Database

LBF/ZAB21 Software equipment in dental office

Mgr. Markéta Trnečková, Ph.D.
www.marketa-trneckova.cz

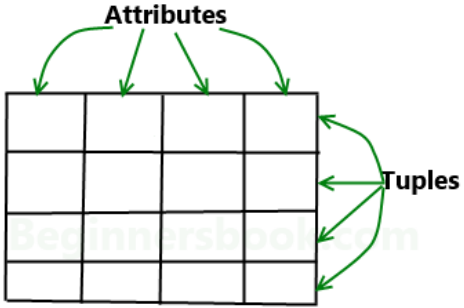
Palacký University, Olomouc

Database

- **database** = organized collection of data
- earlier network, hierarchy
- **relational database** - most common
- **data base** = data itself
- **DataBase Management System** (DBMS) = software that interacts with end users, applications, and the database itself to capture and analyze the data
- **Examples of DBMS**:
  - mySQL - web projects
  - MS SQL - Microsoft, smaller projects, especially in MS softwares
  - Oracle - big database projects
- **SQL** - Simple Query Language
- there exists non-relational databases - for example NoSQL

Relational database

- basic model - **tables**
- tables are interconnected via relationships
- **attributes** - columns in tables, characteristic of objects in tables
- **tuples** (objects) - rows of tables
- objects are the same kind, but each is in database only once (entity integrity)
- attributes $\neq$ concrete values, only characteristic



Attributes

- columns in table, characteristic
- describe the instances in the row of a database
- **domain** - set of possible values for a given attribute
- **data type** - is derived from domain, e.g. integer
- DBMS purpose: do not allow to write a value in the column that does not fit the data type
- **attribute atomicity** - attribute is atomic (or simple) if it does not contain any meaningful smaller components (name $\rightarrow$ name, surname)
- each attribute contains one value - if we need more - new table
- example:
  - name, surname, phone-number
  - more phone numbers per person
  - 2 tables; name surname; phone-number, person

Special attributes

Primary key

- tuples are not ordered
- we need to identify each tuple
- one attribute **unique** = primary key
- **!** each tuple has unique value of primary key in table **!**
- each table **must** contain primary key

Foreign key

- primary keys within a database are used to define the relationships among the tables
- e.g. person: person-id, name, surname
- when a PK migrates to another table, it becomes a foreign key in the other table
- e.g. phone: phone-id, phone-number, person-id

Data types

- **Integer** - number
- **Varchar** - short text (max. 255 chars)
- **Text** - long text
- **Boolean** - logical value (true/false,...)
- **Date/DateTime** - date, or date and time

Relationships

- logical connection between different tables
- e.g. v database of library - tables author and book
- meaning of relationship (author wrote book), multiplicity

Multiplicity of relationship

- how many tuples in first table could be connected with one tuple in second table and vice versa
- e.g. How many books can author write, how many authors could write one book

relation 1:n - one-to-many

- simple
- e.g. football team (1 team contains more players, each player belongs to 1 team)
- new foreign key attribute in table player, that identifies team

relation m:n - many-to-many

- record in both tables can relate to any number of records (or no records) in the other table
- e.g. book, author
- many-to-many relationships require a third table, known as an associate or linking table, because relational systems can't directly accommodate the relationship
- e.g. table book-author

Multiplicity of relationship

relation 1:1 - one-to-one

- only one (or no) record on either side of the relationship
- e.g. spouses — you may or may not be married, but if you are, both you and your spouse have only one spouse
- not common

Database normalization

- process of structuring a relational database in accordance with a series of so-called **normal forms** in order to reduce data redundancy and improve data integrity

First normal form (1NF)

- domain of each attribute contains only atomic values, and the value of each attribute contains only a single value from that domain
- **reason**: non-atomic attribute = limited work with the table (e.g. searching for people from the same city)

Second normal form (2NF)

- is in first normal form (1NF), all attributes are **dependent on primary key**
- bad example: table student (id, name, surname, university, university-address)
- university-address is not dependent on student
- an error could arise when changing the address of the university (we need to change the address for all students)
- **solution**: new table with informations about university

Normalizace databáze

Third normal form (3NF)

- is in 2NF, all attributes are **dependent directly on primary key**
- example employee and pay
- if pay is derived from job position, than is not dependent on primary key but on position
- new table with positions and pays

Reference integrity

- is a property of data stating that all of its references are valid
- well-designed database = DBMS can provide between the linked tables et reference integrity
- we are not able to remove author, if there exist book he wrote
- we are not able to change primary key, if it is used as foreign key in different table

Relational scheme

- attributes: A_1, A_2, ..., A_n
- relational scheme: $R = (A_1, A_2, \ldots, A_n)$
- e.g. book = (id, title, author, number-of-pages)
- relation over relational scheme: r(R)
- e.g. title(book)
- **instance**: actual values - table
- relation entity: tuple (row in tabule)
- eg. 1 "Romeo and Juliet" "William Shakespeare" 151

Basic operations

Selection

- selection of rows (tuples)
- usually uses some condition $\rightarrow$ rows satisfying the condition
- `select * from book where number-of-pages > 100`
- we can compose the conditions via logical operators - or, and, ...

