be	Beryllium		4	1,85	2	2
Bh	Bohrium		107		7	7
Bi	Bismutium	Bismuth	83	9,8	6	15
Bk	Berkelium		97	13,25	7	Ac

- **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.

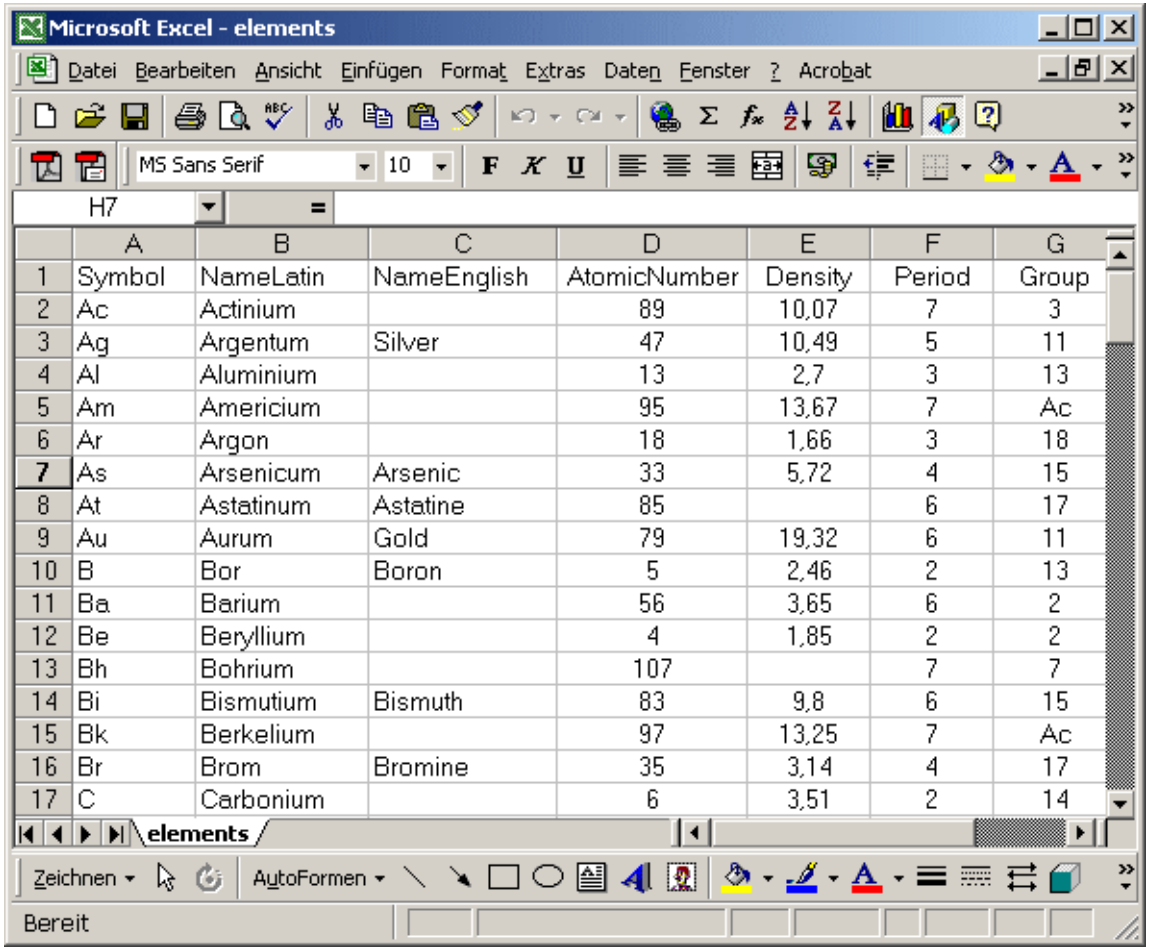


- Contained in each Office package on Microsoft PCs, there is an even more powerful tool for managing data, a so-called spreadsheet program called **MS Excel** supporting files with extension **.xls**.

- 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.

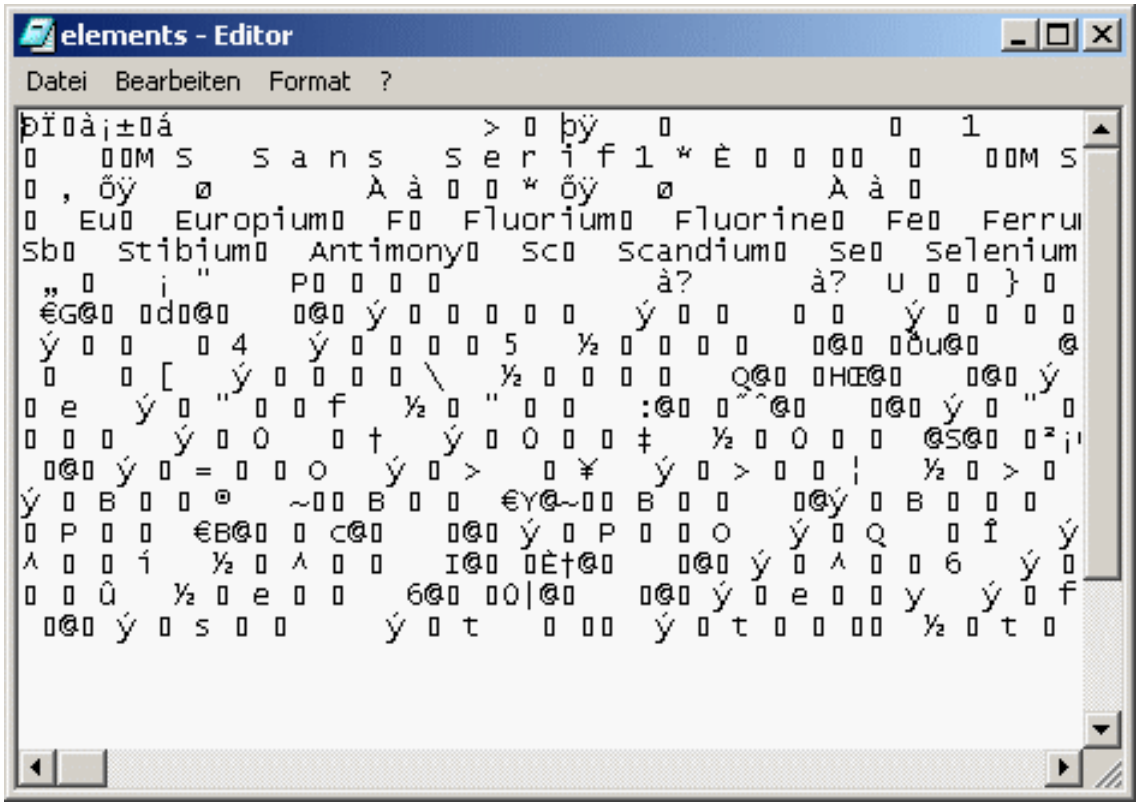
- Excel, however, is specialized in **statistical evaluations** of numerical data.



	A	B	C	D	E	F	G
1	Symbol	NameLatin	NameEnglish	AtomicNumber	Density	Period	Group
2	Ac	Actinium		89	10,07	7	3
3	Ag	Argentum	Silver	47	10,49	5	11
4	Al	Aluminium		13	2,7	3	13
5	Am	Americium		95	13,67	7	Ac
6	Ar	Argon		18	1,66	3	18
7	As	Arsenicum	Arsenic	33	5,72	4	15
8	At	Astatinum	Astatine	85		6	17
9	Au	Aurum	Gold	79	19,32	6	11
10	B	Bor	Boron	5	2,46	2	13
11	Ba	Barium		56	3,65	6	2
12	Be	Beryllium		4	1,85	2	2
13	Bh	Bohrium		107		7	7
14	Bi	Bismutium	Bismuth	83	9,8	6	15
15	Bk	Berkelium		97	13,25	7	Ac
16	Br	Brom	Bromine	35	3,14	4	17
17	C	Carbonium		6	3,51	2	14

- 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.



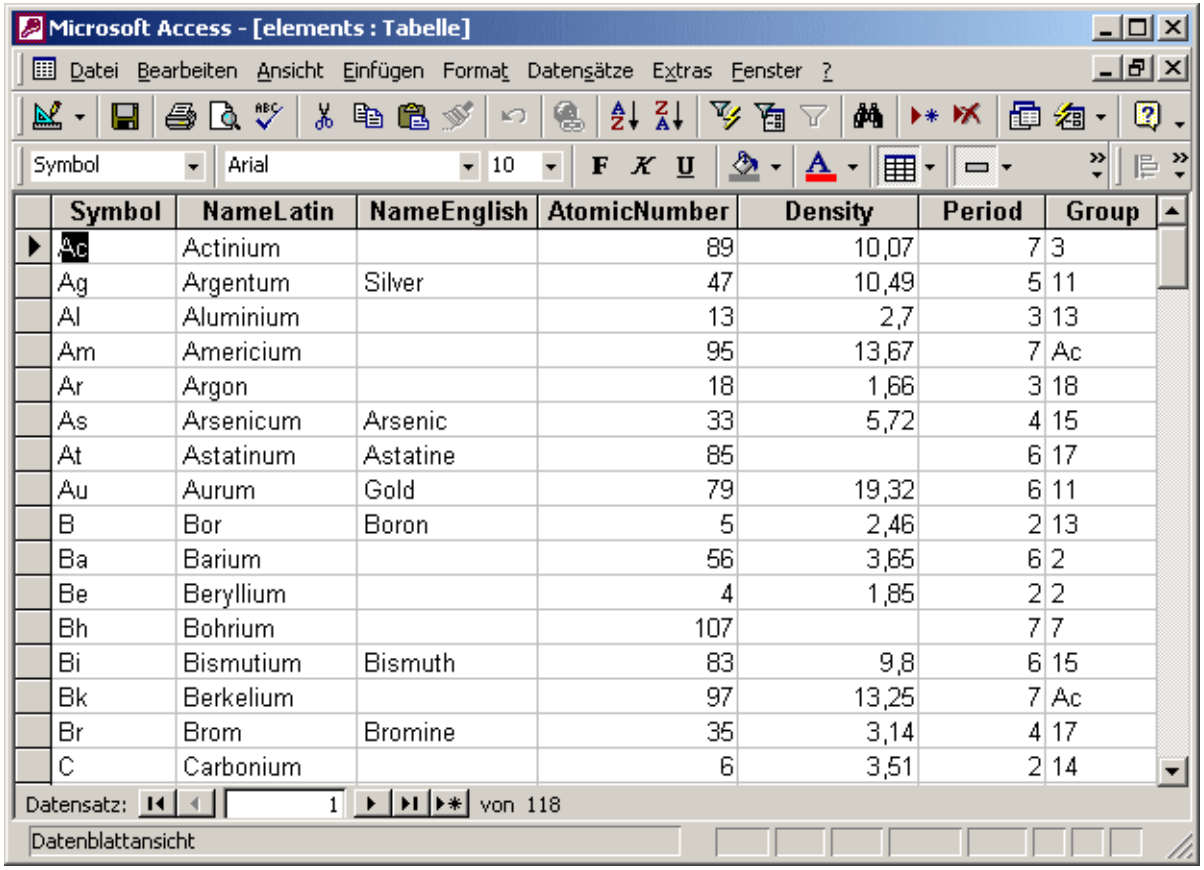
- What matters is the special software managing these „enhanced text files“, in this case the Excel system.

- 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**.

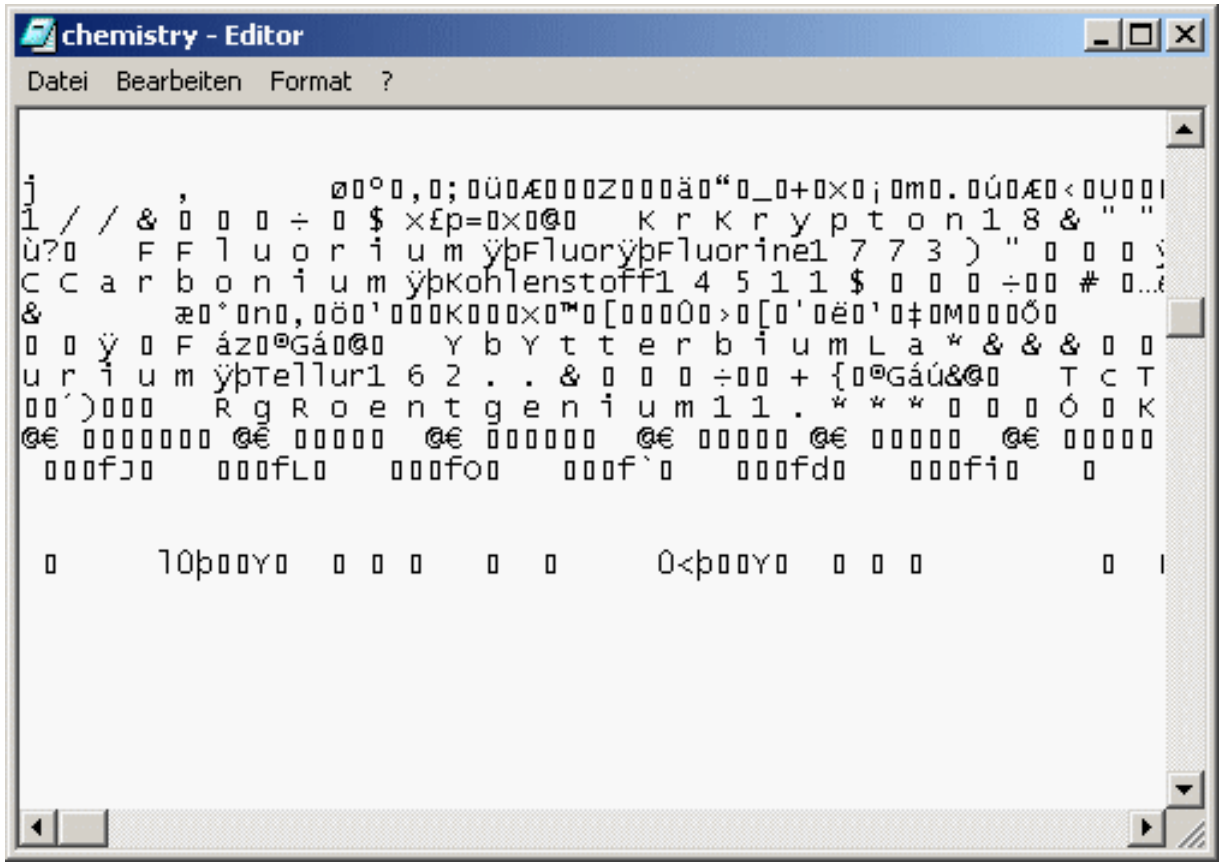


Symbol	NameLatin	NameEnglish	AtomicNumber	Density	Period	Group
Ac	Actinium		89	10,07	7	3
Ag	Argentum	Silver	47	10,49	5	11
Al	Aluminium		13	2,7	3	13
Am	Americium		95	13,67	7	Ac
Ar	Argon		18	1,66	3	18
As	Arsenicum	Arsenic	33	5,72	4	15
At	Astatinum	Astatine	85		6	17
Au	Aurum	Gold	79	19,32	6	11
B	Bor	Boron	5	2,46	2	13
Ba	Barium		56	3,65	6	2
Be	Beryllium		4	1,85	2	2
Bh	Bohrium		107		7	7
Bi	Bismutium	Bismuth	83	9,8	6	15
Bk	Berkelium		97	13,25	7	Ac
Br	Brom	Bromine	35	3,14	4	17
C	Carbonium		6	3,51	2	14

- Opening an mdb.file (alias an Access database) with a text editor reveals the „true nature“ of the representation again: Heavyly 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.

Throughout the course, we will use a small, but handy DBMS available on most PCs.

- Access is a DBMS for relational databases (data organized in form of tables), developed and distributed since 1992 by Microsoft.
- "Access-Homepage" at Microsoft:
  http://www.microsoft.com/office/access/default.asp
- 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
  - Maggie Strapland‘s Access pages at University of Bristol, UK
    http://www.bris.ac.uk/is/services/software/packages/access/
  - Jakob Lindenmeyer‘s Access tutorial at ETH Zürich, Schweiz
    http://www.inf.ethz.ch/personal/lindenme/publications/access/AccessTutorial.html
- 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.

- 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:
  http://www.cia.gov/cia/publications/factbook/index.html

- 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!

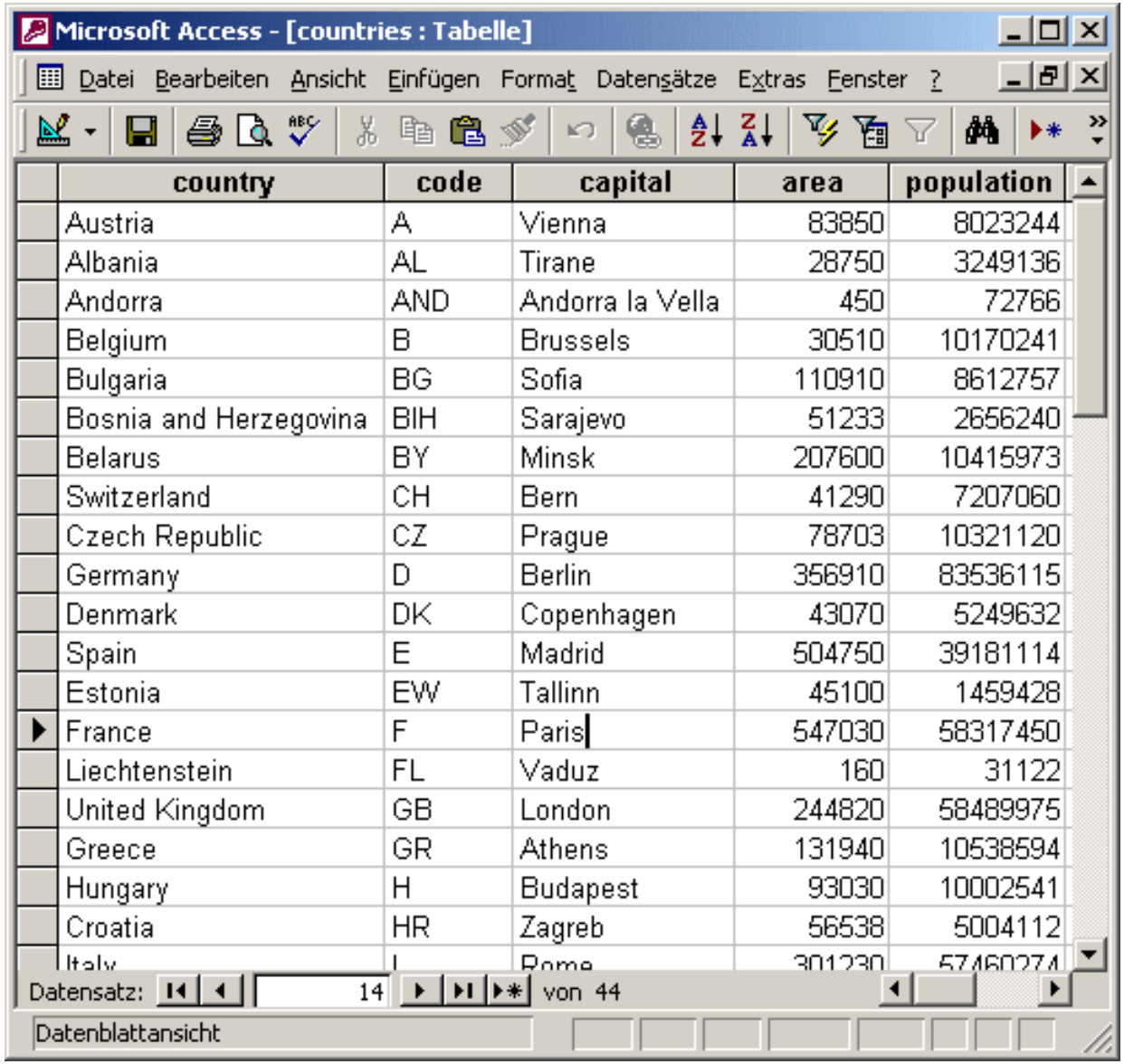
- At this moment in our lecture, the geo database serves as a first „appetizer“ to (relational) database management.

