

Interreg



Austria-Czech Republic

European Regional Development Fund

INFORMATICS

Database



UNIVERSITY
OF APPLIED SCIENCES
UPPER AUSTRIA



EUROPEAN UNION

Contents

1. Basic concepts of the database	4
1.1. Database management system	4
1.2. Database Structure	6
1.3. Database table.....	6
1.4. Primary key	7
1.5. Relation.....	7
1.6. Primary and secondary table	7
1.7. Types of relations	8
1.8. Referential integrity	9
1.9. Operation of the database.....	9
2. Database models	10
2.1. Hierarchical data model	10
2.2. Network Data Model.....	10
2.3. Relational data model.....	11
2.4. Attached application	12
2.5. Object databases.....	12
2.6. Fundamentals of Object Orientation. Objects and classes.....	13
2.7. Literals.....	14
2.8. Operation	14
3. Integrity of the database.....	16
3.1. Types of integrity constraints	16
3.2. Maintaining integrity constraints.....	16
3.3. Relations between tables	17
3.4. Normal forms.....	18
3.5. Database integrity	18
4. Relational database model.....	20
4.1. 12 rules for relational DBMS:	20
4.2. Relational Data Model	22

4.3.	Relational Data Model Properties.....	23
4.4.	Relationships in the relational model	23
5.	SQL Basics	25
5.1.	Introduction to SQL (description and properties)	25
5.2.	SQL queries	27
6.	SQL - more complex queries	33
6.1.	Example 1	33
7.	Graph databases.....	37
7.1.	Relational databases.....	37
7.2.	True graph databases.....	37
7.3.	One-way relationships.....	38
7.4.	Two-way relationships.....	39
7.5.	Summary	41
7.6.	Graphic Database Modeling Methodology.....	41
7.7.	Implementation of the conceptual model.....	42
8.	NoSql database	43
9.	Transaction	45
9.1.	Solution: Transactions	45
9.2.	Transactions in SQL	46
10.	Procedures and functions, triggers and sequences.....	48
10.1.	Formal view	48
11.	Analytical tools - OLAP.....	52
11.1.	OLAP - Information Under Control	52
11.2.	What and how it simplifies work.....	52
11.3.	How to get this information?	53
11.4.	Analytical tools.....	55
11.5.	OLAP cube	55
12.	Specifics of database systems. Access technology to databases. Geographic information systems.	57
12.1.	Data Access Overview in ASP.NET Technology	57

13.	Database Objects	58
13.1.	Basic concepts of database technology.....	59
13.2.	Geographic information system	60
13.3.	Geographic data	61
13.4.	Data Collection.....	61
13.5.	Multidimensional cube	62
13.6.	Saving data in the OLAP database	63

I. BASIC CONCEPTS OF THE DATABASE

The database (or data base) is a certain set of information (data), mostly a table with records stored on a storage medium. In the broader sense, the database includes software resources that enable to manipulate and access stored data.

Database is a set of records that are gathered for a specific purpose. We mainly use databases for storing comprehensive information. Database systems are available as a part of office packages (e.g. MS Access, OpenOffice.org Base). These systems are also available as standalone programs that are used to create large databases, for example, MySQL, Oracle and others.

Database is a set of data (information about real-world objects) that is interlinked in some way. Data is an expression for data used to describe a phenomenon or property of the observed object. They represent a form of presentation of real objects (characters, symbols, images, facts, events), thus reflecting the state of reality at a certain point in time.

Information is a message that a certain phenomenon has occurred. It arises from assigning meaning to data and exists in relation to the recipient. It serves to inform about changes in perceived reality. We encounter databases in everyday life very often. We present the common use of large-scale databases:

- database of timetables,
- the state administration database,
- Information systems of banks, schools, offices
- hospital patient recording systems,
- Libraries databases.

I.I. Database management system

Database management system (DBMS -) is software equipment that ensures working with the database, i.e. it creates an interface between application programs and stored data. Sometimes this concept is confused with the concept of a database system. However, the database system is a DBMS along with a database.

Entity

Any object (person, animal, thing or phenomenon) of the real world that is captured in the data model. An entity must be distinguishable from other entities and exist independently of them.

Data

Expression for data used to describe a phenomenon or property of the observed object. Data are obtained by measuring or observing, and can be divided into continuous and attribute data. Continuous data are related to a continuous scale, while attributive data are not.

Information

Information is data that brings us new findings.

Records and attributes

Table rows with attribute values for one object (entity) that must differ from each other. Attributes are columns of the table. Attributes are assigned a specific data type and domain, which is a set of admissible values of the given attribute. The row is cut over the columns of the table and serves to save the data.

Attributes, fields

Objects (entities) properties monitored:

- create table columns
- can acquire various values
- the fields are of a particular data type (number, text, date, ...)

Primary key

It identifies the record (line of the table). It is an attribute (field) that has a unique value for each entity, such as a birth number; usually it is an auxiliary field with a record ID number.

Foreign key

An attribute that is the primary key in another table.

Index

This is a way of arranging a table.

- the order of the records in the table during the "life" of the database does not change; the index helps to find data quickly in the table
- The index creates an auxiliary file with a table arranged by a specific field
- one table can have several indexes arranged by various attributes
- the primary key is always an index

1.2. Database Structure

The most common databases are relational databases. In these, the data are stored in smaller tables to ensure minimal data redundancy. Tables are interconnected using relations. Relations determine relationships between tables and determine the interconnectiveness of individual tables. Each of these tables should contain data related to only one type of object (e.g. table of orders, clients, prices, goods, etc.). In order to create a good database, it is necessary first to suggest the appropriate structure of individual tables. These tables must then be interlinked by relations. Tables form the basis of the entire database structure. The basic rules for table design are as follows:

- any information should only be included in the database once,
- each table should contain information about one type of object,
- when designing tables, the future extent of the data should be taken into account.