Projection

- selection of columns, we want to show
- `select author from book`

Operations between tables

Difference

- $r - s$
- two relations need to be **compatible** (same number of attributes, same domains)

Table : r

A	B
α	1
α	2
β	1

Table : s

A	B
α	2
β	3

Table : $r - s$

A	B
α	1
β	1

Operations between tables

Union

- $r \cup s$
- **compatible** relations

Table : r

A	B
α	1
α	2
β	1

Table : s

A	B
α	2
β	3

Table : $r \cup s$

A	B
α	1
α	2
β	1
β	3

Operations between tables

Cartesian product

- $r \times s$
- relations have different attributes (if they have same name, one must be renamed)
- union all rows of r with all rows of s

Table : r

A	B
α	1
β	2

Table : s

C	D	E
α	10	a
β	10	a
β	20	b
γ	10	b

Table : $r \times s$

A	B	C	D	E
α	1	α	10	a
α	1	β	10	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

Another operations

Rename

- it is not classic operation
- we can name the result of some operation

Composition

- more complex operations = composition of basic operations
- e.g. cartesian product $r \times s$ and selection such rows, where $A = C$

Table : r

A	B
α	1
β	2

Table : s

C	D	E
α	10	a
β	10	a
β	20	b
γ	10	b

A	B	C	D	E
α	1	α	10	a
β	2	β	10	a
β	2	β	20	b

Additional operations

- for practical reasons, other derivative operations are introduced - simplification

Conjunction

- $r \cap s$
- compatible relations
- result: rows, contained in both relations

Table : r

A	B
α	1
α	2
β	1

Table : s

A	B
α	2
β	3

Table : $r \cap s$

A	B
α	2

Additional operations

Natural join

- $r \bowtie s$
- associated relations r and s have one or more pairs of identically named columns
- columns must be the same data type
- result: join, where identically named rows are the same (values)

Table : r

A	B
α	1
β	2

Table : s

A	D	E
α	10	a
β	10	a
β	20	b
γ	10	b

Table : $r \bowtie s$

A	B	D	E
α	1	10	a
β	2	10	a
β	2	20	b

Additional operations

Division

- $r \div s$
- query with "for all"
- input relations: $r = (A_1, \ldots, A_n, B_1, \ldots B_m)$, $s = (B_1, \ldots B_m)$
- returned relation: $r \div s = (A_1, \ldots, A_n)$
- $r \div s = \{t | t \in \Pi_{R-S}(r) \wedge \forall u \in S : tu \in r\}$
- $\Pi_{R-S}(r)$ means selection of columns

Table : r

A	B	C	D	E
α	a	α	a	1
α	a	γ	a	1
α	a	γ	b	1
β	a	γ	a	1
β	a	γ	b	3
γ	a	γ	a	1
γ	a	γ	b	1
γ	a	β	b	1

Table : s

D	E
a	1
b	1

Table : $r \bowtie s$

A	B	C
α	a	γ
γ	a	γ

Additional operations

Table : r

A	B	C	D	E
α	a	α	a	1
α	a	γ	a	1
α	a	γ	b	1
β	a	γ	a	1
β	a	γ	b	3
γ	a	γ	a	1
γ	a	γ	b	1
γ	a	β	b	1

Table : s

D	E
a	1
b	1

Table : $r \bowtie s$

A	B	C
α	a	γ
γ	a	γ

D E = a 1

A	B	C
α	a	α
α	a	γ
β	a	γ
γ	a	γ

D E = b 1

A	B	C
α	a	γ
γ	a	γ
α	a	β

MS Access

- Microsoft Access is a Database Management System (DBMS) from Microsoft
- part of the Microsoft Office suite of applications

MS Access

To use MS Access, you will need to follow these four steps

- Database creation - create database and specify what kind of data you will be storing
- Data input - data can be entered into the database
- Query - term to basically describe the process of retrieving information from the database
- Report (optional) - information from the database is organized in a nice presentation that can be printed

MS Access - objects

- **Table** - table is an object that is used to define and store data
- you need to define fields which is also known as column headings
- each field must have a unique name, and data type
- define a primary key
- **Query** - an object that provides a custom view of data from one or more tables
- when you build a query, you are defining specific search conditions to find exactly the data you want
- you can define queries to Select, Update, Insert, or Delete data