Europe.mdb is a small database for introductory purposes.

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

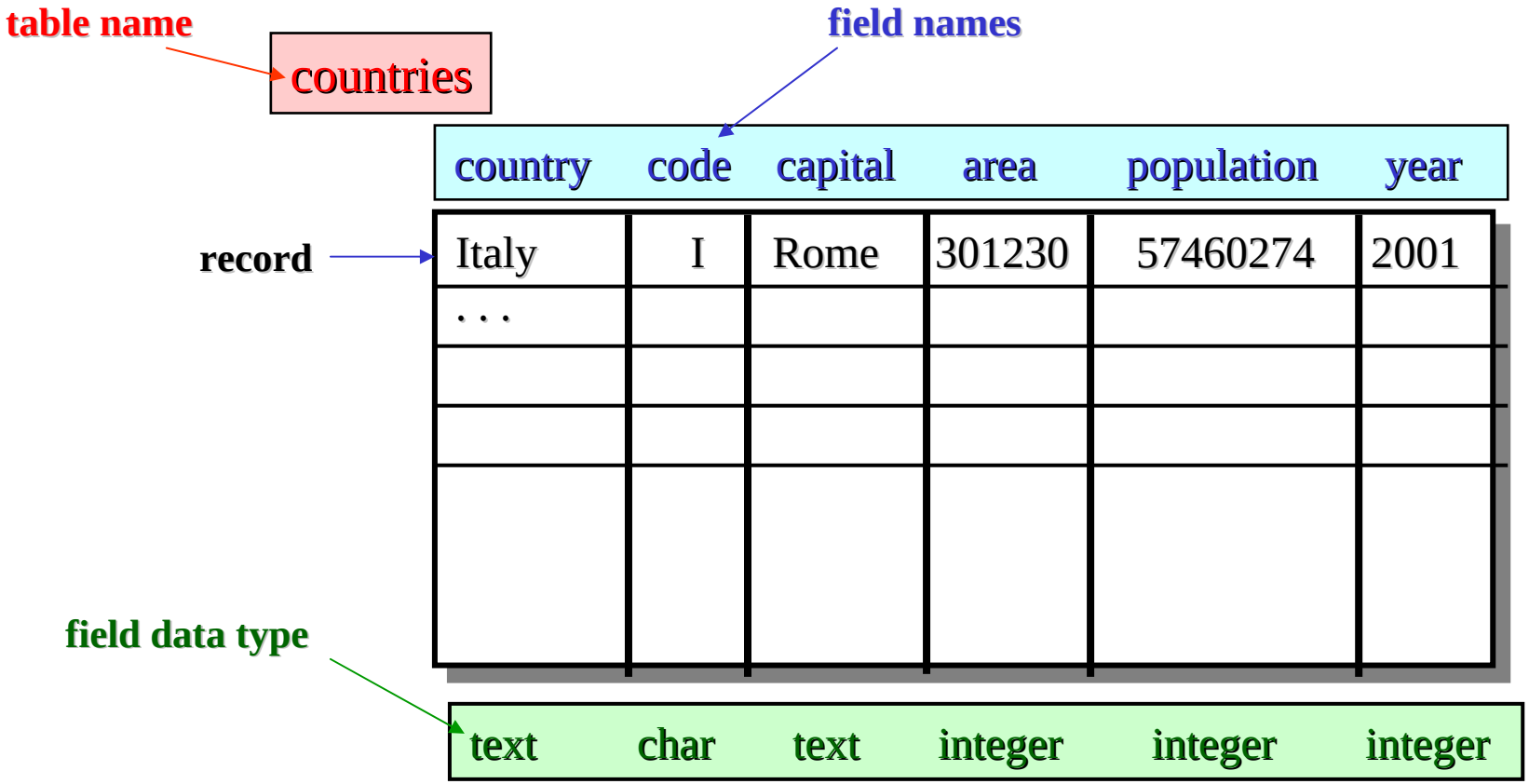
Today we will just learn the most basic ideas about relational databases – some of you will be already familiar with this.

And we will discuss the important question why a simple text file is not sufficient for keeping data.

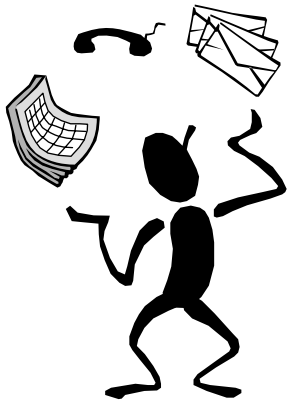


country	code	capital	area	population
Austria	A	Vienna	83850	8023244
Albania	AL	Tirane	28750	3249136
Andorra	AND	Andorra la Vella	450	72766
Belgium	B	Brussels	30510	10170241
Bulgaria	BG	Sofia	110910	8612757
Bosnia and Herzegovina	BIH	Sarajevo	51233	2656240
Belarus	BY	Minsk	207600	10415973
Switzerland	CH	Bern	41290	7207060
Czech Republic	CZ	Prague	78703	10321120
Germany	D	Berlin	356910	83536115
Denmark	DK	Copenhagen	43070	5249632
Spain	E	Madrid	504750	39181114
Estonia	EW	Tallinn	45100	1459428
France	F	Paris	547030	58317450
Liechtenstein	FL	Vaduz	160	31122
United Kingdom	GB	London	244820	58489975
Greece	GR	Athens	131940	10538594
Hungary	H	Budapest	93030	10002541
Croatia	HR	Zagreb	56538	5004112
Italy	I	Rome	301230	57460274

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



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:



theory	SQL	Access
relation	table	datasheet
tuple	row	record
attribute	column	field name
domain	data type	field data type

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 „countries":

Datasheet view

Microsoft Access - [countries : Tabelle]

Datei Bearbeiten Ansicht Einfügen Format Datensätze Extras Fenster ?

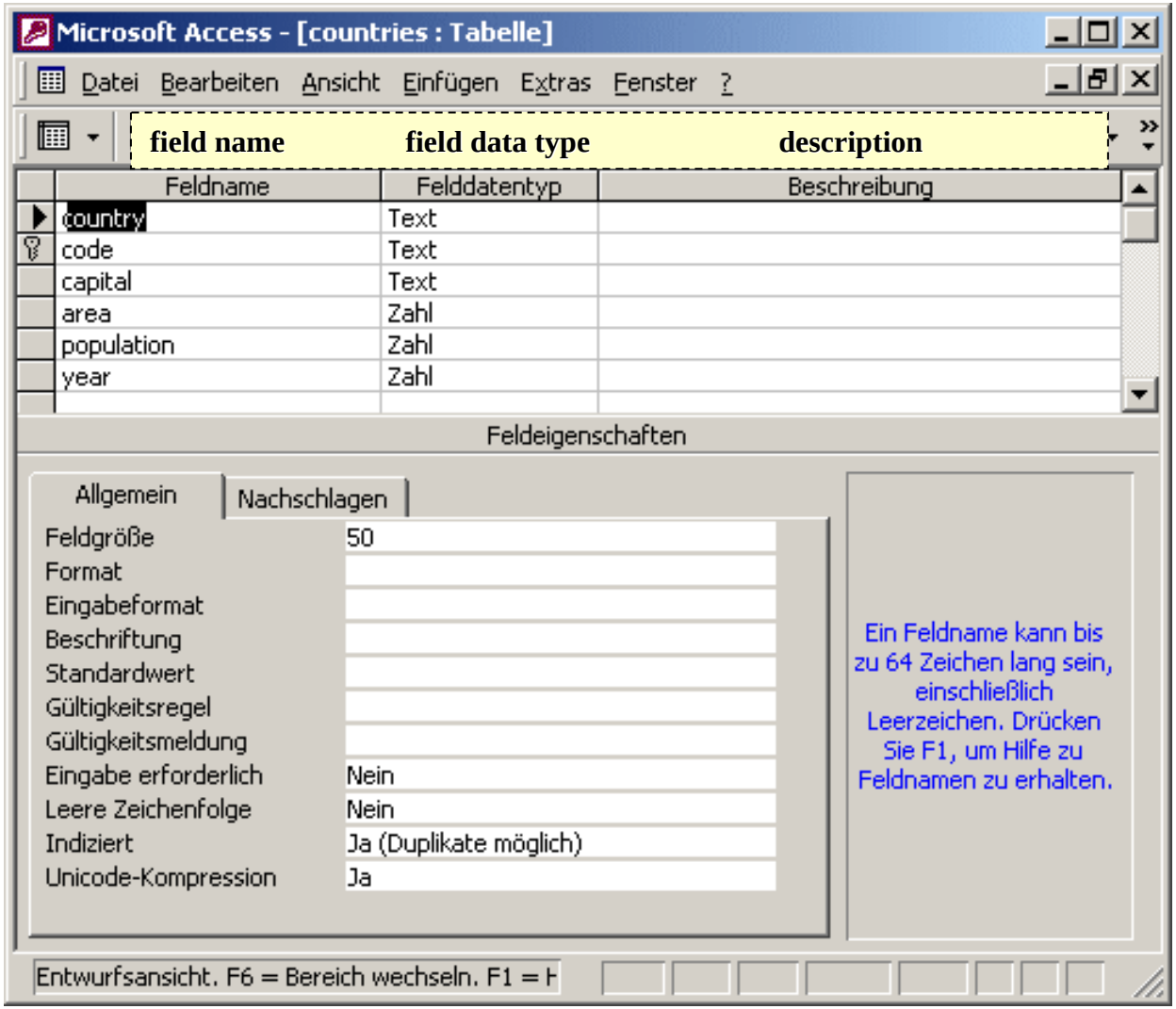
country	code	capital	area	population
Austria	A	Vienna	83850	8023244
Albania	AL	Tirane	28750	3249136
Andorra	AND	Andorra la Vella	450	72766
Belgium	B	Brussels	30510	10170241
Bulgaria	BG	Sofia	110910	8612757
Bosnia and Herzegovina	BIH	Sarajevo	51233	2656240
Belarus	BY	Minsk	207600	10415973
Switzerland	CH	Bern	41290	7207060
Czech Republic	CZ	Prague	78703	10321120
Germany	D	Berlin	356910	83536115
Denmark	DK	Copenhagen	43070	5249632
Spain	E	Madrid	504750	39181114
Estonia	EW	Tallinn	45100	1459428
France	F	Paris	547030	58317450
Liechtenstein	FL	Vaduz	160	31122
United Kingdom	GB	London	244820	58489975
Greece	GR	Athens	131940	10538594
Hungary	H	Budapest	93030	10002541
Croatia	HR	Zagreb	56538	5004112
Italy	I	Rome	301230	57460274

Datensatz: 14 von 44

Datenblattansicht

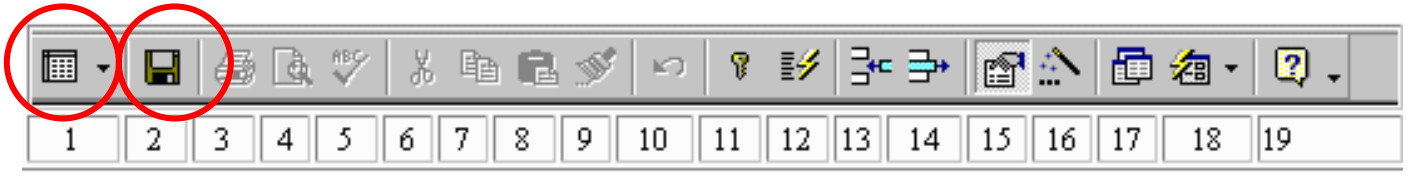
Access table „countries":

Design view



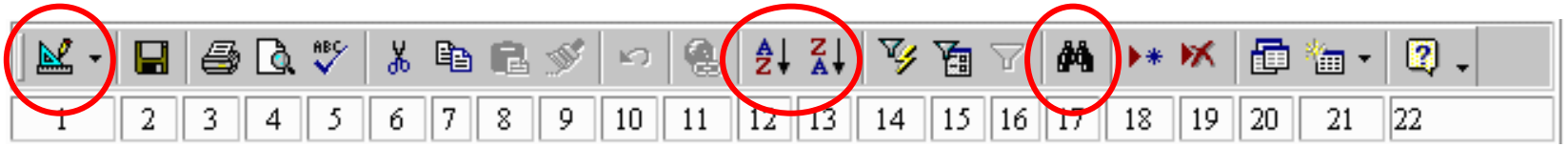
field size
format
input mask
caption
default value
validation rule
validation text
Required
Allow zero length
Indexed
Unicode compression

- In design view:



1 Switch to datasheet view
2 Save table design

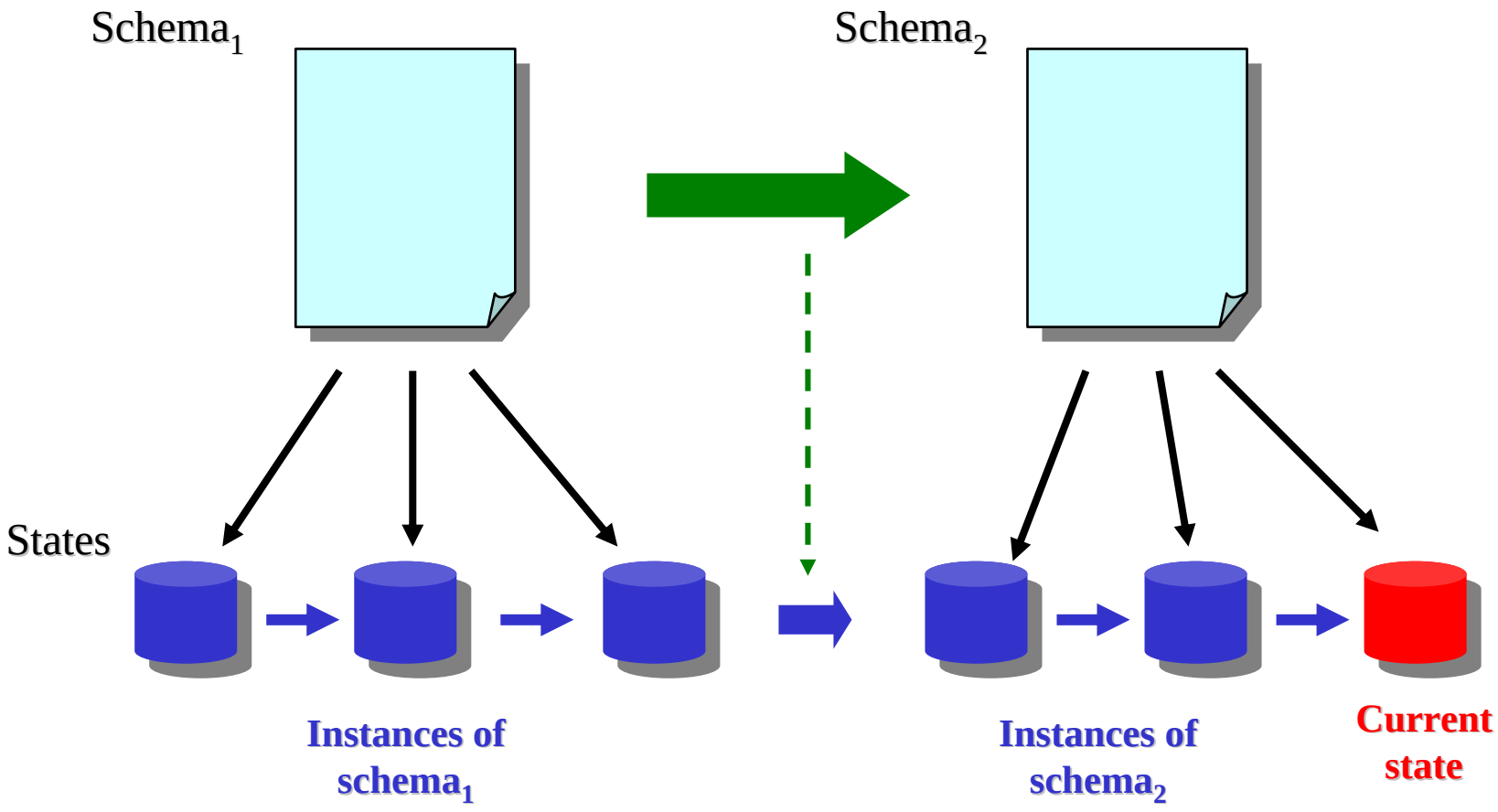
- In datasheet view:



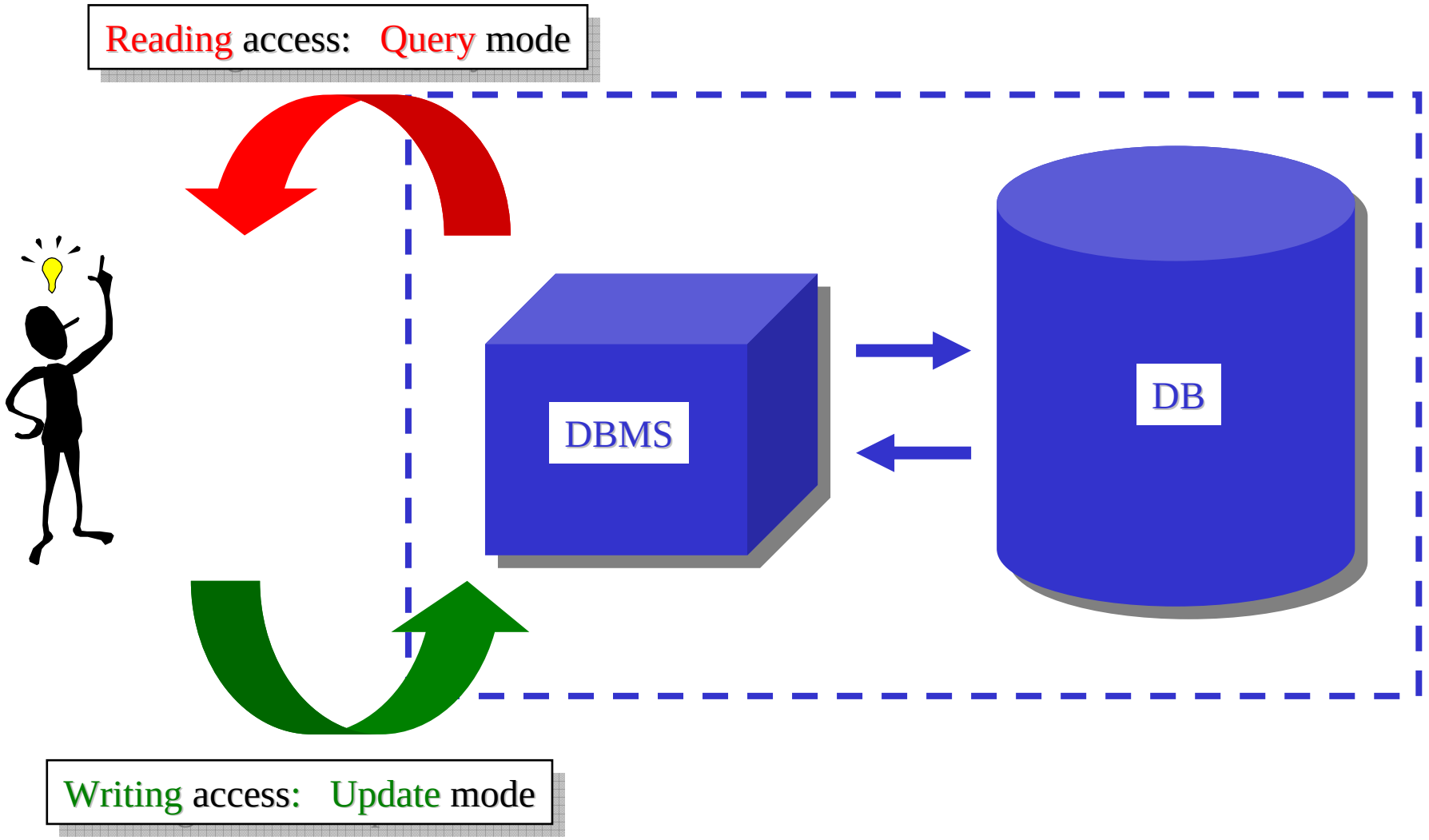
1 Switch to design view
12/13 Order data in ascending or descending order
17 Search via "pattern matching" in data sheet

- 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") !