1.3. Database table

One table should contain information about one type of object. The database table is similar to a regular table. The lines contain records (about one object) and as columns, items or fields are designated. The field sometimes refers to the intersection of a line and a column that contains a single value (data element).

Zaměstnanci							
	ID_zamestn	Jméno	Příjmení	Datum naro	Pohlaví	Telefonní čí	ID_funkce
+	1	Tomáš	Novák	28.4.1980	muž	723 123 456	F_03
+	2	Josef	Koblížek	15.3.1976	muž	728 452 123	F_01
+	3	Petra	Maková	5.2.1985	žena	724 556 115	F_02
+	4	Václav	Sýkorka	30.6.1970	muž		F_06
+	5	Denisa	Rosolová	12.12.1956	žena		F_05
+	6	Michal	Aspik	3.6.1976	muž		F_04
+	7	Dominik	Kokeš	14.2.1981	muž		F_04
+	8	Tereza	Železná	25.10.1978	žena		F_02
+	9	Vladimíra	Mimořádná	17.11.1989	žena		F_02
+	10	Jakub	Pekelník	5.12.1973	muž		F_05

In the above table, the employees create rows of records with information about the individual employees. The columns represent fields where we always see one type of data (text, date, number). Each column (item) has a name, selected data type (e.g text, number, yes / no, date and time) and size. Other features (format, default, etc.) can be assigned. See more about this issue in video guides.

1.4. Primary key

In most cases, we need to identify each entry in the table. The so-called primary key is used for this purpose. A primary key is an field that is intended to ensure unambiguous identification of the individual records in the table. The primary key is usually a single field (so-called simple primary key), but it can also consist of more fields of the tablefield (the so-called composite primary key).

There can be only one primary key in each table. With the primary key, information is searched for more quickly, and relations with other tables are created. The values in the primary key field must be unique for each record. Primary key is one of the indexes. For quicker recordssearch and sorting according to a particular table field it is advisable to use field indexing (indexes). If we need to searchaccording to a different table column than the primary key, we set the index to it. With the index set, sorting, searching and editing the values in the tables is faster.