MS Access - objects

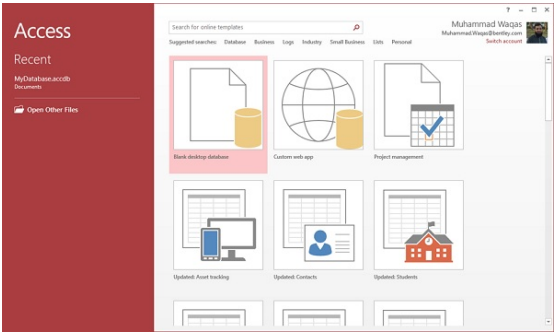
- **Form** - an object in a desktop database designed primarily for data input or display or for control of application execution
- an easy way to guide people toward entering data correctly
- **Report** - an object in desktop databases designed for formatting, calculating, printing, and summarizing selected data
- you can customize a report's appearance to make it visually appealing

Example - library

- we want to store data about books and authors
- **book** - title, authors, number of pages
- **author** - name, surname

Create database

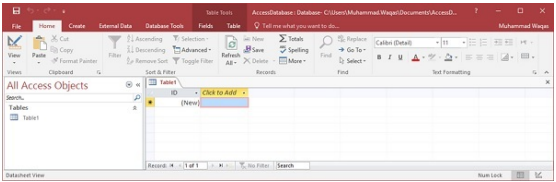
- Open MS Access - you will see the following screen in which different Access database templates are displayed



- Select Blank desktop database. Enter the name and click the Create button

Tables

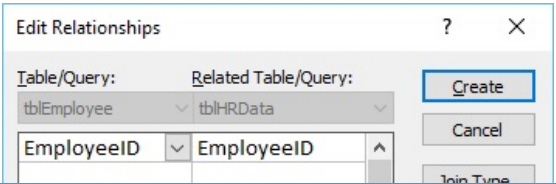
- Access will create a new blank database and will open up the blank table



- you should always start your design of a database by creating all of its tables and then creating any other object
- ID which is an AutoNumber field acts as our unique identifier and is the primary key for this table
- ID field could be renamed to suit our conditions (for example ISBN) — Table Tools - Fields - Name & Caption
- add some more fields by clicking on click to add - specify data type
- field in a table has properties — define the field's characteristics and behavior (the most important DataType)
- click the Save icon

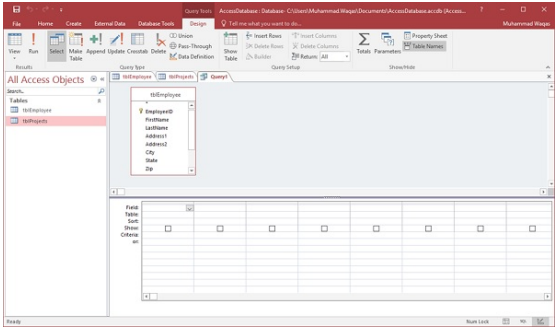
Defining Relationships

- the relationship matches the primary key from one table, which provides a unique identifier for each row, with an entry in the foreign key in the other table
- one-to-many relationship is the most common type of relationship
- a row in table A can have many matching rows in table B, but a row in table B can have only one matching row in table A
- many-to-many relationship, a row in table A can have many matching rows in table B, and vice versa
- create such a relationship by defining a third table, called a junction table, whose primary key consists of the foreign keys from both table A and table B
- Database Tools – Relationships – add tables –drag and drop or double click on one table – new window (create or edit relationship)
- Enforce referential integrity



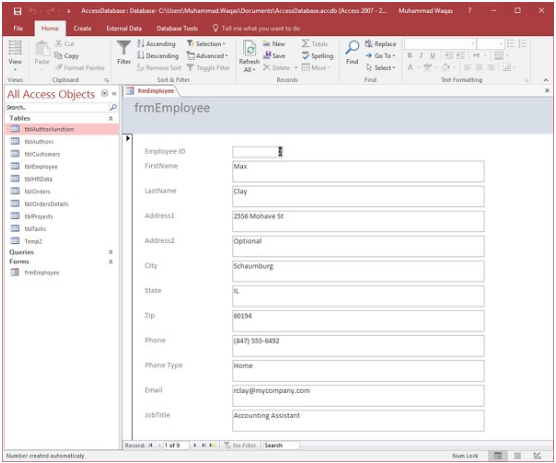
Create Query

- Create tab — Query Design
- Tables tab, on the Show Table dialog, double-click the table
- double-click all those fields which you want to see as result of the query
- Run on the Design tab, then click Run



Forms

- objects through which you can add, edit, or display the data stored in your database
- Create — Form
- automatically created, you can edit them



Task

- create Microsoft Access database for the doctor
- doctor records patients (name, surname, personal identification number, phone, address, . . .)
- records patient's visits (patient, date, symptoms, diagnosis, treatment, . . .)