Schema1
Schema2
States
Instances of schema1
Instances of schema2
Current state

There are two basic forms of interaction with a database:



How to „read" from a database ?

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 ist support of one or more query language.

- 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:

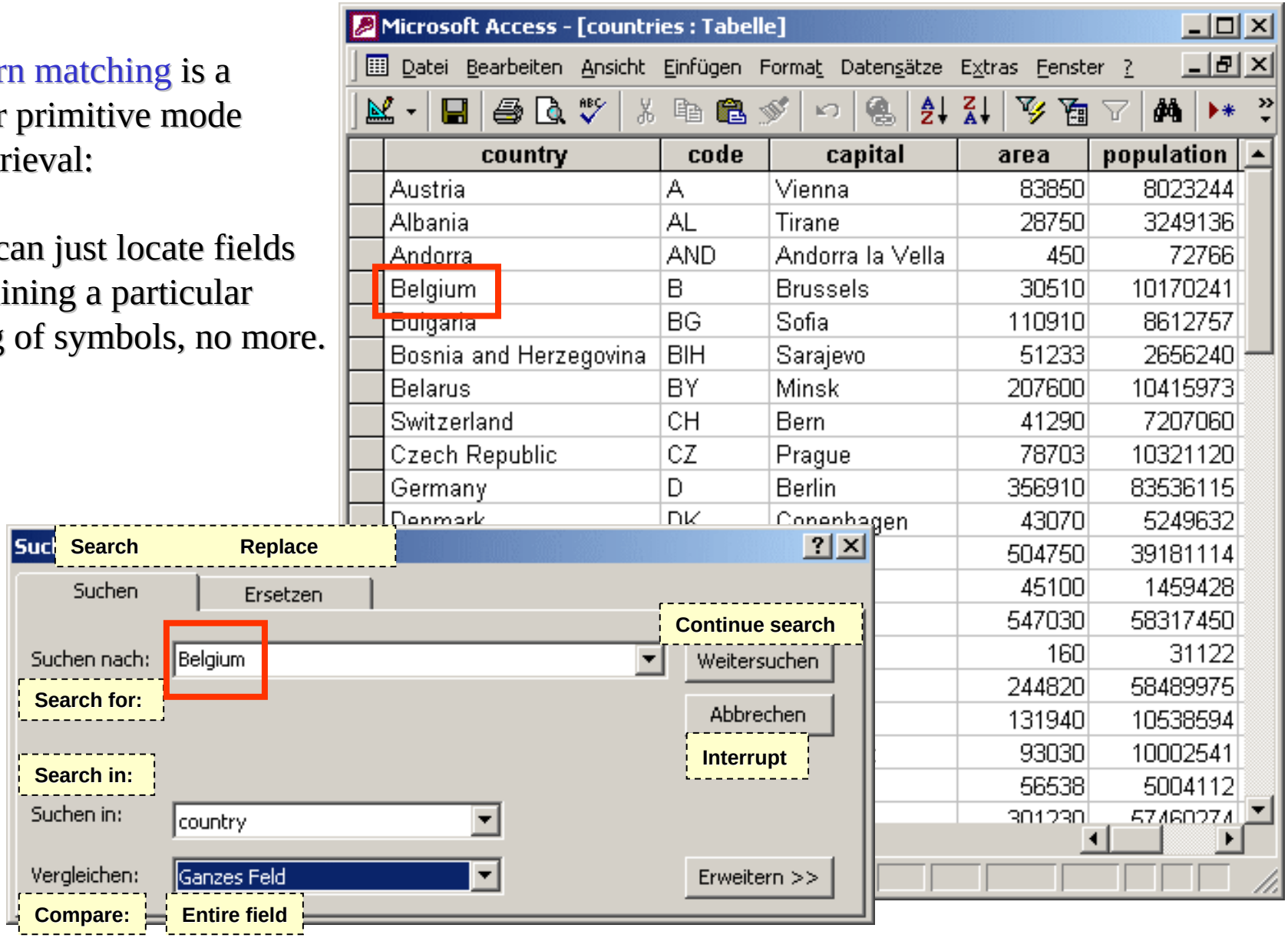
  - Graphically-interactive: "Query-by-Example" (QBE)
  - Textual: "Structured Query Language" (SQL)

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

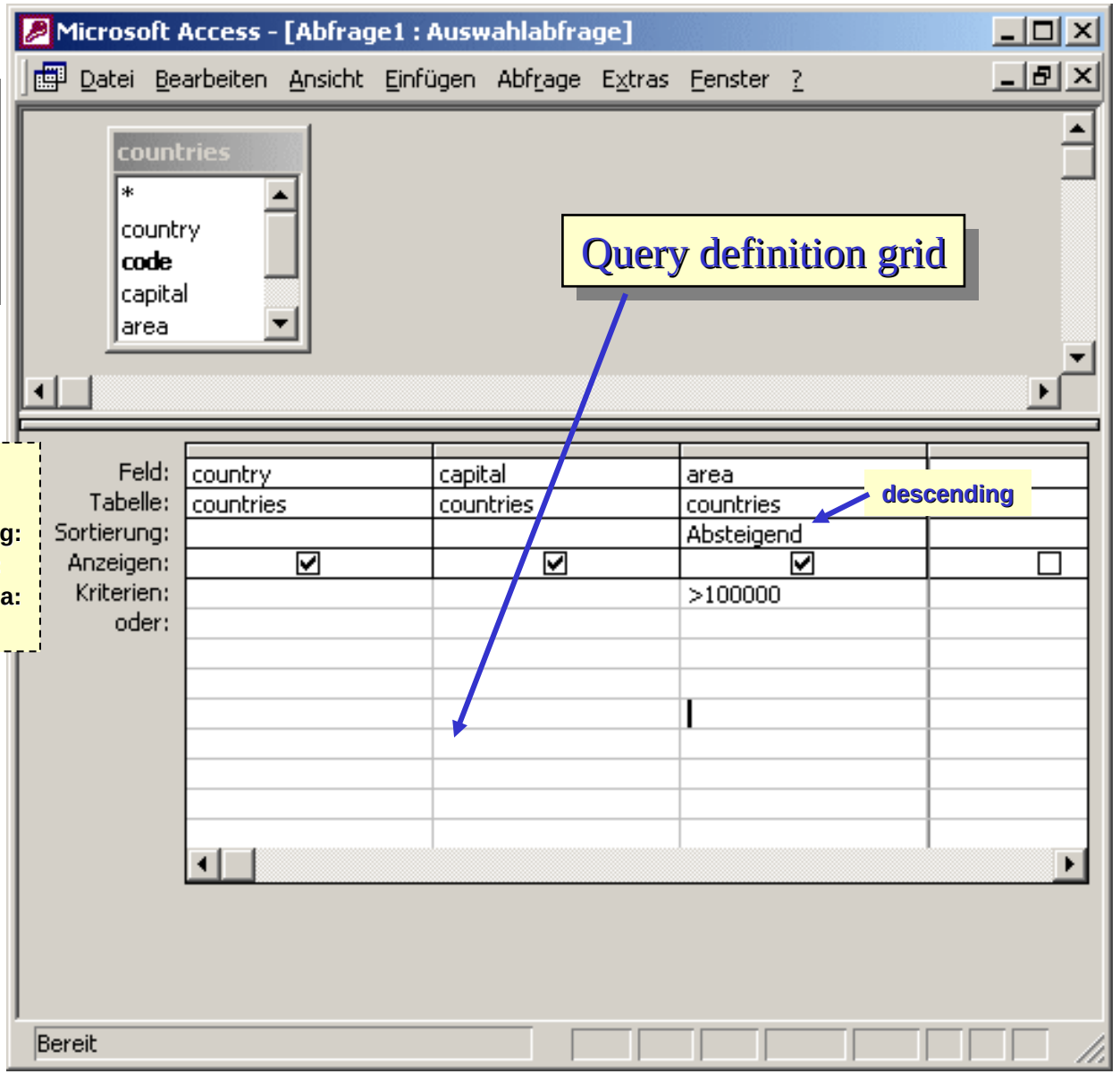
- SQL is standardized and is „understood“ by any commercial DBMS.

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.

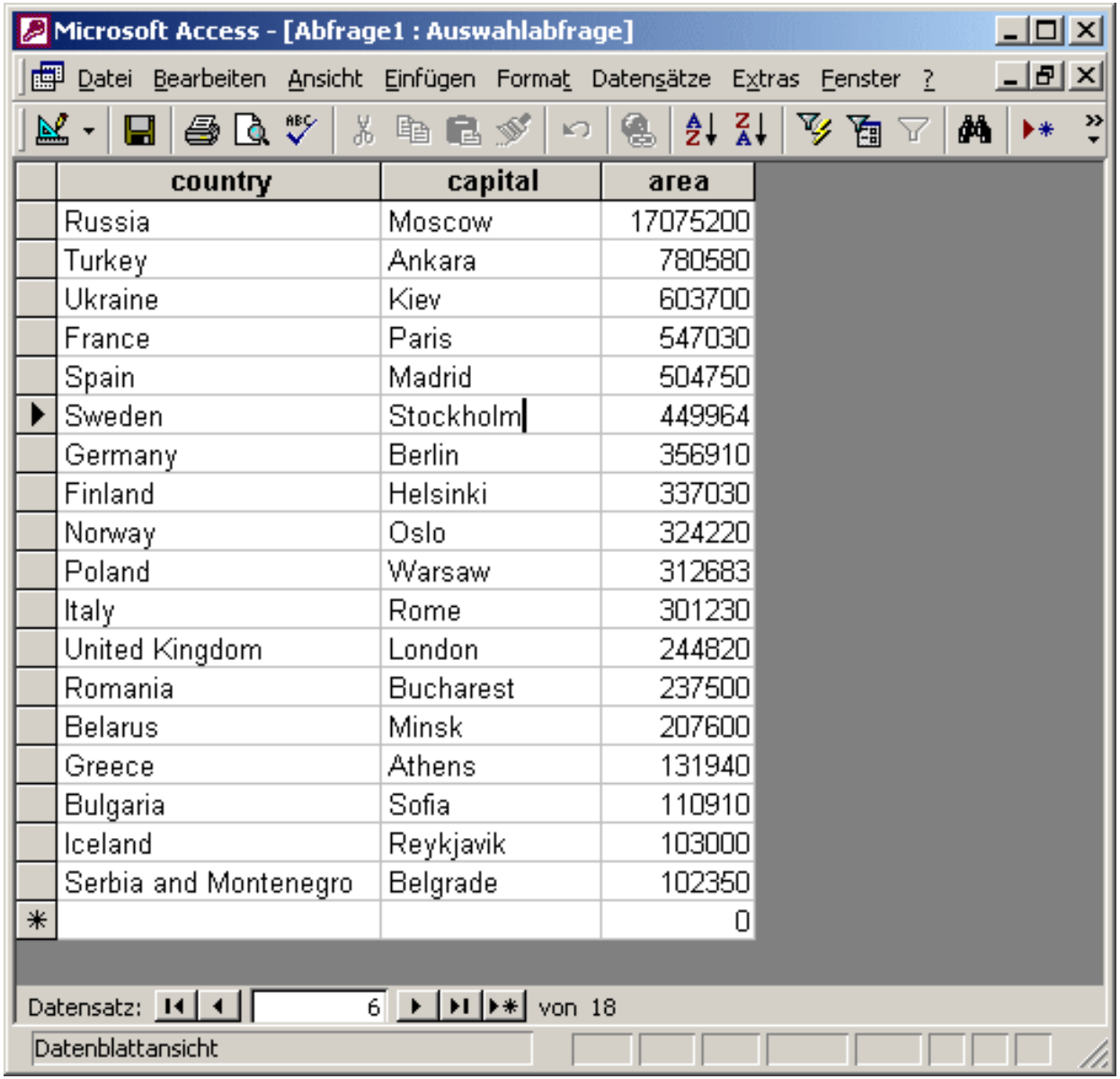
Retrieve name, capital and area of all countries larger than 100000 km^2 in descending order of size !

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

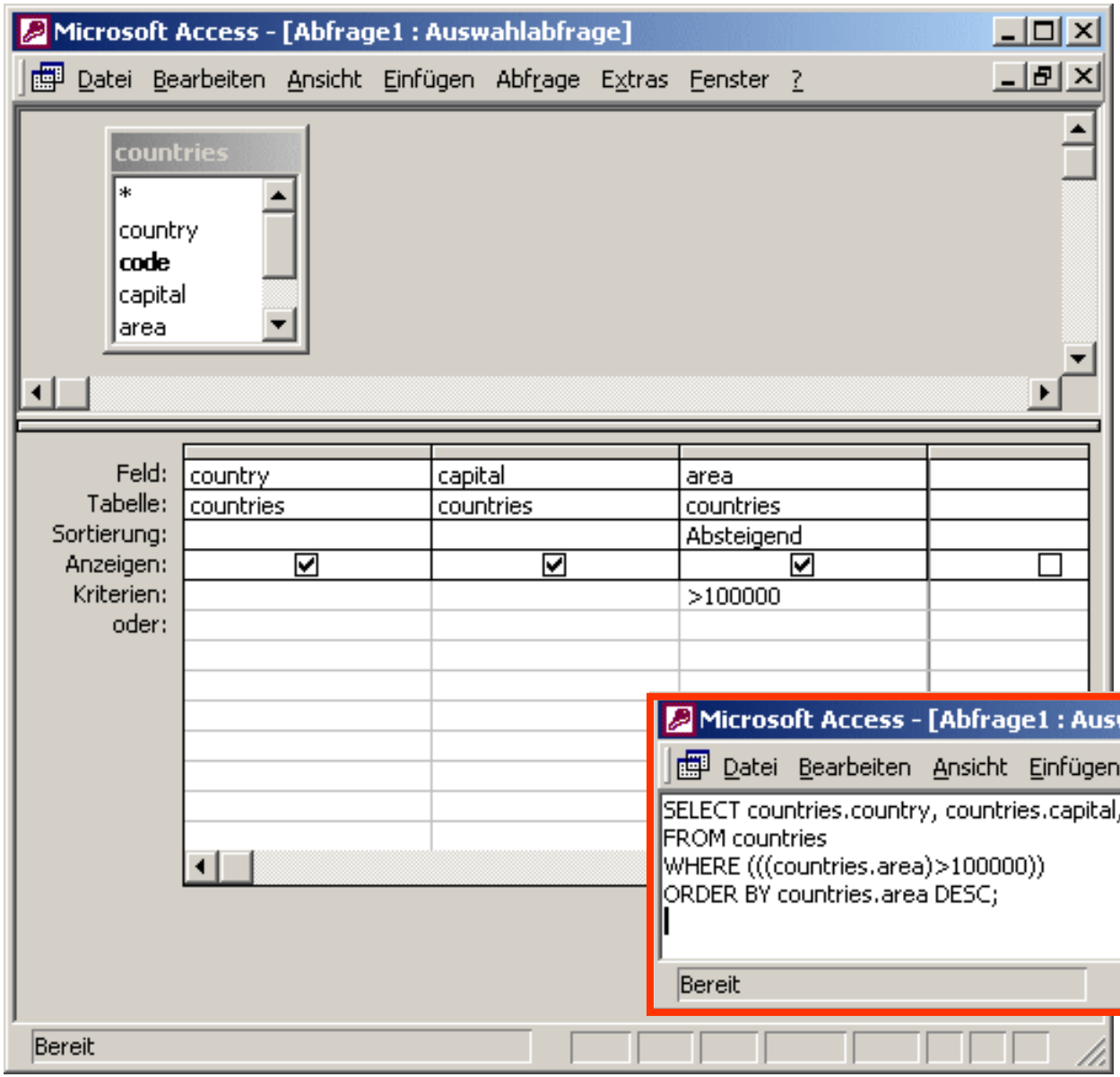
Thus, they may be „reused“ as input for further queries.

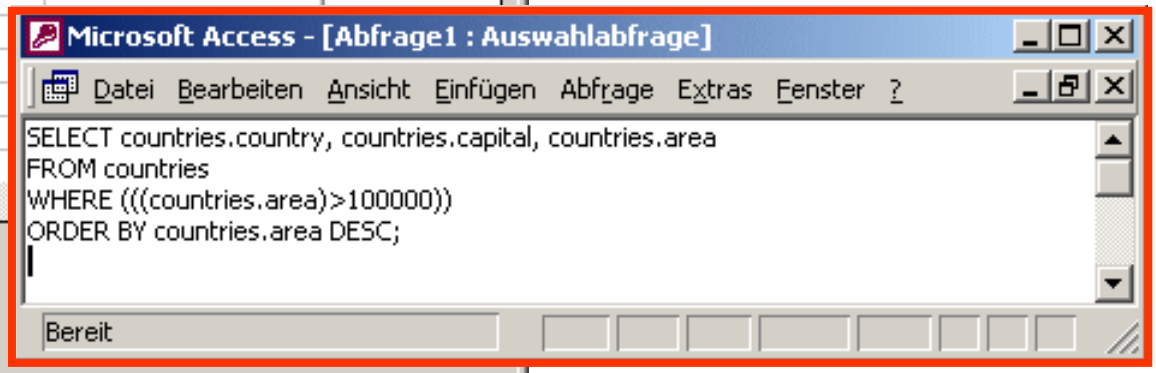
However, these tables are not stored in the DB !
They are „virtual“ tables recomputed each time the query is asked.

Datasheet view, too.