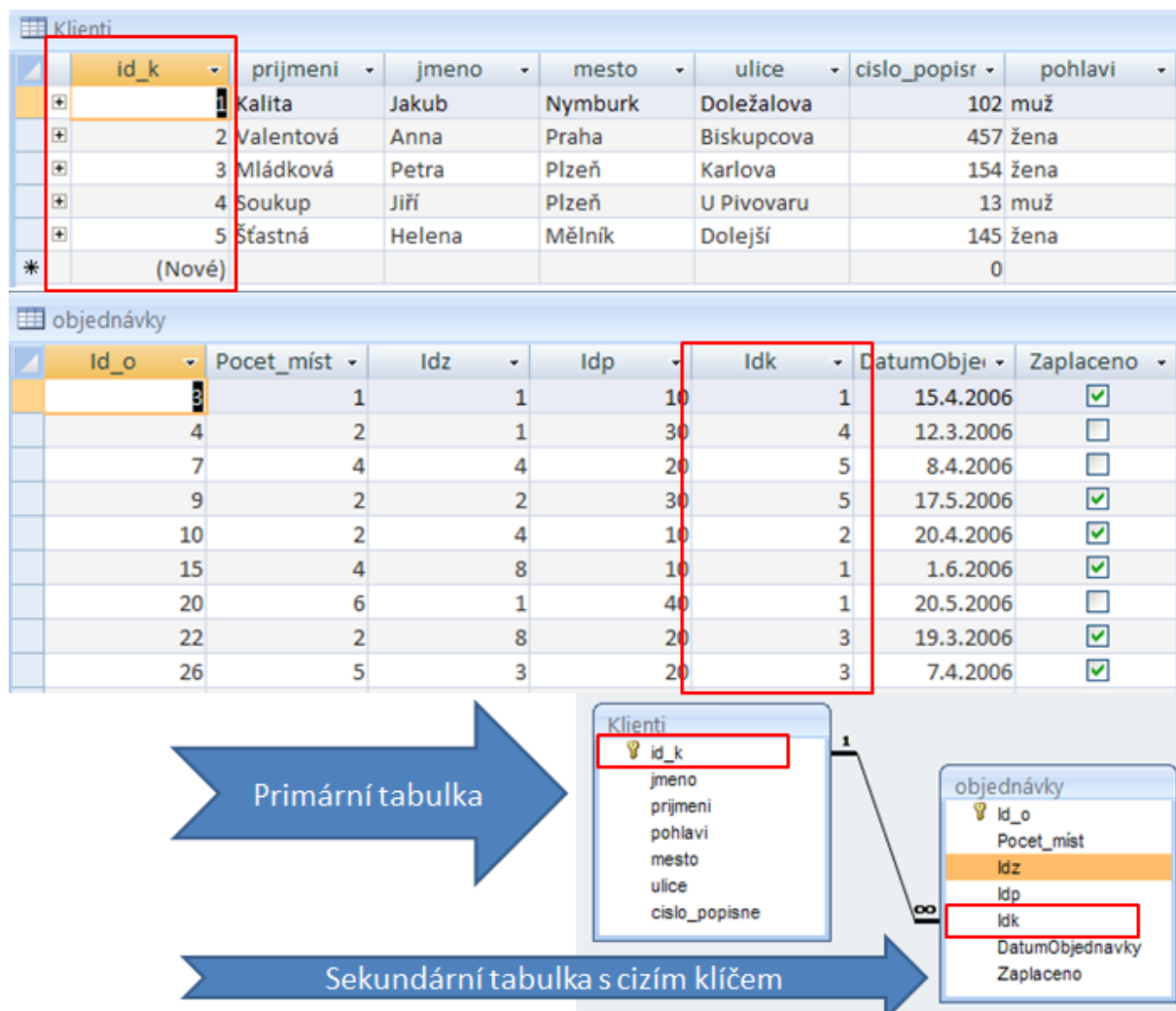
1.5. Relation

In a relational database, individual tables are linked by relations. The main purpose of relations between tables is to limit the occurrence of redundant data. Data should not be repeated in different tables within a single database. The relation is based on the link of the same values between the unique field (primary key) of one table and the corresponding field of another table. To create a relation between two tables, you must have a special field in each table. In one of these tables this is the primary key. We refer to such a table as the primary table. In the second table, we create a special field for relation purposes. We denote this field as a foreign key, it contains the values of the primary key from another table. A table containing a foreign key is referred to as a secondary table.

1.6. Primary and secondary table

The primary key field contains only unique values. The foreign key field may contain the same values. The primary and foreign key fields must contain the same values for correct relation creation. The figure below shows a primary table that contains individual customer records. The primary key is the first field that contains the customer number. This table is linked to the secondary table, which consists of records of customer orders. The foreign key in the secondary table is the field containing the number of the customer who placed the order. From the example above, we can see that one record in the primary

table corresponds to one or more records from the secondary table. In other words, one customer can place one or more orders. This is relation type 1: N.



1.7. Types of relations

We distinguish three basic types of relations:

Relation type 1: 1 (extraordinary) - one record of the primary table corresponds to just one record in the secondary table. For example, one person is assigned just one birth number.

Relationship type 1: N (most common) – One record in the primary table corresponds to one or more records in the secondary table. We have described this case on the above mentioned example with customers and orders.

Relation type M: N (very often) - one or more primary table records correspond to one or more records in the secondary table. We solve these relations using a joining table that we connect with the original tables using two relations of type 1: N.

1.8. Referential integrity

Referential integrity maintains relations between tables. It will not allow us to insert a record in the secondary table that does not have a corresponding record in the primary table. It also monitors the change of the foreign key values when the primary key is changed. Within referential integrity, it is possible to set the rules for deleting records.

1.9. Operation of the database

Professional databases contain a great deal of important information. Persons working with such a database can be divided into several groups:

- **database specialist** designs and creates professional databases,
- **user** inserts data, maintains data and retrieves information from the database,
- **database manager** provides users with access to certain data, is responsible for database recovery after a crash or a fatal error.

2. DATABASE MODELS

From the point of view of storing data and the links between them, we divide databases into the basic types:

2.1. Hierarchical data model

Data are arranged into a tree structure. Each record represents a node in the tree structure, the relations between the records are of the parent / child type. Finding data in a hierarchical database requires navigating through the records to the child, back to the parent, or to the side to the next offspring. The biggest disadvantages of the hierarchical layout are the complexity of inserting and deleting records and, in some cases, the unnatural organization of data.

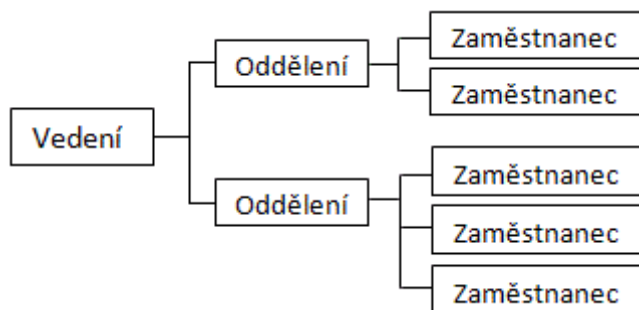


Fig. 1 - Hierarchical model

Legend: vedení – management, oddělení – department, zaměstnanec - employee

2.2. Network Data Model

The network data model is essentially a generalization of a hierarchical model complemented by multiple relations (sets). These sets interconnect records of the same or different type, the connection being made to one or more records. The access to interlinked records is straightforward without further searching; there are operations at disposal: finding the key record, moving the first child in the sub-set, moving the other child in the set, moving up from the child to its parent in another set. The disadvantage of network database is particularly rigidity and a difficult change in its structure.

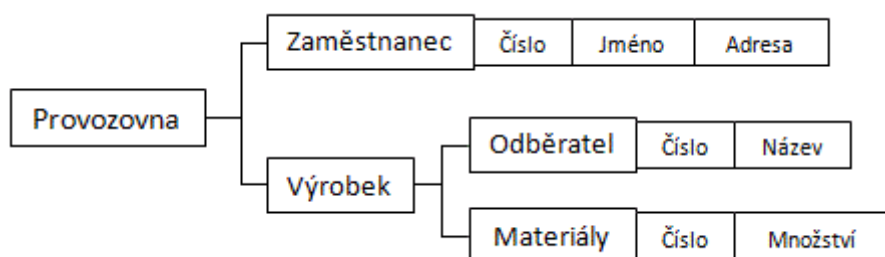


Fig. 2 - Network model

Legend: provozovna – premise, zaměstnanec – employee, výrobek – product, číslo – number, jméno – name, adresa – address, odběratel – customer, název – name, materiály – materials, množství – quantity

2.3. Relational data model

The relational database model is among the youngest and most used ones. At present, it is most often used by DBMS. The model has a simple structure, data are organized in tables that consist of rows and columns. All of these database operations are performed in these tables.

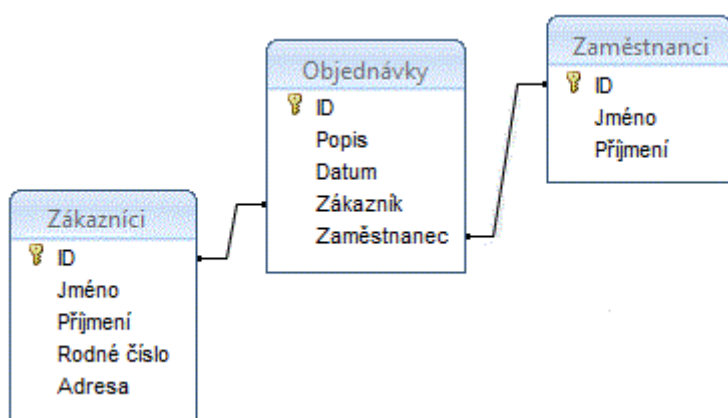


Fig. 3 - Relational model

Legend: zákazníci – customers, Jméno – name, příjmení – surname, rodné číslo – birth number, adresa – address, datum – date, objednávky – orders, popis – description, zaměstnanec – employee

The relational model database must meet the following two criteria:

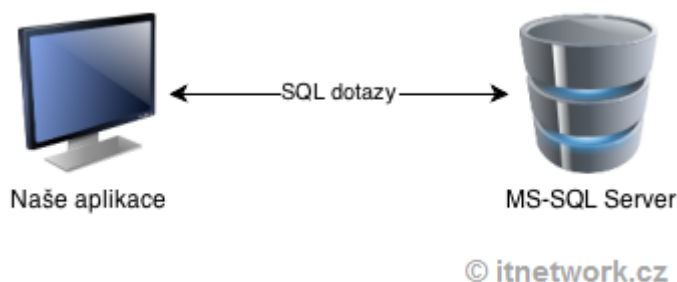
- The database is perceived by the user as a set of relations and nothing else.
- The selection, projection and connection operations are available in the relational DBMS without requiring explicitly predefined access paths to perform these operations.

In addition to relational databases, there are aforementioned object databases. It addresses the problem of object and relational incompatibility. They provide the same comfort as ORM, but internally do not need to convert data into tables, they are stored as objects. In theory, there is no performance or other reason for not replacing relational databases. In practice, however, they are almost unused and we can only hope it will change over time.

2.4. Attached application

You can use the access app when you need to read or change data in real time. Using DataReader, Command and Connection classes, we send SQL statements directly to SQL and get results.

The situation is illustrated in the picture:



2.5. Object databases

Database management systems (DBMS) are currently the most widely used tool for storing and manipulating commercial application data. They provide relatively simple management of large databases, integrity, efficient information search and processing, fault and failure tolerance, simultaneous multi-user access and other benefits.

The Relational Data Model - the most common model used in today's database systems - limits the structure and relations of the stored data to a mere set of tables above a pre-defined set of basic data types. The undisputed advantage of this model is its simplicity and, as a result, easy standardization and portability. However, the disadvantage is the difference of the real data schema from the internal database table model. All relations

between data can only be represented by tables, which, for an application with a more complex data model, leads to a number of tables interconnected by auxiliary links, the so-called keys. This results in a loss of clarity, the database becomes less manageable, and future changes in the application data model force programmers to intervene in their table representation. Relational databases are and will remain the primary means of managing commercial applications that are characterized by a large amount of data with a simple structure. However, many applications, such as design, multimedia, geographic systems, etc., need such a data model that will allow for better correspondence between complex real data and their representations in the database system. This model is an object data model. The object data model in database systems is based on the principles of object-oriented modeling and programming. However, it is further enriched with techniques of persistence, representation of relationships, questioning, transaction access, etc.

2.6. Fundamentals of Object Orientation. Objects and classes

An object-oriented model is based on the decomposition of real world information into so-called objects. An object is any (even structured) entity that is uniquely and independently identifiable within a certain context of the outside world. Thus, the object has an unambiguous identity, and every two or more similarly identical objects are mutually distinct. The object identity is determined by the object identifier (oid), which is generated by the system, unique, unchanging during the the object lifetime, hidden for both the programmer and the end user. Objects are characterized by means of classes. Class is an abstract description of the object; it determines the data components of the object and the operations (called methods) that can be performed on the object. Each object is an instance of a class, it is possible to instantiate a generally unlimited number of structurally identical objects from one class. in addition, we distinguish the class interface

2.7. Literals

In addition to objects, the concept of literal is introduced within an object-oriented model. A literal is a data entity of a particular data type that, unlike object, does not have its own identity. Literals typically occur as object attributes. The set of operations above the literal data type is fixed, it can not be changed. Objects and literals are related to the concept of mutability. Mutability is perceived as the ability to change data while preserving identity - in this sense, objects are variable because it is possible to change the values of their data components while retaining their original identity. In contrast, literals are not variable.

2.8. Operation

There are several basic types of operations on objects. Each object has one or more constructors. The purpose of the constructor is to initialize the object at the time of its creation. Another part of each object is a destructor. Destructor is called at the moment of object destruction and its purpose is to clean the object before it is removed. An important operation on objects is copying. There are so-called shallow and deep copies. In the case of a shallow copy, the attributes of the object are copied, but any references to other objects point to the same objects as the original object. In contrast, a deep copy will not only copy the attributes of the copied object, but also create copies of the objects to which the original object referred to. Other typical operations are methods for detecting and assigning attribute values, methods performing calculations and manipulation with object attributes, methods producing user output, etc.