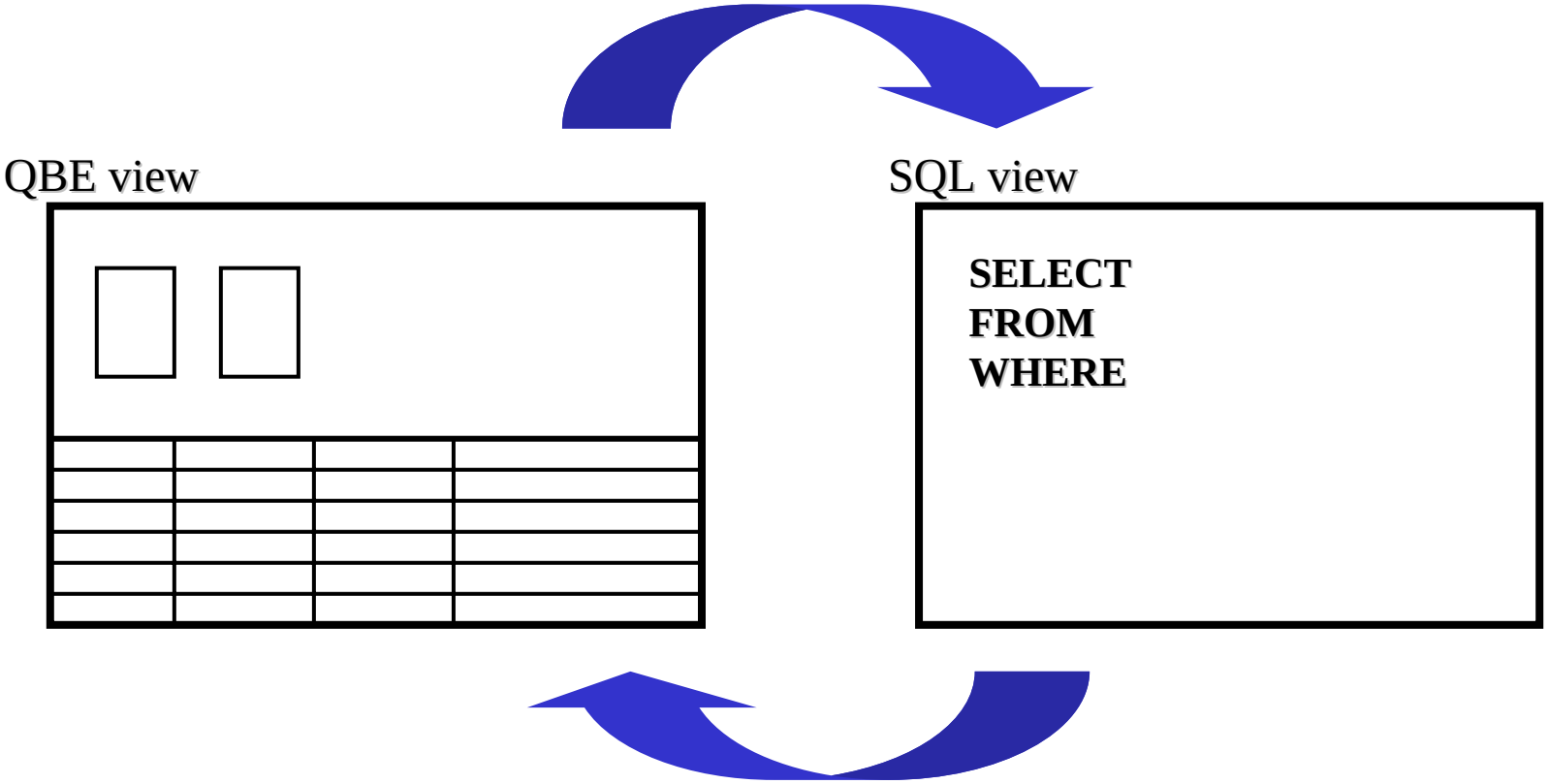
country	capital	area
Russia	Moscow	17075200
Turkey	Ankara	780580
Ukraine	Kiev	603700
France	Paris	547030
Spain	Madrid	504750
Sweden	Stockholm	449964
Germany	Berlin	356910
Finland	Helsinki	337030
Norway	Oslo	324220
Poland	Warsaw	312683
Italy	Rome	301230
United Kingdom	London	244820
Romania	Bucharest	237500
Belarus	Minsk	207600
Greece	Athens	131940
Bulgaria	Sofia	110910
Iceland	Reykjavik	103000
Serbia and Montenegro	Belgrade	102350
		0

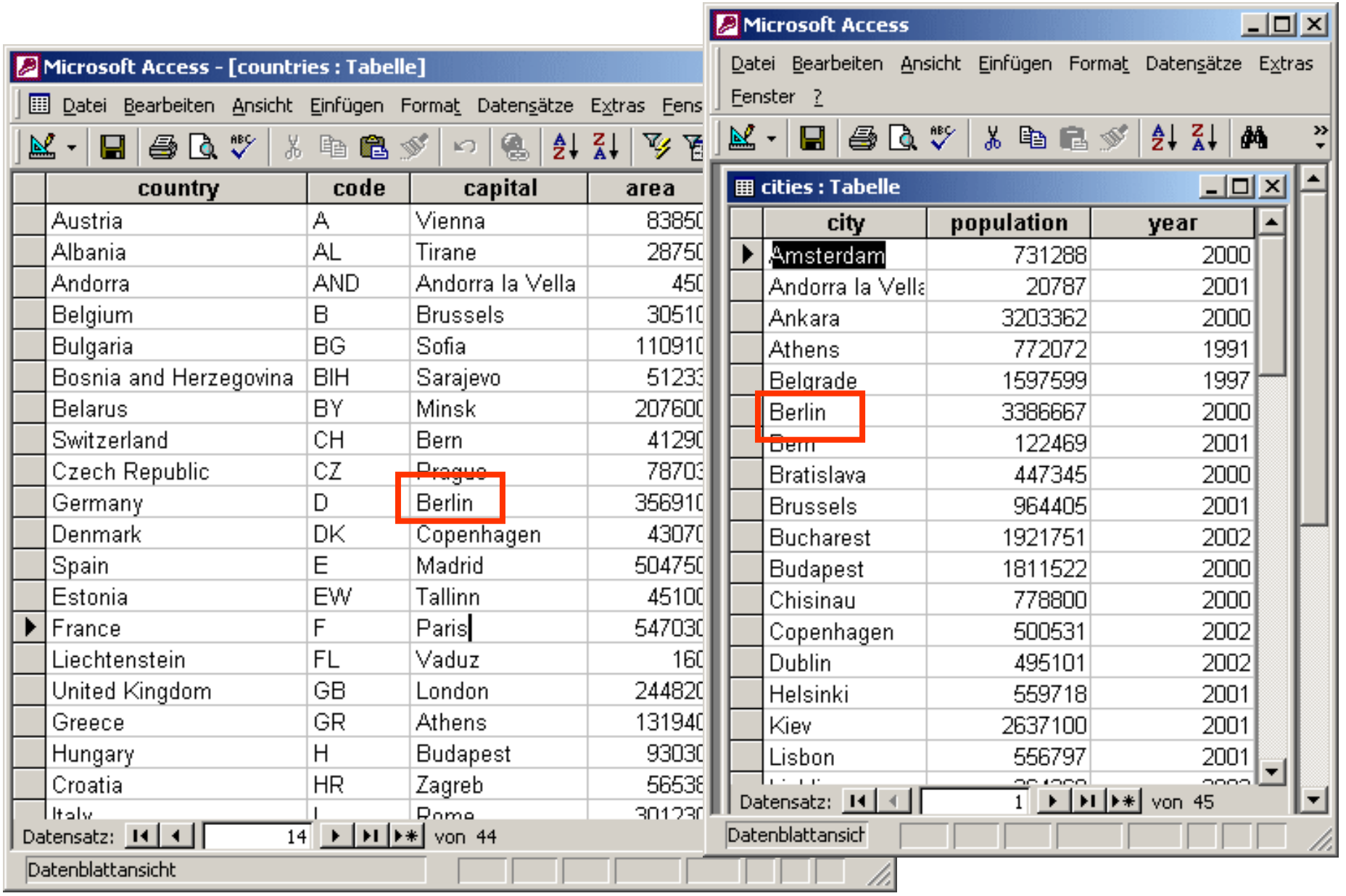


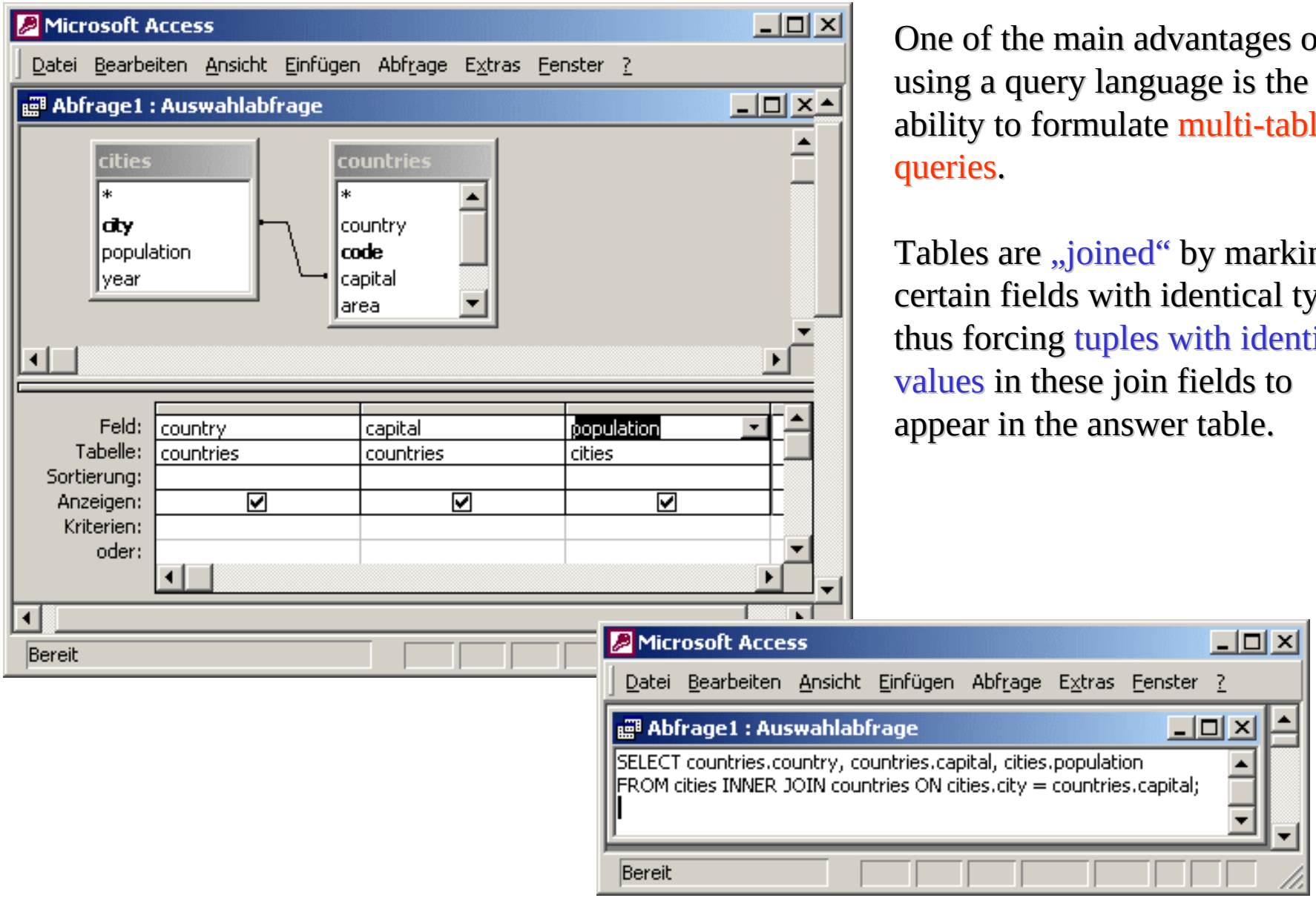


The same query expressed in textual style as an SQL table expression:

Changes in the query formulation in one view are automatically and immediately passed to the other view: Both representations are fully synchronized !

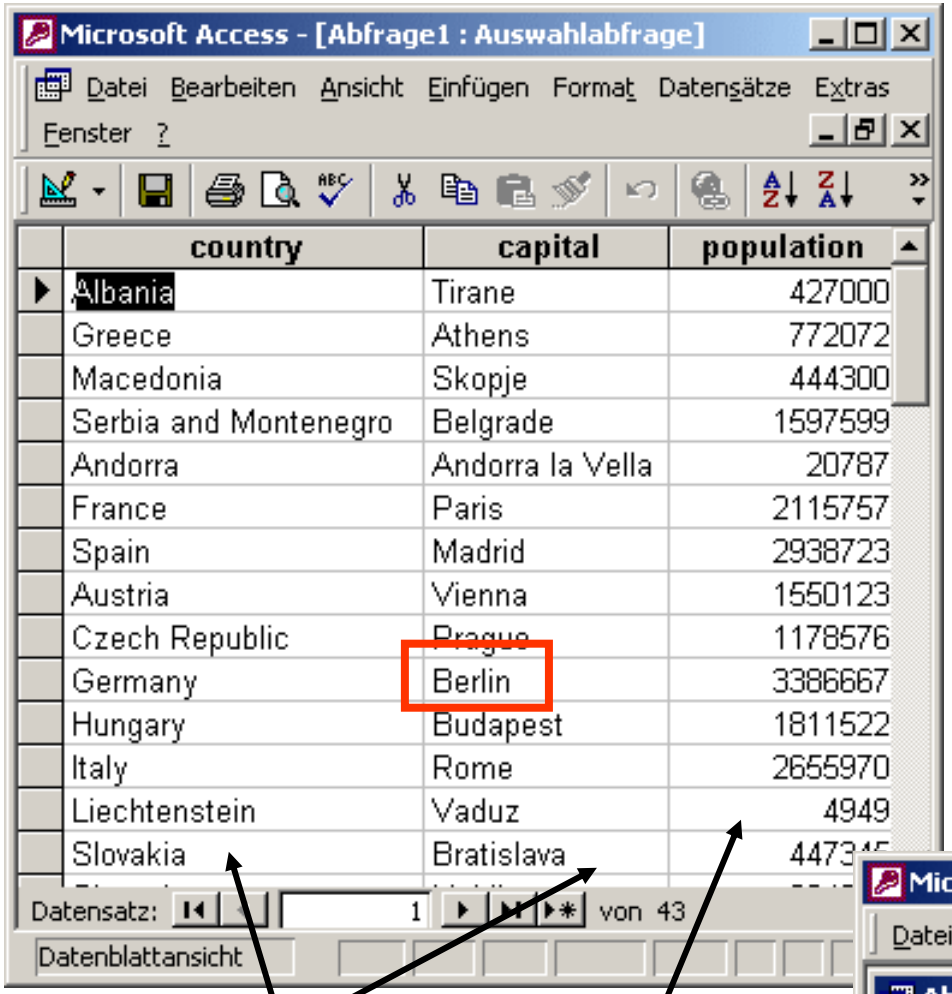



Microsoft Access - [countries : Tabelle]
Datei Bearbeiten Ansicht Einfügen Format Datensätze Extras Fens
country | code | capital | area
Austria | A | Vienna | 83850
Albania | AL | Tirane | 28750
Andorra | AND | Andorra la Vella | 450
Belgium | B | Brussels | 30510
Bulgaria | BG | Sofia | 110910
Bosnia and Herzegovina | BIH | Sarajevo | 51233
Belarus | BY | Minsk | 207600
Switzerland | CH | Bern | 41290
Czech Republic | CZ | Prague | 78703
Germany | D | Berlin | 356910
Denmark | DK | Copenhagen | 43070
Spain | E | Madrid | 504750
Estonia | EW | Tallinn | 45100
France | F | Paris | 547030
Liechtenstein | FL | Vaduz | 160
United Kingdom | GB | London | 244820
Greece | GR | Athens | 131940
Hungary | H | Budapest | 93030
Croatia | HR | Zagreb | 56538
Datensatz: 14 von 44
Datenblattansicht
Microsoft Access
Datei Bearbeiten Ansicht Einfügen Format Datensätze Extras
Fenster ?
cities : Tabelle
city | population | year
Amsterdam | 731288 | 2000
Andorra la Vella | 20787 | 2001
Ankara | 3203362 | 2000
Athens | 772072 | 1991
Belgrade | 1597599 | 1997
Berlin | 3386667 | 2000
Bern | 122469 | 2001
Bratislava | 447345 | 2000
Brussels | 964405 | 2001
Bucharest | 1921751 | 2002
Budapest | 1811522 | 2000
Chisinau | 778800 | 2000
Copenhagen | 500531 | 2002
Dublin | 495101 | 2002
Helsinki | 559718 | 2001
Kiev | 2637100 | 2001
Lisbon | 556797 | 2001
Datensatz: 1 von 45
Datenblattansicht



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.



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

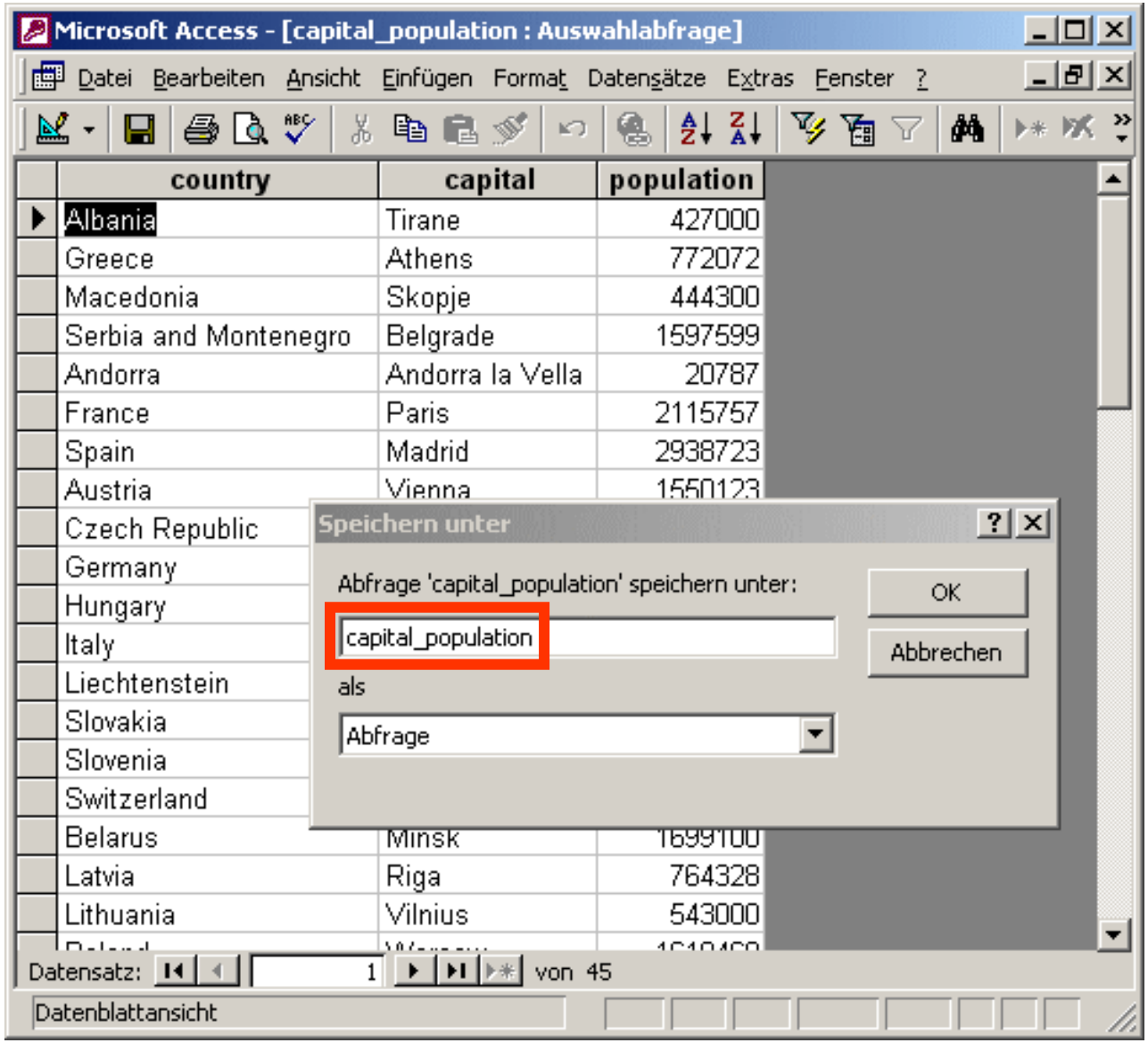
The answer table combines fields from both input tables and arranges them in a newly defined manner.