An object data model is the data stored in the object structure. It is usually the object's interlayer between the code and the database where there are data the application works with and does not burden the DB server with queries.

Object-relational data model

The object-relational data model is a classical tabular database extended by **abstract data types** (ADT). ADT are user-defined types consisting of the basic data types of the database. Which violates 1NF?

- Object features are now implemented in all major DBMSs.
- Object types and their methods are stored together with data in the database, so the programmer does not have to create similar structures in each application.
- The programmer can access a set of objects as if they were one object.

- Objects can simply represent bindings where one entity consists of other entities (without the need to use bindings).
- Methods are run on the server - there is no inefficient data transmission over the network.

Object Data Types

It may include:

- data - attributes
- operations - methods
- The ORSRBD-specific is that we can see the data from both relational (records, operations) and object point of view (objects, methods).

Types of methods:

- Member methods - are called on a particular object.
- Static methods - are called on the data type.
- Constructor - The default constructor is defined for each object type.

3. INTEGRITY OF THE DATABASE

The integrity of the database means that the data stored are consistent towards the defined rules. Only data that meets the predefined criteria can be entered (for example, respecting the data type set for the given table column, or other limitations permissible for the given column). Integrated constraints serve to ensure integrity. These are tools that prevent incorrect data from being inserted or lost or damaged by existing records while working with the database. For example, it is possible to secure deleting of data that have already lost its meaning - for example, deleting the user, removing the rest of his records in other database tables.

3.1. Types of integrity constraints

Entity integrity constraint - Mandatory integrity constraints that ensure that the primary key of the table is complete (prevents the storage of data that would be the same as in any other row of the table).

Domain integrity constraints - Ensure adherence to the data types / domains defined for the database table columns.

Referential integrity constraints - deal with the relations of two tables, where their relations are determined by the binding of the primary and foreign keys.

Active referential integrity - Defines the activities that the database system performs when some rules are violated.

3.2. Maintaining integrity constraints

In principle, there are three ways to ensure that integrity constraints are respected.

1. Locating simple mechanisms for maintaining integrity constraints on the database server.
 - This is the best way to protect data.
 - However, the user usually has to wait longer for the system response and their portability to another database system cannot always be secured.
2. Locating the protection mechanisms to the client side.
 - the best option for the convenience and independence on the database system,

- need for control mechanisms for each operation can cause errors for applications, and multiple applications need to be repaired.
3. Separate server-side program modules.
- in modern database systems, so-called triggers are implemented for this purpose = independent procedures that can be initiated automatically before and after operations manipulating with data.
 - This way also enables the implementation of complex integrity constraints,
 - The disadvantages are the very limited possibility of transfer data to another database system on the server.

The ideal solution is the combination of the previous ones depending on the specific conditions. Checking integrity constraints is typically performed after each operation, which reduces server requirements. There is no need to record which checks are to be made later. However, more complex integrity constraints cannot always be verified, it is therefore possible to check compliance with the rules only after completing the entire transaction.

3.3. Relations between tables

Relations are used to bind data that are related and located in different database tables. In principle, there are four types of relationships.

- there is no connection between tables, so we do not define any relationship,
- 1: 1 (one record in the table corresponds to just one record in another database table and vice versa),
- 1: N (assigns one record of the table to multiple records of another table),
- This is the most used type of relation because it corresponds to many situations in real life.
- M: N (allows several records from one table to be assigned to several records from the second table).
- for practical reasons, this relationship is most often implemented by combining two relationships 1: N and 1: M which point to the auxiliary, so-called coupling table consisting of a combination of both keys used.

3.4. Normal forms

Normalization means the process of simplifying and optimizing the proposed database table structures. The main objective is to design database tables so that they contain a minimum number of redundant data. The correctness of the structure design can be evaluated by any of the following normal forms.

Zero Normal Form (0NF)

- The table contains at least one column (attribute) that can contain more kinds of values.

First Normal Form (1NF)

- Table columns cannot be further divided into parts bearing some information -> the elements must be atomic.
- One column does not contain composite values.

Second Normal Form (2NF)

- The table contains only columns that are dependent on the whole key.

Third Normal Form (3NF)

- The table is in the third normal form if there are no dependencies between non-primary columns.

Fourth Normal Form (4NF)

- The columns in the table describe only one fact or one connection.

Fifth Normal Form (5NF)

- by adding any new column, the table would split into multiple tables

3.5. Database integrity

The database integrity means that the database complies with the specified rules - integrity constraints. These integrity constraints are part of the definition of the database, and the database management system is responsible for their fulfillment.

Integrity constraints may refer to individual values entered into the database fields (for example, a mark from a subject must range from 1 to 5), or it may be a condition for combining values in some fields of a record (for example, the date of birth must not be

later than the date of death). Integrity constraints may also apply to a whole set of records of a given type - this may be a requirement for the uniqueness of the values of a given field or combination of fields within the entire set of records of that type that occur in the database (for example, the ID number in the person records).

Frequently used integrity constraints in relational databases are referential integrity. It is a requirement for a record field to contain a reference to another record somewhere in the database; such a referenced record actually must exist, so that such a link does not lead to "empty" and not a so-called database orphan.

Integrity constraints

It should be ensured that only records corresponding to real-world relationships (such as the age attribute should not get negative values) get into the database. To handle such cases, integrity constraints are used to determine a range of values that a specific attribute can take. Integrity constraints specify what limitations and internal relationships the database data must meet. Relations that meet the integrity constraints are permissible.

Relational database system

- must enable efficient management and analysis of the stored data,
- has to fulfill the following three basic functions:
 - enable to define the data type,
 - enable to work with data,
 - the system should include the means to perform the required activities with data – e.g. sorting data, filtering, calculating, managing data management,
- In particular, the system checks especially the access to the data, i.e. who is authorized to access the data and who can update it.
 - The system controls data sharing among multiple users.

4. RELATIONAL DATABASE MODEL

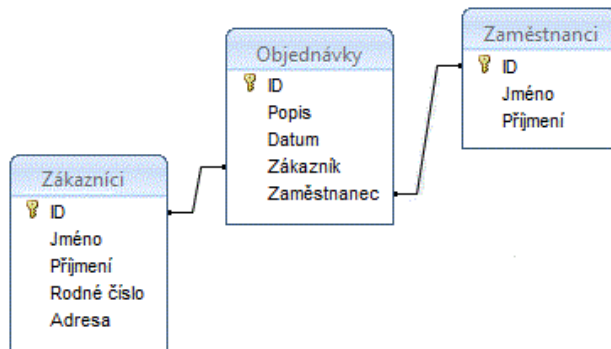


Fig. 3 - Relational model

The relational model database must meet the following two criteria:

- The database is perceived by the user as a set of relations and nothing else.
- At least, selection, projection and connection is possible in the relational DBMS without requiring explicitly predefined access paths to perform these operations.

4.1. 12 rules for relational DBMS:

1. Information rule:

All information in the relational database is expressed explicitly at the logical level in a single way - by the values in the tables.

2. The security rule:

All data in the relational database are guaranteed to be accessible by combining the table name with the primary key values and the name of the column.

3. Systematic processing of zero values:

Zero values are fully supported by the relational DBMS to represent information that is not defined, regardless of the data type.

4. Dynamic on-line catalog based on relational model:

The description of the database is expressed at the logical level in the same way as customer data, so an Authorized User can apply the same relational language to his query as a user when working with data.

5. A comprehensive data sub-language:

The relational system can support several languages and different modes used in terminal operation. However, there must be at least one command language with a well-defined syntax that supports data definition, view definition, interactive and program data manipulation, integrity constraints, authorized database access, transaction commands, and so on.

6. View creation rule:

All views that are theoretically possible are also system-configurable.

7. Insertion, creation and deletion capability:

Ability to maintain relational rules for both basic and derived relations is maintained not only when viewing data, but also by data penetrating, adding, and deleting operations.

8. Physical data independence:

Application programs are independent of the physical data structure.

9. Logical data independence:

Application programs are independent of changes in the logical structure of the database file.

10. Integral independence:

Integral constraints must be defined by relational database resources or its language and must be able to be stored in the catalog and not in the application program.

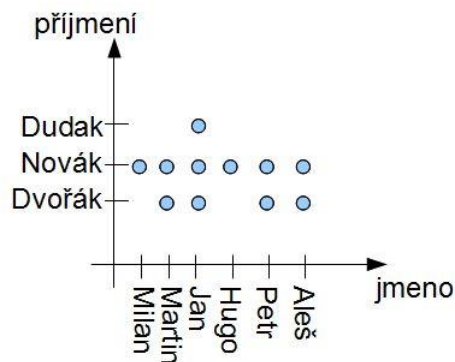
11. Independence of distribution:

Relational DBMS must be able to implement on other computer architectures.

12. Database access rule:

If the relational system has a low level language, then this level can not be used to create integrity constraints, and it is necessary to use a higher level relational language.

4.2. Relational Data Model



A relational data model is a way of retaining data in tables. It is called Relational because the table is defined over a relation.

Relation is basically a table. It is defined as a subset of the Cartesian product of the domains.

The relation in the figure on the right is therefore a subset of the set of $\{\text{Dudák, Novák, Dvořák}\} \times \{\text{Milan, Martin, Jan, ..., Aleš}\}$

Unlike a mathematical relation, the database one changes over time (by adding and removing relation elements). In addition to basic set operations, there is a selection operation - row selection and projection - column selection for a database relation.

atributy	
jméno	příjmení
Jan	Dvořák
Milan	Novák
Martin	Dvořák
Jan	Dudák
Hugo	Novák
...	...

entita

A domain is a set of all the values that an attribute may acquire, in other words, the range of attribute values. In practice, the domain is given by integrity constraints (IC).

* Note Figure: Accepting Attribute Domain is a set of $\{\text{Dudák, Novák, Dvořák}\}$. **Attribute** is an entity property. In terms of table, it is a column.

The relational schema can be seen as a structure of the table (attributes and domains).

Example for a table (relation) Teacher: Attributes: ID, name, surname, function, office

Domains:

- D1 - Three letters from last name, three digits of the order Numbers
- D2 - Name calendar
- D3 - set of surnames
- D4 - set of functions (assistant, scientist, teacher, ...)
- D5 - A101, A102, ... A160

Relational scheme: Teacher (ID, name, address, function, office)

Relation: Teacher = {(nov001, lukas, novak, scientist, A135), (kom123, jan, komensky, teacher, A111)}

It is defined as $R(A, f)$, where A is a set of attributes (A_1, A_2, \dots, A_n) and the function $f(A_i) = D_i$ assigns domain to attribute.

4.3. Relational Data Model Properties

From the definition of relation the following table properties result:

- Column homogeneity (domain elements)
- Each value (the value of the attribute in the column) is an atomic entry
- The order of rows and columns does not matter (they are sets of elements / attributes)
- Each row of a table is uniquely identifiable by the values of one or more attributes (primary key)

4.4. Relationships in the relational model

Generally, bindings in the relational model are implemented using another relation (table). This is the so-called coupling table. It contains the relation attributes (tables involved in the bindings) that uniquely identify their entities - the primary keys. If the table contains an attribute that serves as the primary key in a different table, it contains a foreign key. The coupling table therefore contains foreign keys.