from countries

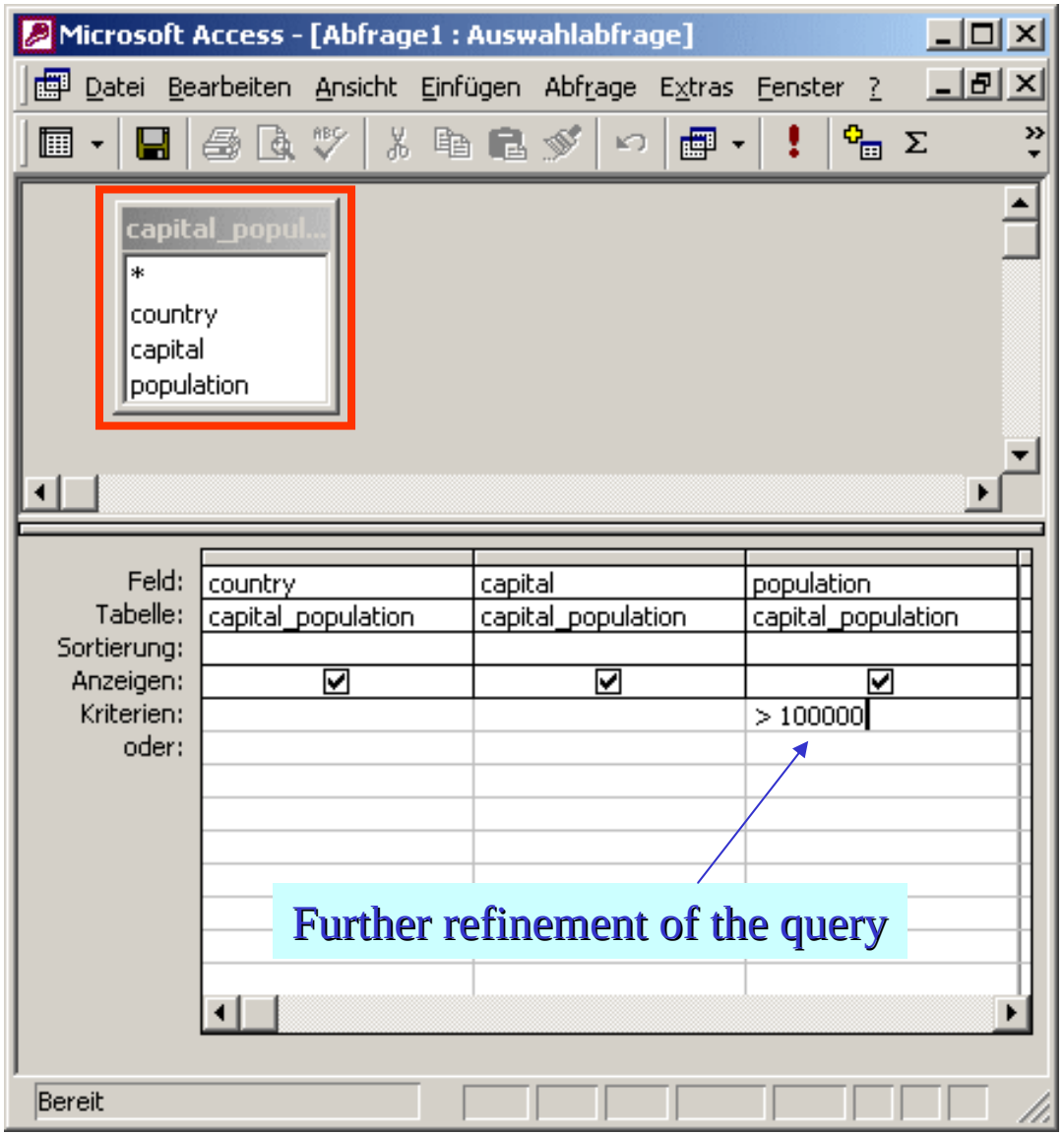
from cities

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!

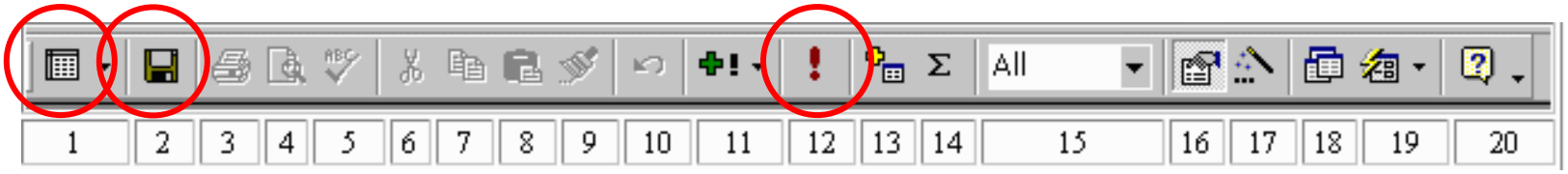


Stored queries may serve as input for subsequent queries in the same way as tables are.



Queries, not tables !!

- 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

- 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)

  Aggregation

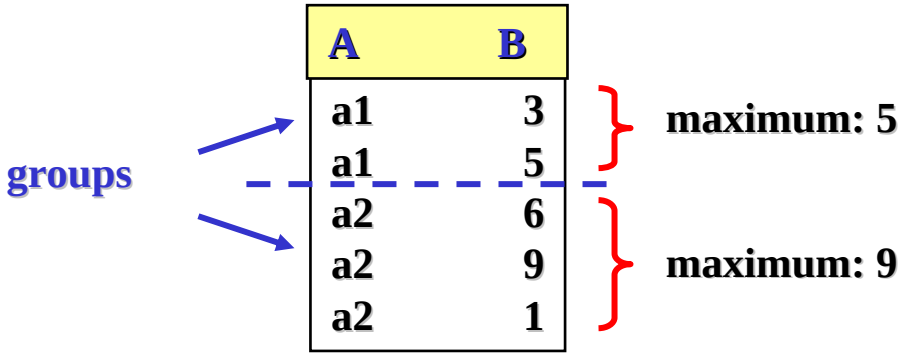
- Corresponding arithmetic functions: Aggregate functions

- In Access design mode for queries: By clicking on the function symbol Σ, 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:

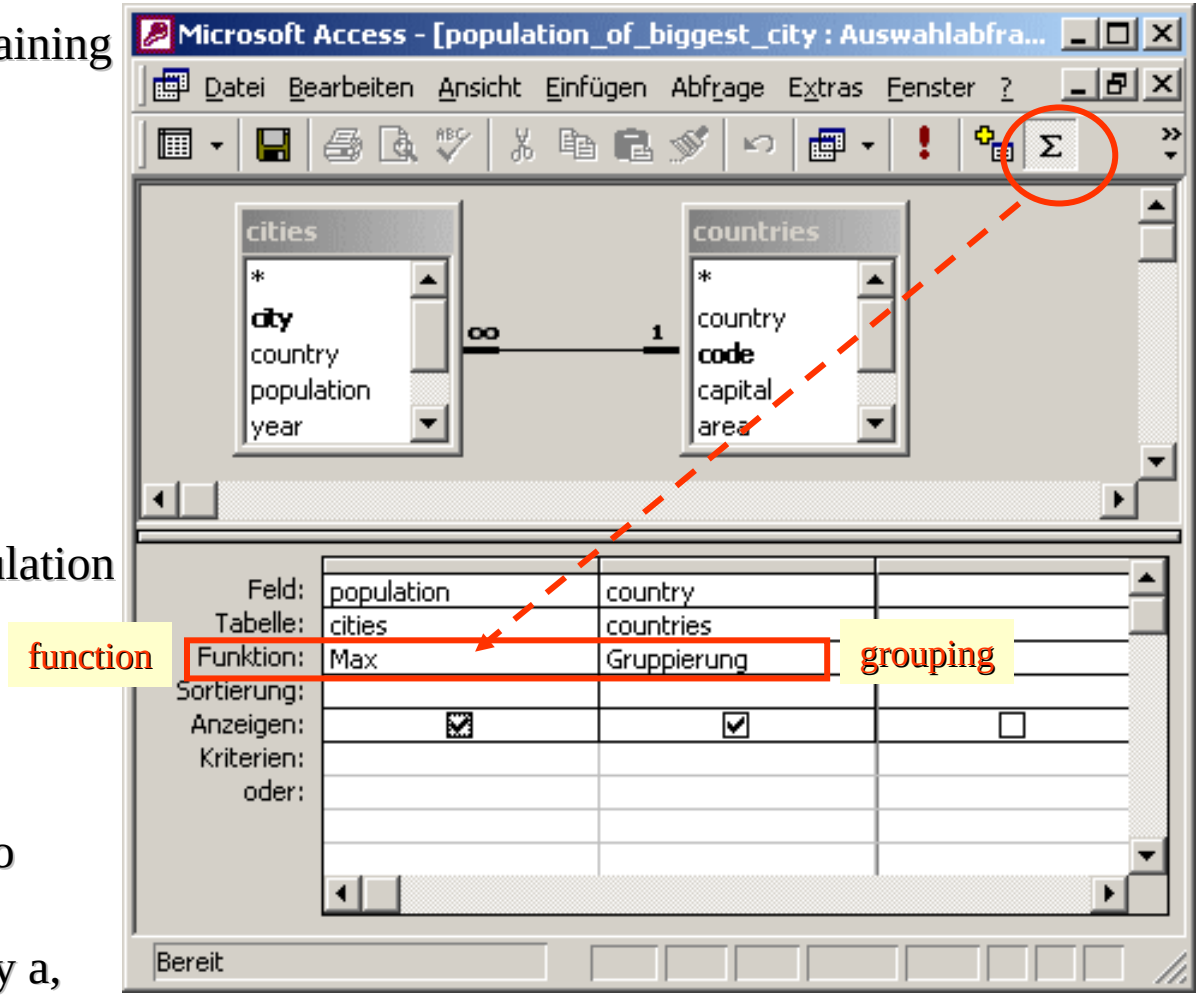
  Grouping

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



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:

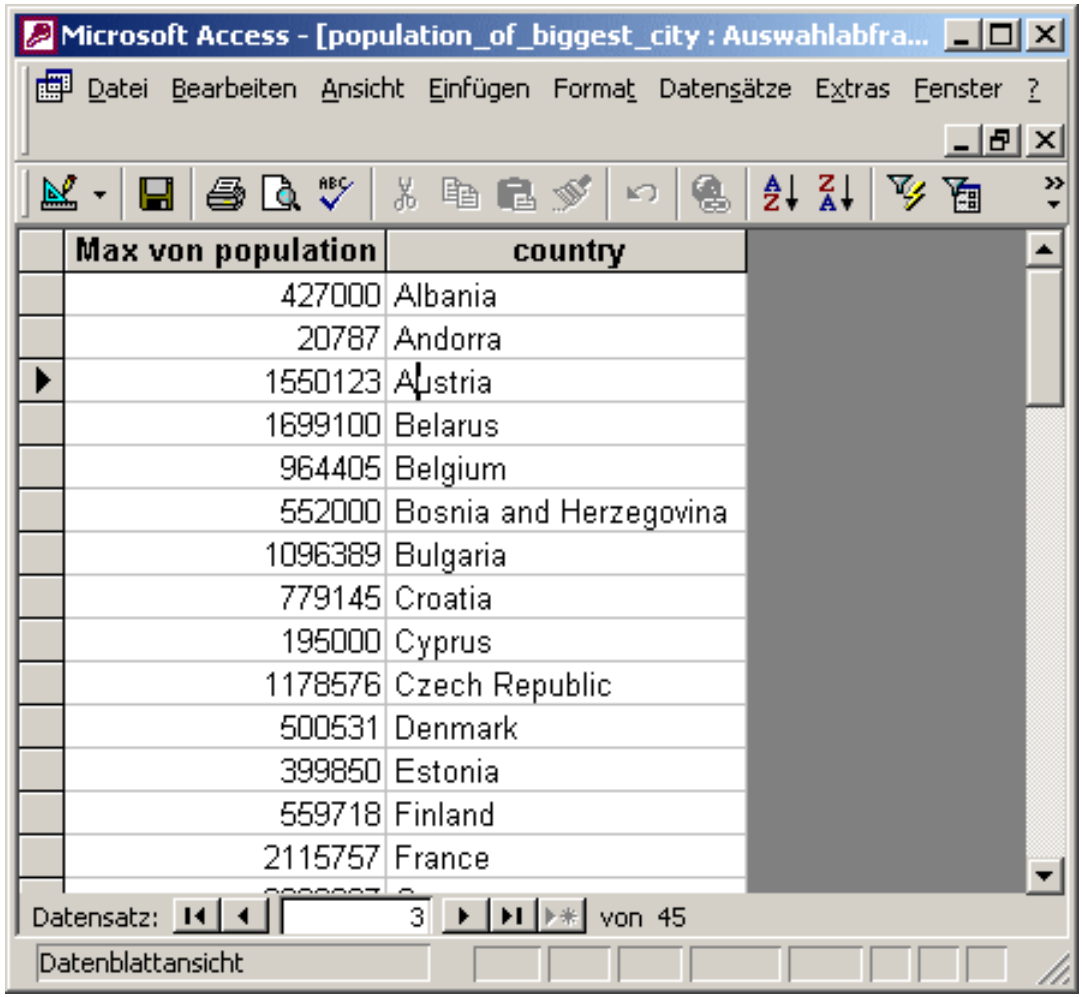
It is impossible in Access to include other fields not either grouped, restricted by a, condition or aggregated upon into a query containing aggregation !

(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)



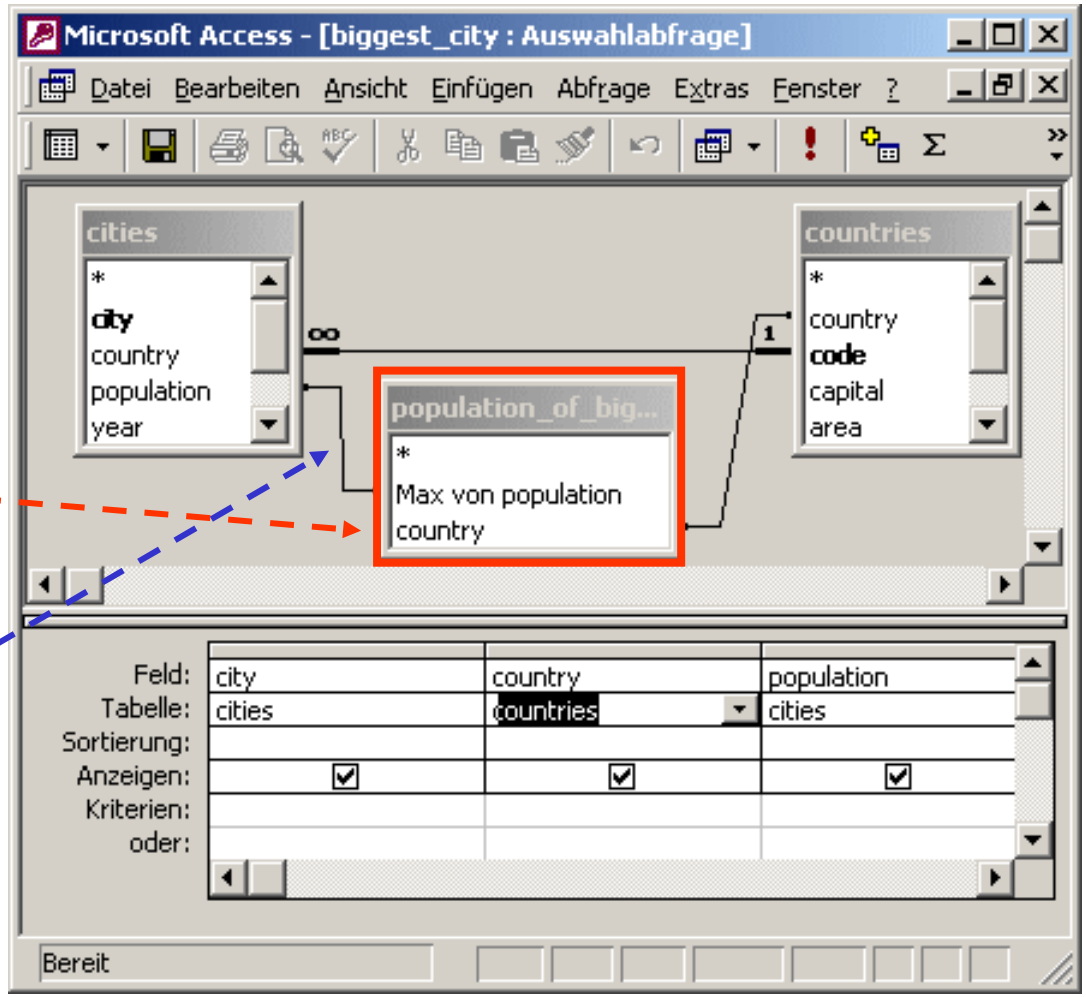
Max von population	country
427000	Albania
20787	Andorra
1550123	Austria
1699100	Belarus
964405	Belgium
552000	Bosnia and Herzegovina
1096389	Bulgaria
779145	Croatia
195000	Cyprus
1178576	Czech Republic
500531	Denmark
399850	Estonia
559718	Finland
2115757	France

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:

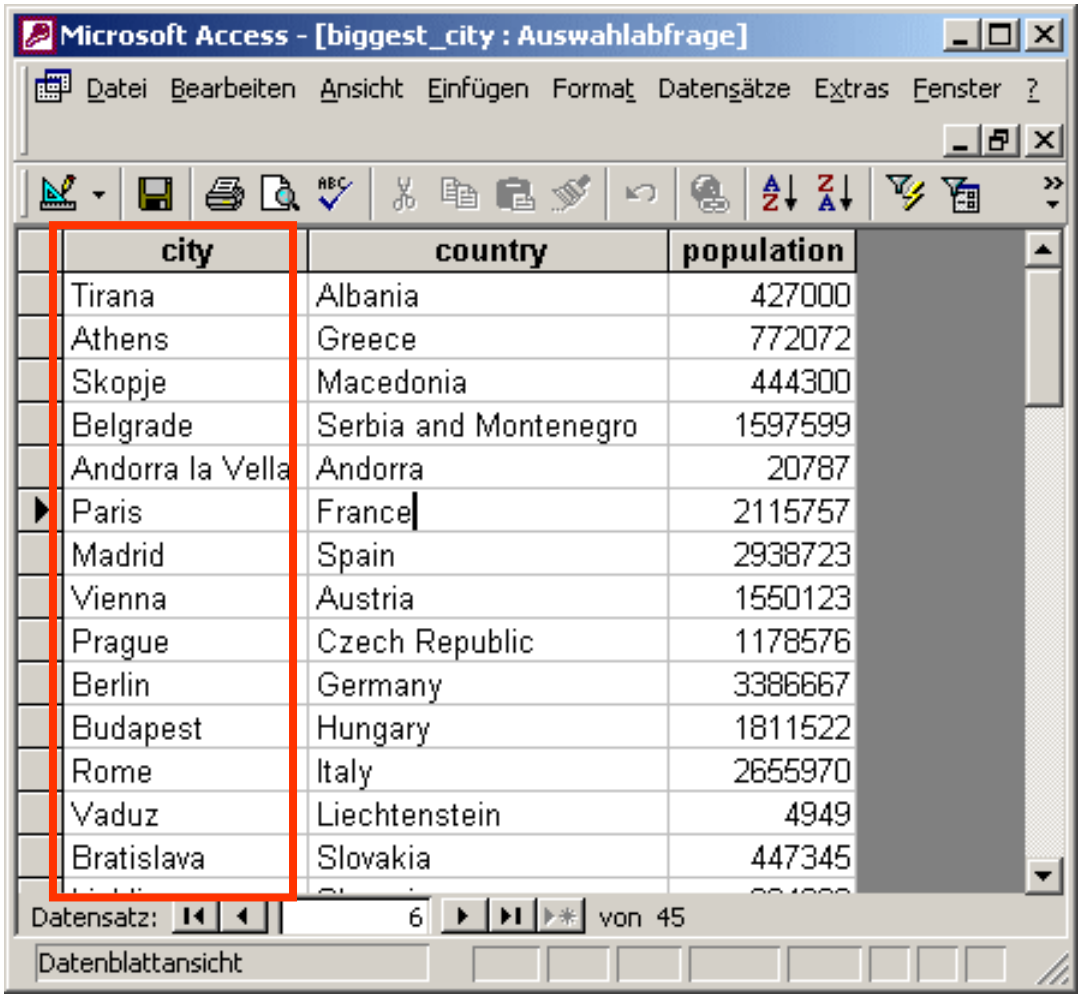
Find name and population of that city in each country, which is equal in size to the biggest city in that country ?