In the figure below, there is the Teacher relation with Primary id Key and the Subject Relation with Primary cp Key. In order to express the relationship Teacher TEACHES The subject a new TEACHES Learning relation was created, which contains two foreign keys (idu referring to the teacher and cp referring to the entity of the subject). Now, we can link the teacher with the subjects using the coupling table, according to who teaches what.

And why wasn't a subject attribute added to the Teacher table? Simply because the teacher can teach more subjects. Between the teacher and the subject is M: N relationship. So even if we added Teacher attribute to the Subject table, this relationship would not be solved (the subject can be taught by more teachers). For a 1: N relationship, it would be possible, and it is performed in practice. So the coupling table is only necessary for the implementation of the M: N relations.

Učitel			Učí		Předmět	
idu	jméno	příjmení	idu	cp	cp	nazev
dvo01	Jan	Dvořák	dvo01	3	1	matematika
kov01	Marie	Kovářová	dvo01	1	2	anglický j.
kov02	Martin	Kovadlina	kov01	2	3	fyzika
chy01	Jana	Chtrá	chy01	1	4	biologie
mal01	Libuše	Malinová	mal01	5	5	český j.

5. SQL BASICS

5.1. Introduction to SQL (description and properties)

SQL - Structured Query Language - is a general tool for manipulating, managing, and organizing data stored in a computer database. It is primarily intended for users, although in many ways it is also used by application creators. It is adaptable to any environment.

The name of this tool is brief, but not accurate. SQL is not just a query language, it can also define data, i.e. table structure, fill columns of data table (field) with data, and define relations and organization among data items. It also allows data access control, i.e., granting and removing access at different levels, thereby protecting data from accidental or intentional destruction, unauthorized reading, or manipulation, and it also enables shared data usage and smooth operation when more users access data at a time.

SQL is a specialized programming language that is used in an appropriate environment, either user-friendly or interactively for immediate task-solving (most frequently queries), or its commands are translated into the host language. However, SQL is not a fully-fledged programming language, because in most implementations there are no control program constructs and other required elements that each general programming language should contain. SQL is a standardized tool for working with relational databases. It is not a database system but a differently integrated part of a database management system.

SQL is primarily an interactive query language - it allows you to get answers to very complicated queries almost immediately. It is a non-procedural tool with a data set access and it is a standardized language, it is comprehensible because it understands data in the form of tables, which is easy to understand for users. It works with relational databases in which the user views data as a set of interlaced tables. Each table represents a data set that is arranged in rows (records) and columns (items). The user refers to the data value as an element in the matrix.

In the majority of cases, the result of a task described in SQL is a data set from one or more tables, a so-called result table, which is not always the final product. It can serve as a set of input data for further processing, e.g. for printing labels or drawing a charts, etc.

SQL can serve as "common language," especially when operating in networks where different database products are used.

The SQL language is also used to create previews (queries). SQL views allow you to create different views of table structures and data for different users. Every user sees only the data they are supposed to see. The data are seen by the user as a simple table, although in fact the data come from different tables. The data displayed in the views are dynamic, that is, if the data in the tables (database files) change, the data that display the preview will also change and a vice versa. When working with an update preview (a special type of preview that allows not only data to be seen but also to be added and updated) and the data change, the changes will be reflected in the appropriate tables (database files).

A key term in SQL is a command. Each command begins with a keyword. The word indicates what activity the order is performing. The keyword is followed by one or more optional clauses that specify the nature of the activity being performed, or specify the data with which the order is to work.

Each clause begins with a key word, such as FROM or WHERE. Some clauses are mandatory, others are optional. Each SQL implementation uses its own clauses, in addition to the standard clauses given by the ANSI / ISO conventions, sometimes the function of standard clauses is more or less different or different.

SQL object names contain 1-18 characters in the standard, the first character of which must be a letter, and the name must not contain spaces or punctuation marks. When naming, it is sometimes necessary to qualify the name if the names are the same, and it is not clear what object the name refers to. Name qualification is done using the '.' (Dot) qualifier. Often qualifications are used with table names:

`<Owner> . <Table_name>`

or when qualifying columns, which is very common in database systems:

`<Tablename> . <Columnname>` , generally the qualifiers are also allowed in SQL.

Multilevel:

`<Owner> . <Table_name> . <Column_name>`

5.2. SQL queries

An introduction to a statements in SQL databases.

Introduction to SQL commands

The basics are to introduce basic SQL statements in simple examples. The information can then be used when programming in Visual Basic, Delphi, Web site programming. The basic introduction to what the database is is mentioned in the a Introduction article. If you already know the database theory, you can see how to create a database in Microsoft Access.

List of SQL statements

The most important commands are: SELECT, INSERT, DELETE, CREATE, FROM, WHERE and many others. But first we will make a simple table on which we will introduce the commands (to make it clearer what the command is doing).