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 !



city	country	population
Tirana	Albania	427000
Athens	Greece	772072
Skopje	Macedonia	444300
Belgrade	Serbia and Montenegro	1597599
Andorra la Vella	Andorra	20787
Paris	France	2115757
Madrid	Spain	2938723
Vienna	Austria	1550123
Prague	Czech Republic	1178576
Berlin	Germany	3386667
Budapest	Hungary	1811522
Rome	Italy	2655970
Vaduz	Liechtenstein	4949
Bratislava	Slovakia	447345

Aggregation is very important for life sciences !

- „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

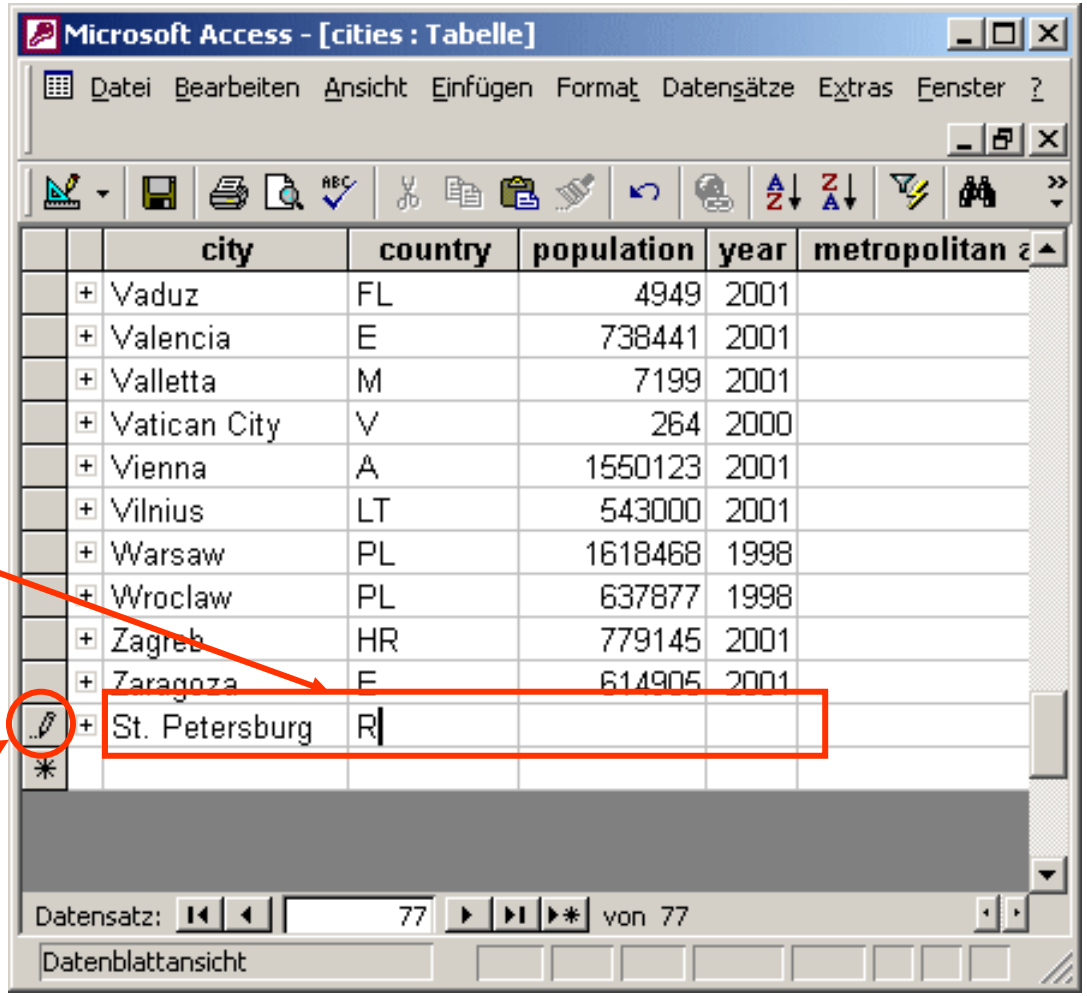
- Insertions and modifications are accepted by the DBMS only if the data types of the resp. fields declared in the schema of the table fit with the values in the new/modified records.

- 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

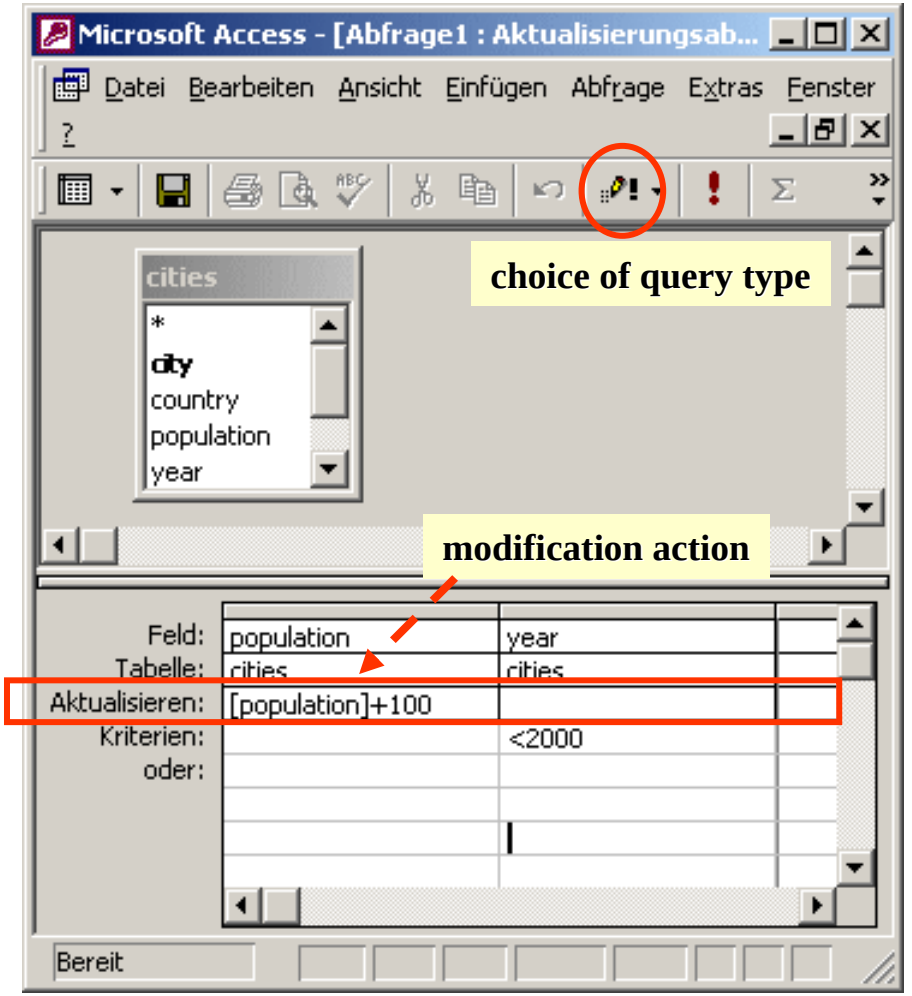
- In Access, individual updates can be performed directly in the datasheet view by manipulating the individual fields and records.

Record to be inserted into table cities (during insert)

Update mode indicated by write icon



- 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

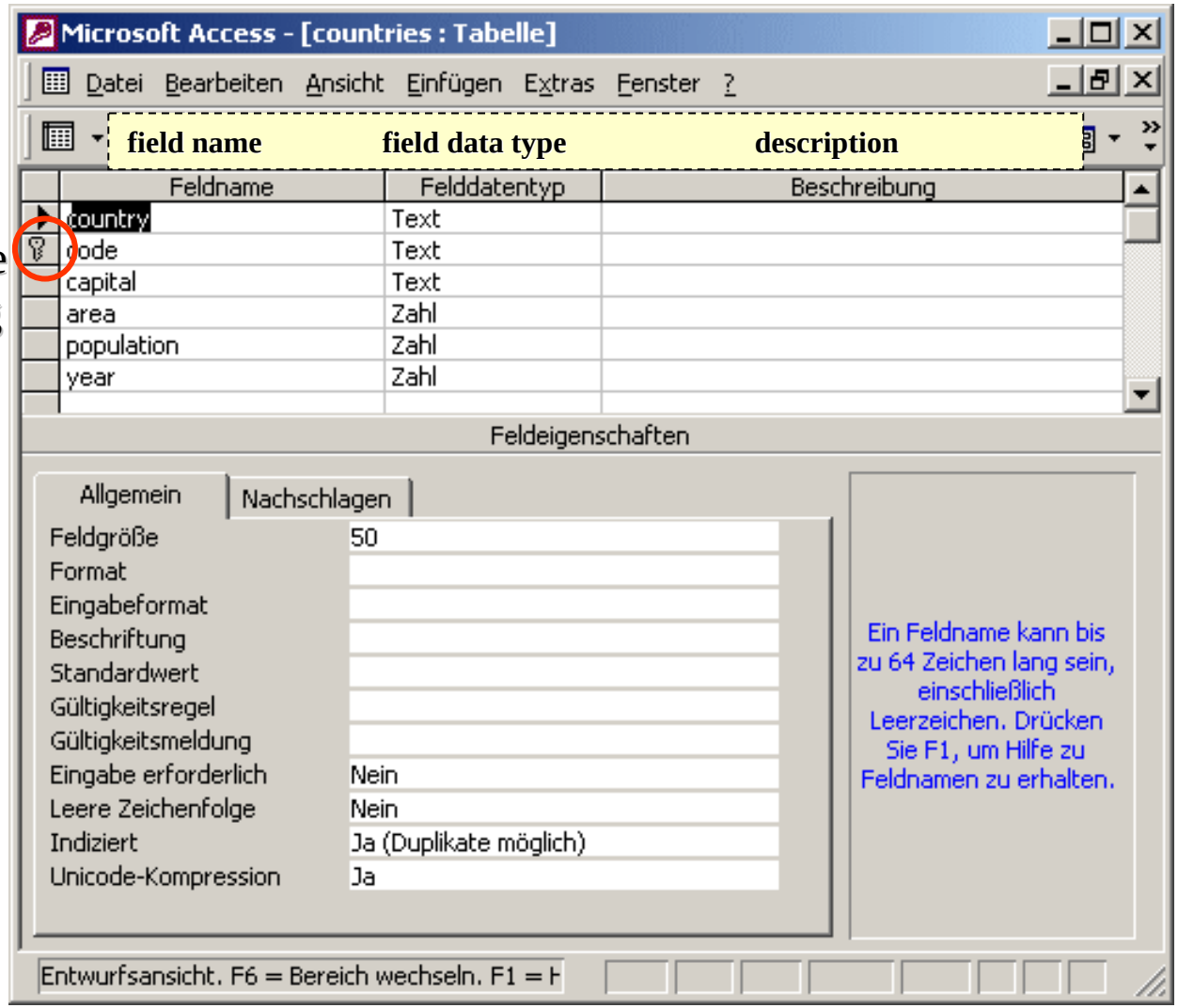


- Example (modify query):
  Increase the population of every city by 100, if the year value is older than 2000 !
- Result:
  - Candidate citites are identified by evaluating criteria conditions.
  - Modification action is applied to all qualifying records !

Access table „countries":

Design view

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

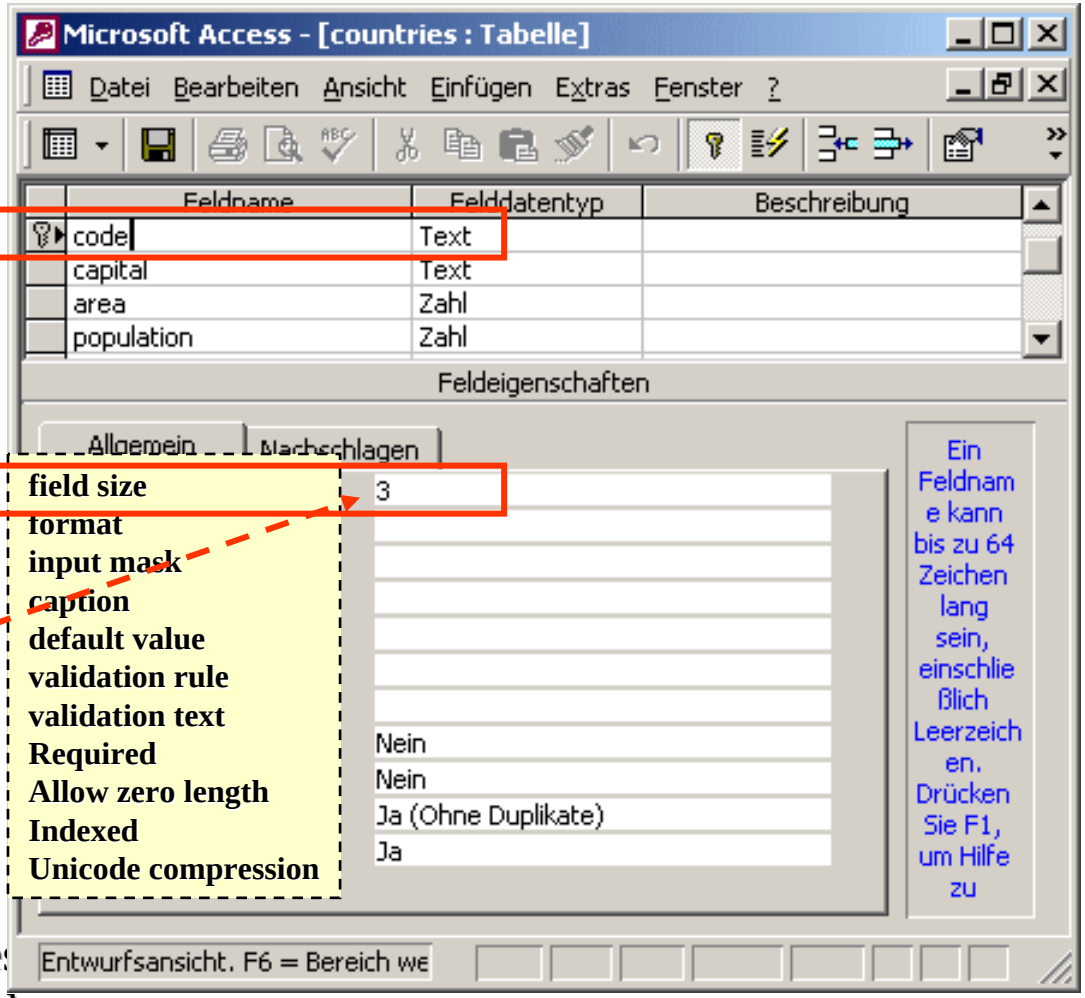


- **field size**
- **format**
- **input mask**
- **caption**
- **default value**
- **validation rule**
- **validation text**
- **Required**
- **Allow zero length**
- **Indexed**
- **Unicode compression**

- For each field in a table, a field data type has to be declared, e.g. text, number, date, yes/no.

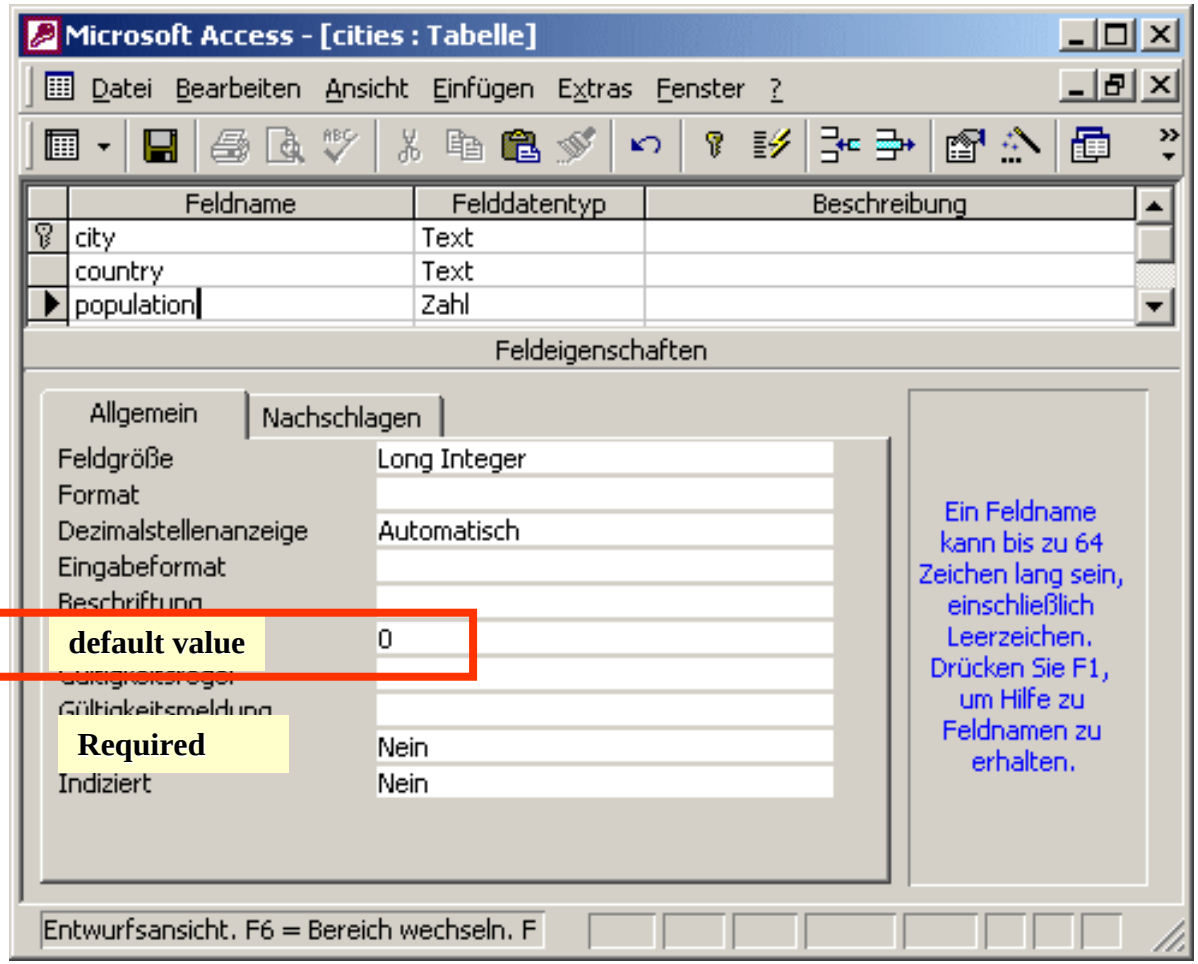
- For many of these types more detailed variations concerning the size of memory required for the resp. field can be declared in addition.

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

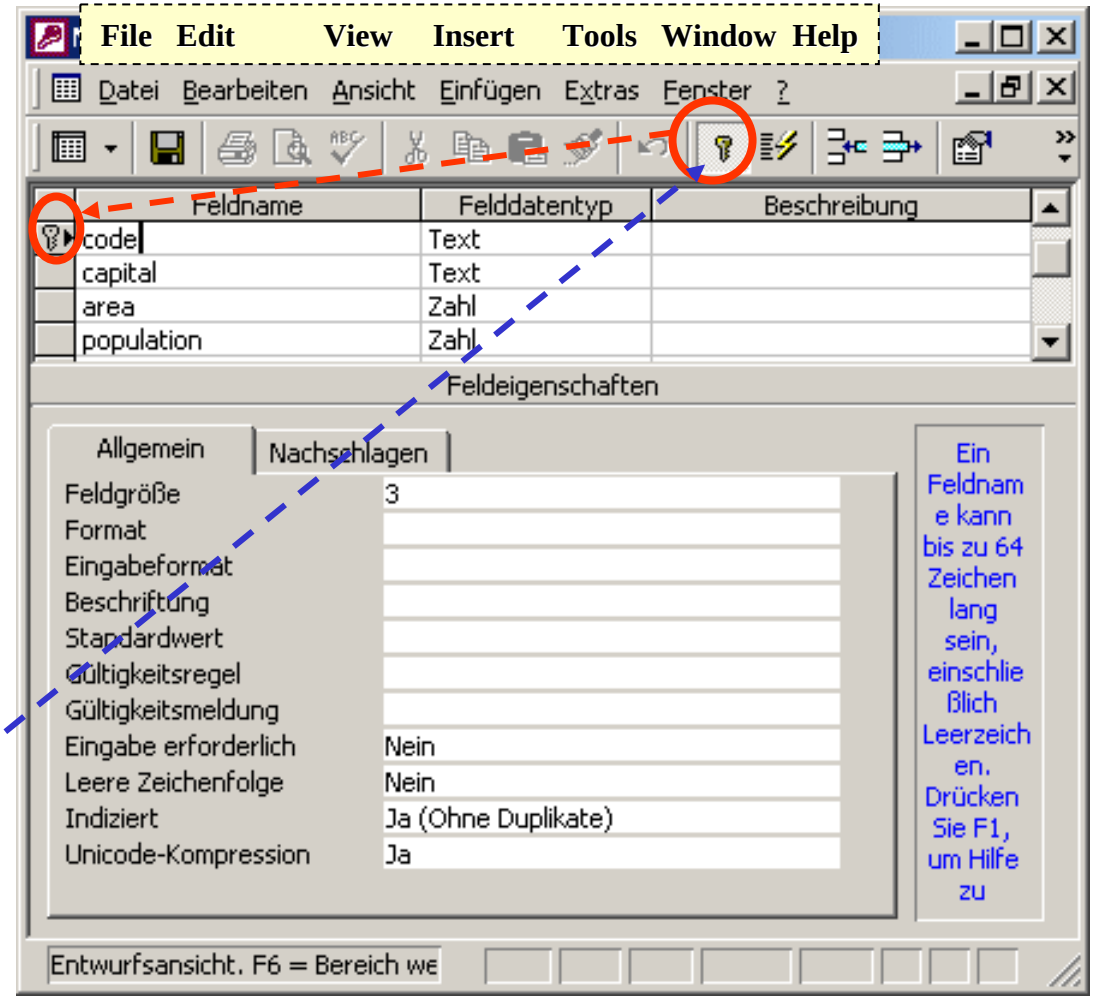


- For type ‚number': Various subtypes can be chosen from a menu accessible by clicking into the field size entry (e.g., integer, long integer, byte)

- 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.

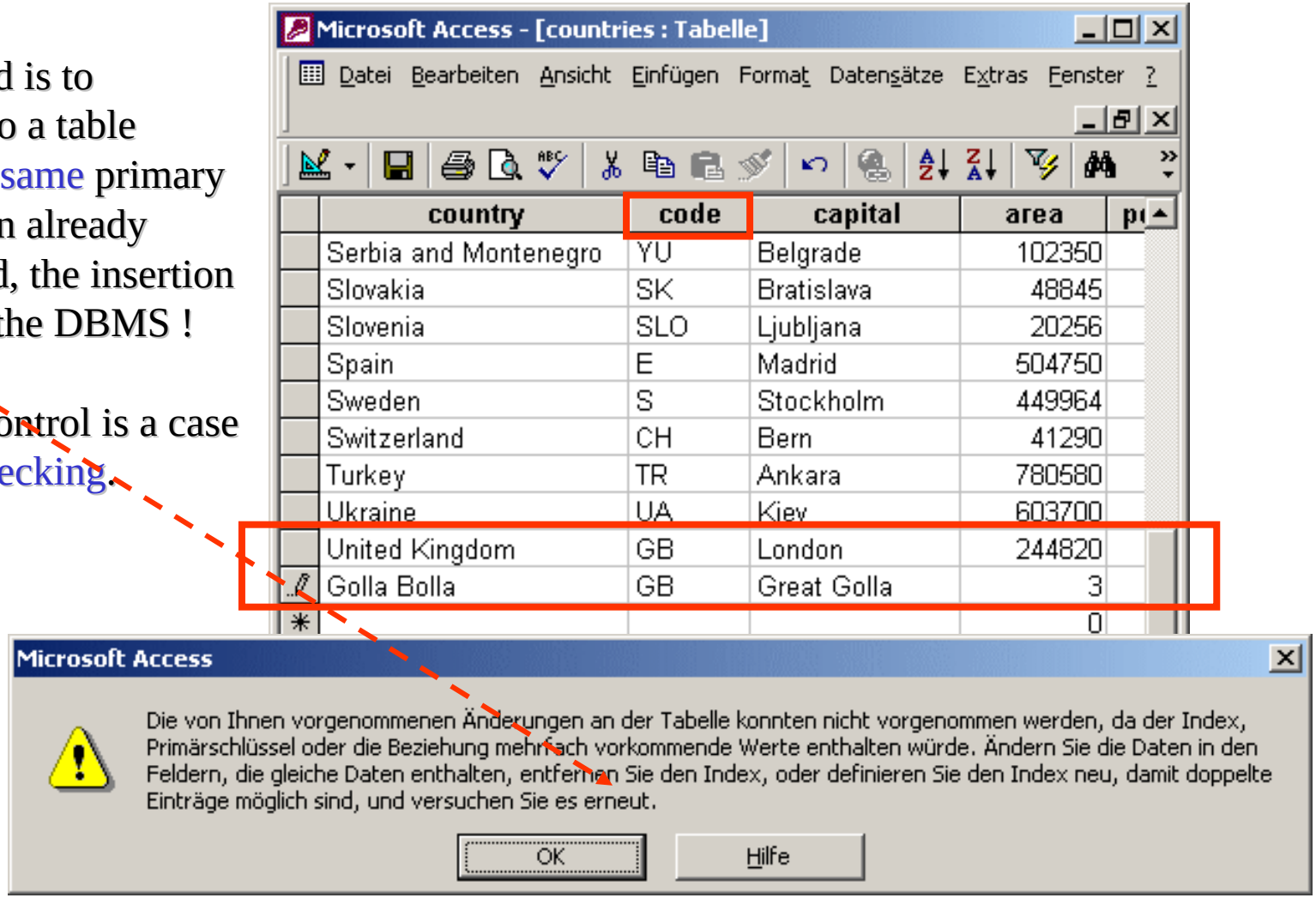


- In most tables, there is one or more field the values of which uniquely identify a particular record of the table.
- Such special fields (or combinations of fields) are called keys of the table.
- One such key ought to be designated at design time as the primary key of the table.
- In the example, 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.

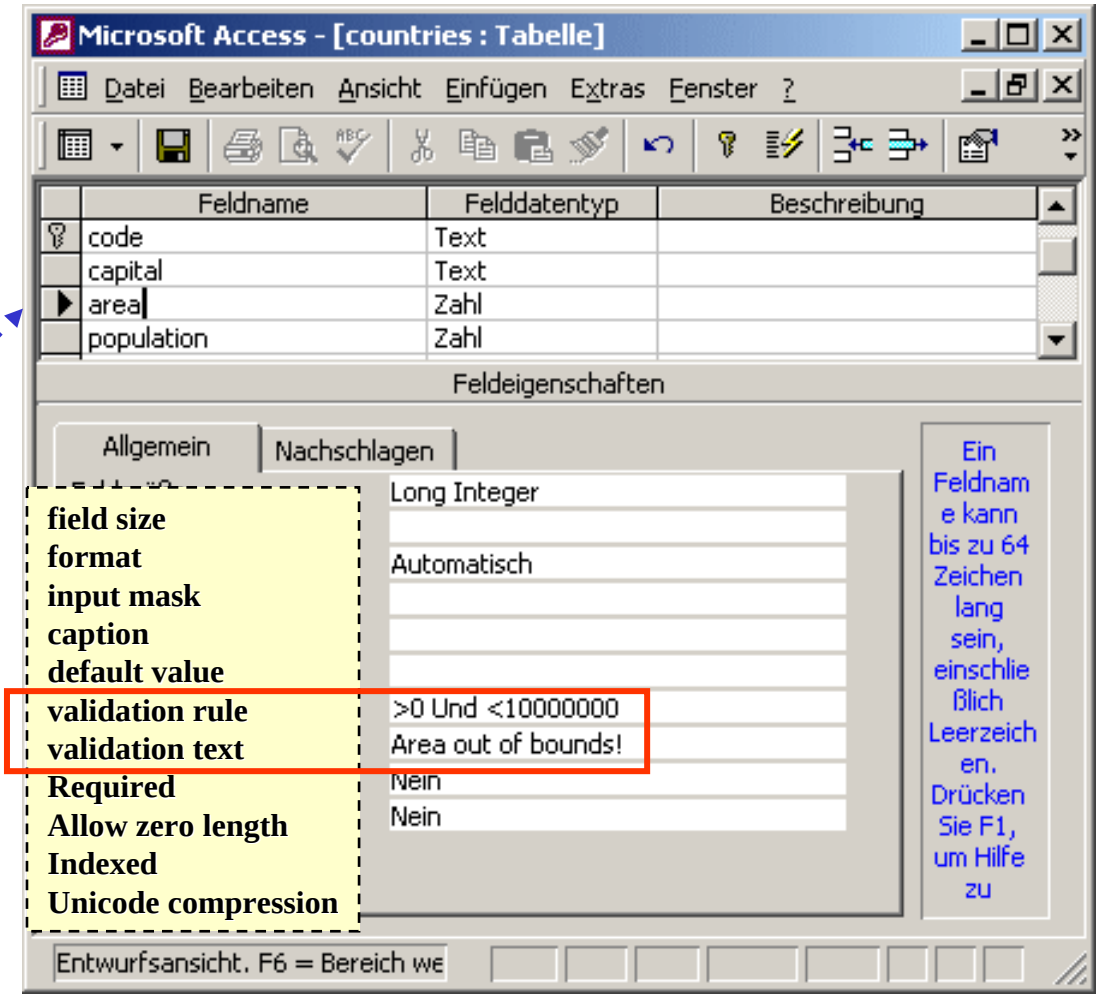


If a new record is to be inserted into a table which has the same primary key value as an already existing record, the insertion is rejected by the DBMS !

This kind of control is a case of integrity checking.

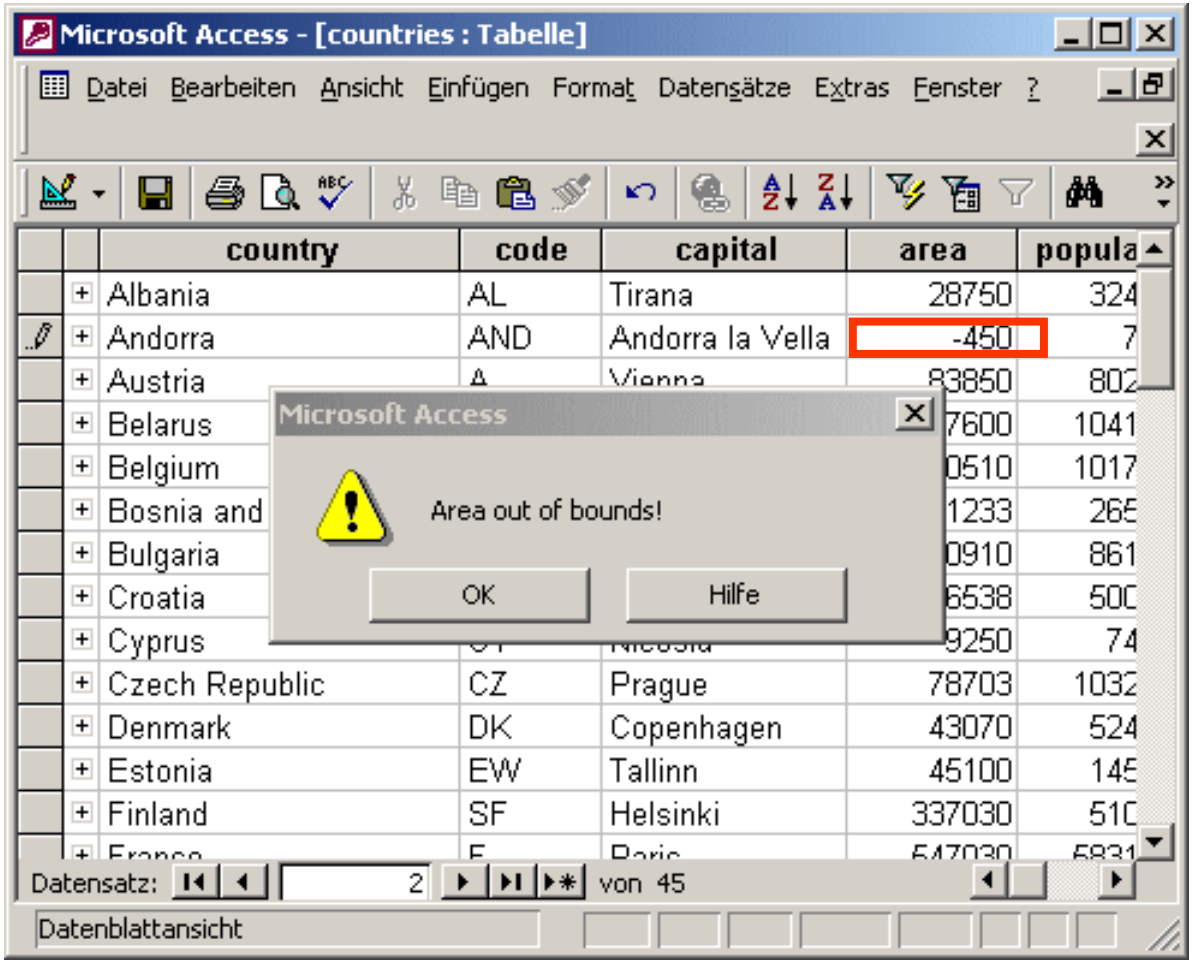


- Another means of controlling the contents of a DB table is the concept of a validation rule.
- Such a rule can be associated with each field of a table (here with ‚area' in „countries").
- A validation rule is a logical condition defining one or more properties of each proper value in this field.
- 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.

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



- 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.

- 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.

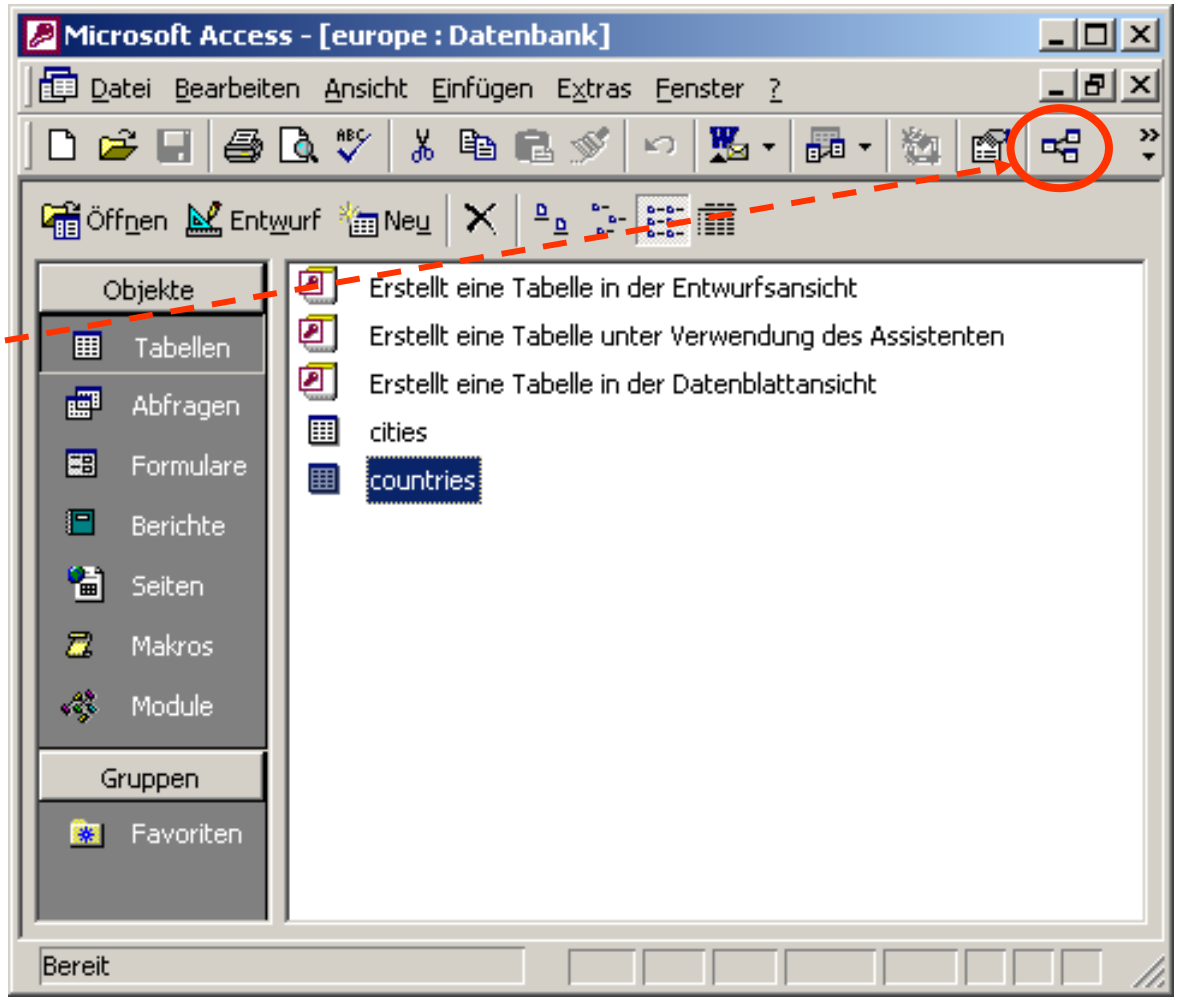
- Integrity constraint violations – likely to happen during DB modifications – are controlled automatically by the DBMS. Each insertion, deletion or update of a table is checked for possibly violating any constraint prior to the execution of the resp. modification:

  Integrity checking

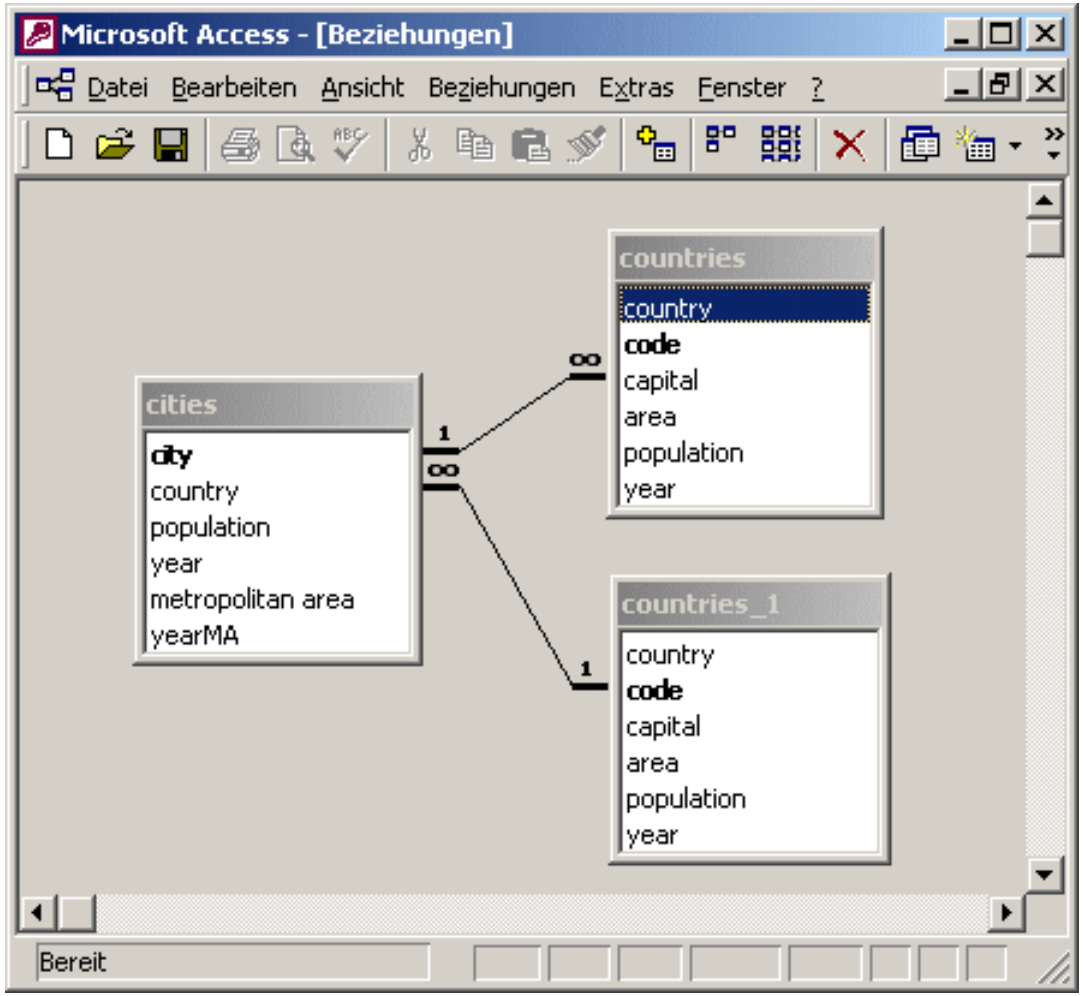
- If integrity violations are detected, the DBMS either refuses to perform the desired modification or „repairs“ the semantic mistake causing the violation automatically, if possible.

- Key and validation rule violations cannot be „repaired“!

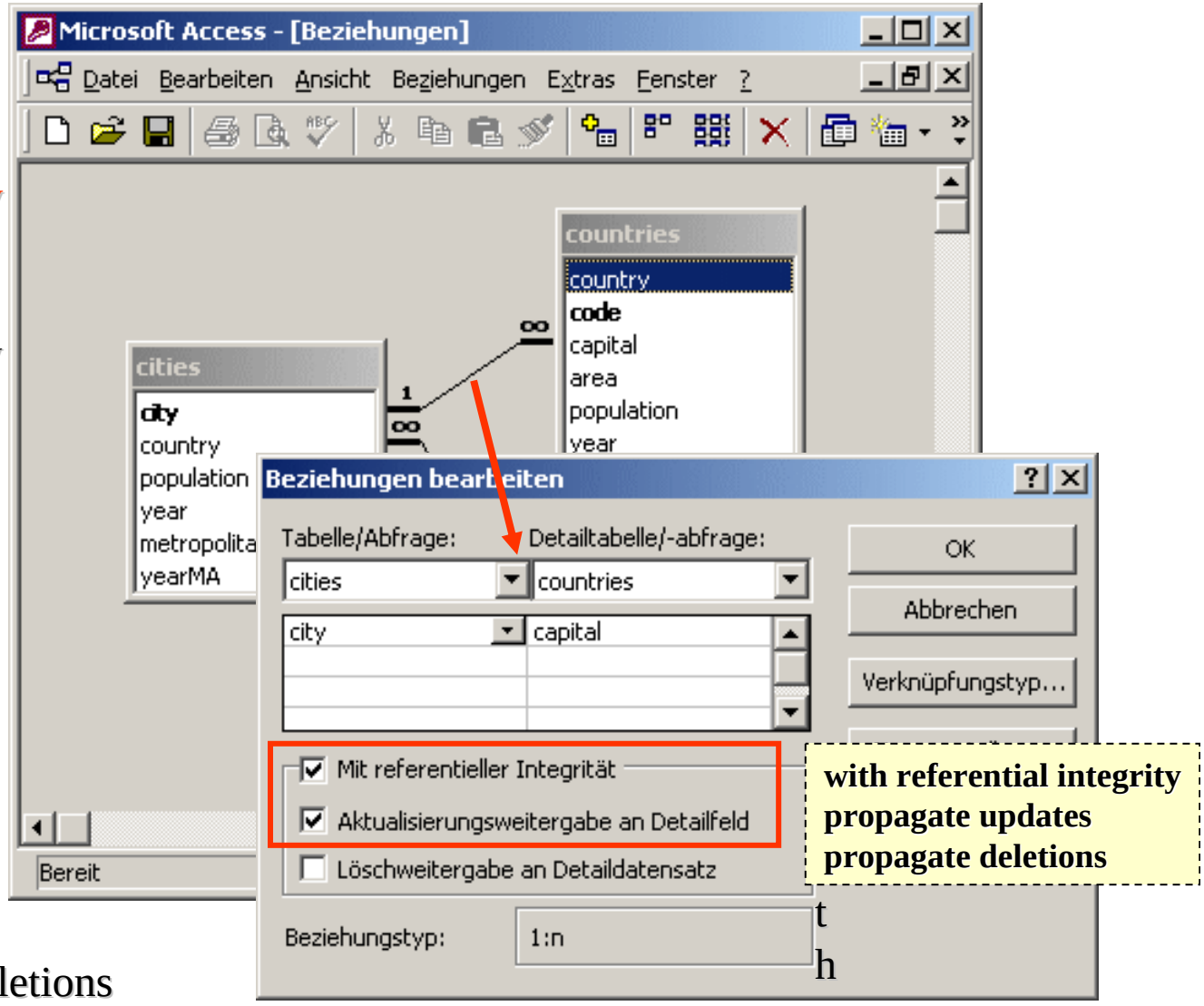
- A third type of integrity constraint can be established, if relationships between tables have been declared before.
- By clicking on the relationship icon in the main icon list of an Access database, you can switch to relationship design mode, which looks like this . . .



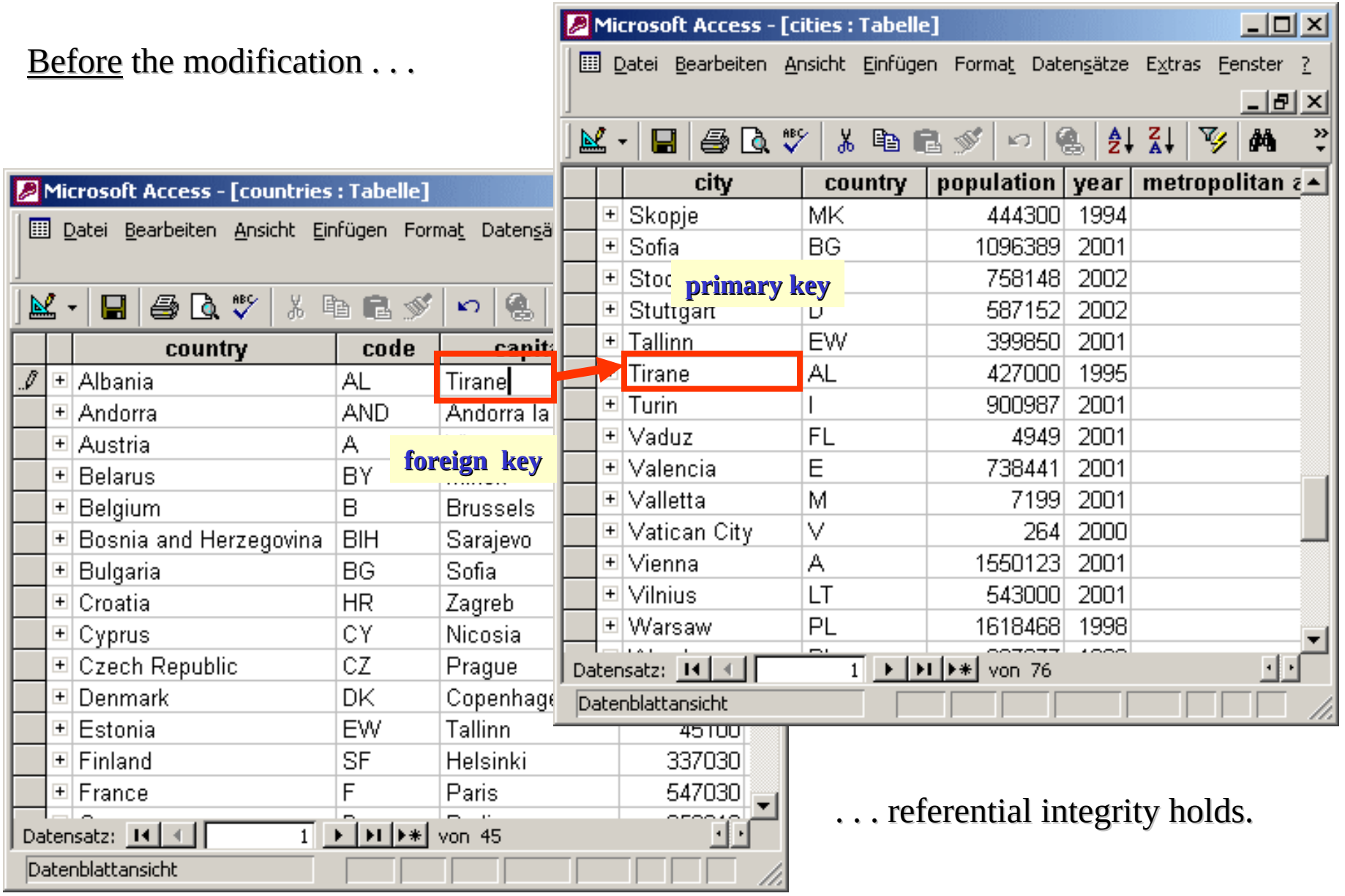
- In our example database on European geography, two relationships have been declared between the two tables.
- As both tables are involved both as referencing as well as as referenced table, one of the has to occur twice.
- Relationships are established between fields of the table which have identical 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 propated to the referencing field
  - Analoguously for deletions



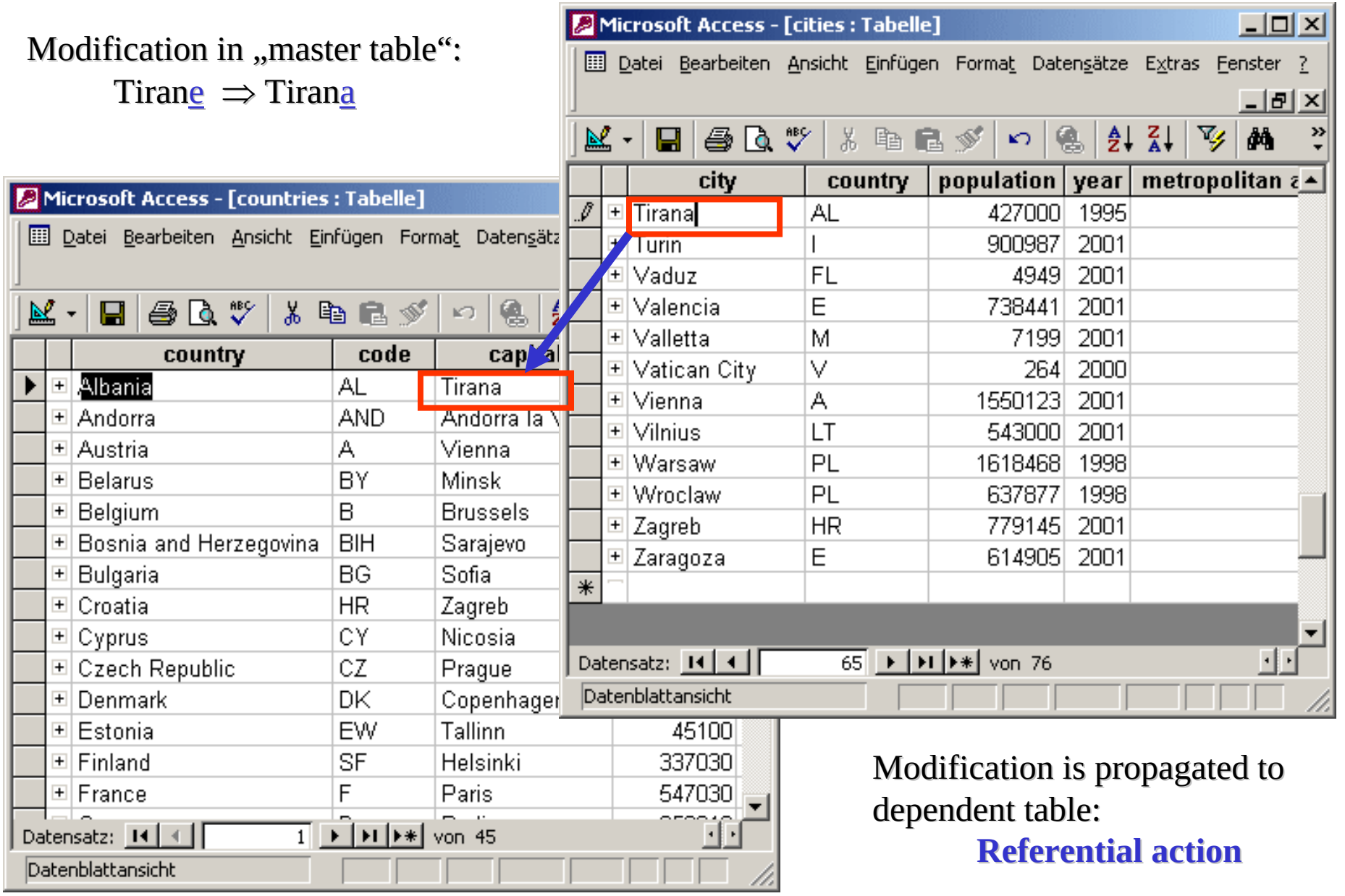
Before the modification . . .



. . . referential integrity holds.

Modification in „master table“:

Tirane ⇒ Tirana

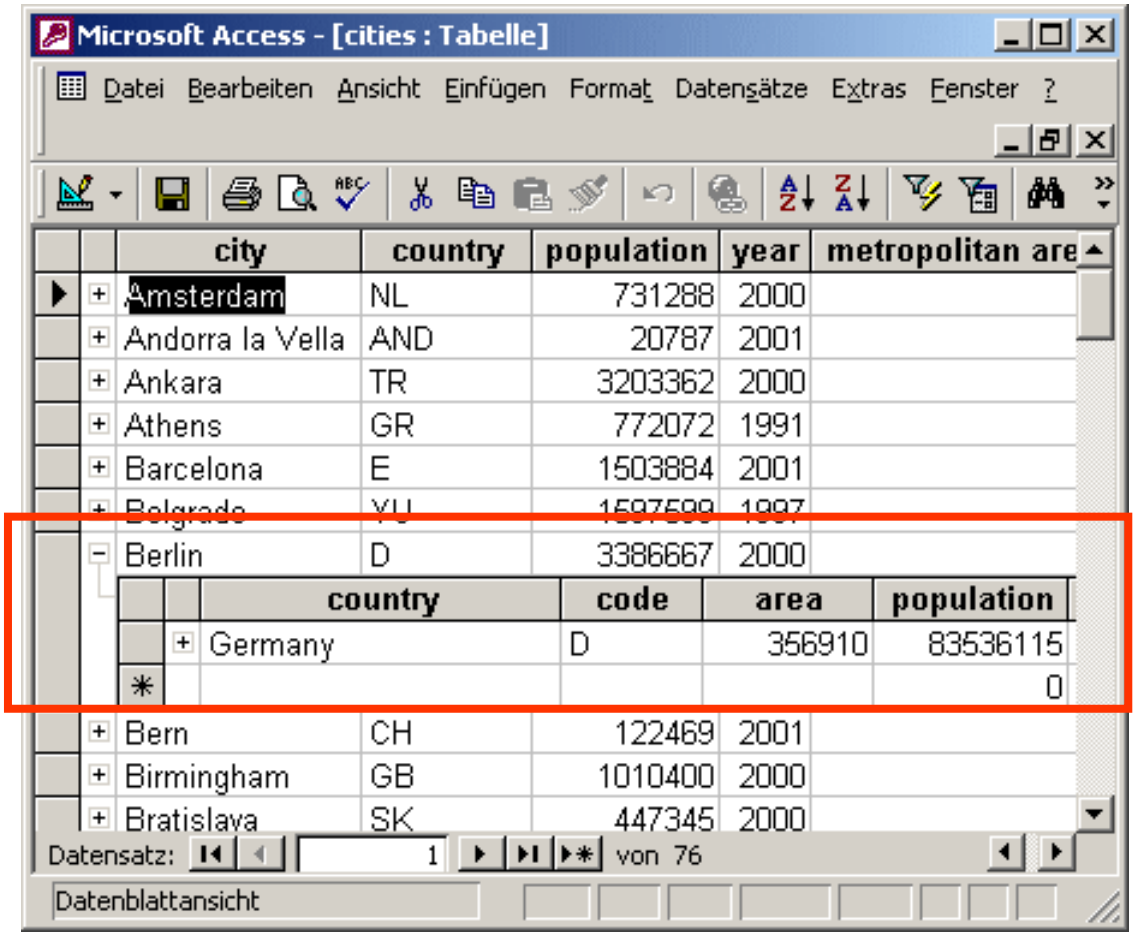


Modification is propagated to dependent table:

Referential action

- 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‘



- 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:

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

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