Create a test table

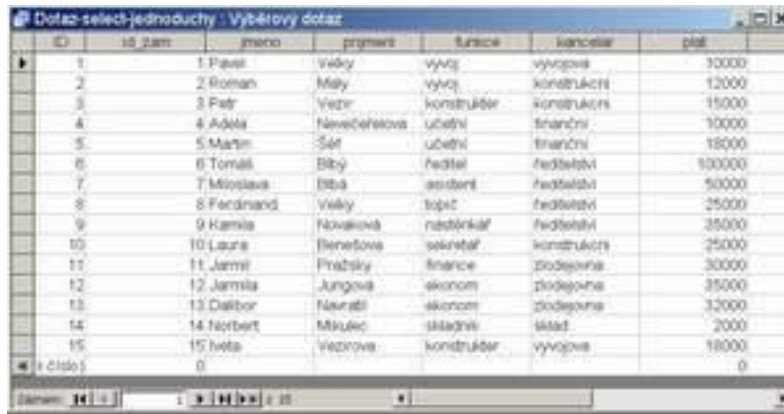


ID	id_zam	jmeno	prijmeni	funkce	kancelar	plat	vek
1	1	Pavel	Velky	vyvoj	vyvojova	10000	20
2	2	Roman	Maly	vyvoj	konstrukci	12000	25
3	3	Petr	Vezir	konstrukter	konstrukci	15000	35
4	4	Adela	Nevecefelova	ucetni	financi	10000	20
5	5	Martin	Sej	ucetni	financi	18000	23
6	6	Tomas	Bibj	reditel	reditelstvi	100000	35
7	7	Miloslava	Bibj	asistent	reditelstvi	50000	16
8	8	Ferdinand	Velky	topi	reditelstvi	25000	66
9	9	Kamila	Novakova	nastenkaf	reditelstvi	35000	25
10	10	Laura	Benesova	sekretaj	konstrukci	25000	55
11	11	Jarmil	Prazsky	finance	zlodejovna	30000	36
12	12	Jarmila	Jungova	ekonom	zlodejovna	35000	28
13	13	Dalibor	Navratil	ekonom	zlodejovna	32000	19
14	14	Norbert	Mikulec	skladnik	sklad	2000	48
15	15	Iveta	Vezirova	konstrukter	vyvojova	18000	47
* (číslo)	0					0	0

In order to clarify the terms of SQL language, it's a good idea to test it on some simple tables. Once they are fully understood, we can put them on a 10,000-line table. This table can be created in MySQL, FoxPro, Access or some other database program.

SELECT command

An important command. We use the created table and test the application for this command. The simplest form of the statement is this:



ID	ID Zam	Jmeno	Priznani	Funkce	Kancelar	Plat
1	1	Pavel	velky	vyvoj	vyvojova	10000
2	2	Roman	Maly	vyvoj	konstrukce	12000
3	3	Patr	Vezpr	konstrukter	konstrukce	15000
4	4	Adela	Havocelkova	ucetni	financni	10000
5	5	Marin	Sat	ucetni	financni	18000
6	6	Tomáš	Bibý	reditel	reditelsti	100000
7	7	Miroslav	Biba	asistent	reditelsti	50000
8	8	Ferdinand	Velky	topict	reditelsti	25000
9	9	Kamil	Novakova	nastnikaf	reditelsti	35000
10	10	Lucie	Benetova	sekretaf	konstrukce	25000
11	11	Jamie	Praslsky	finance	podajona	30000
12	12	Jamila	Jungova	ekonom	podajona	35000
13	13	Dalbor	Neurati	ekonom	podajona	32000
14	14	Herbert	Mikulic	skladnik	sklad	2000
15	15	Iveta	Vezprava	konstrukter	vyvojova	18000

```
SELECT *  
FROM employees;
```

Shows the contents of the entire table "Employees"



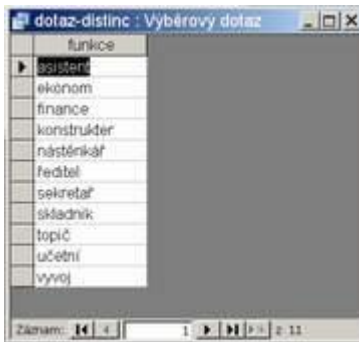
funkce
vyvoj
vyvoj
konstrukter
ucetni
ucetni
reditel
asistent
topict
nastnikaf
sekretaf
finance
ekonom
ekonom
skladnik
konstrukter

SELECT the name

```
SELECT the name  
FROM employees;
```


FROM employees;

Shows the contents of the "Name" column from the Employees table



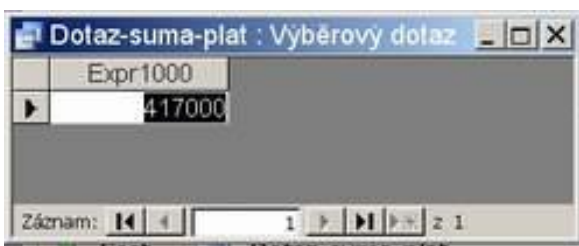
```
SELECT DISTINCT function
FROM employees;
```

Shows the contents of the "function" column in the Employees table, but displays only items without duplicates (so even if there are 4 developers listed only one will show)



```
SELECT AVG(plat)
FROM zamestanci;

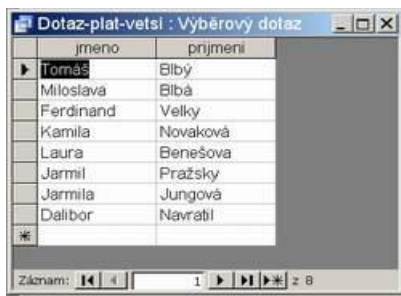
SELECT SUM (flat)
FROM employees;
```



AVG (column) calculates the average from the given column, SUM (column) calculates the sum of the values in the given column, or we can perform calculations in the query itself.

```
SELECT SUM (PLAT + 500)
FROM employees;
WHERE
```

When you have 10,000 lines you will want to limit the calculations to some criteria. This can be accomplished by WHERE. Together with some mathematical symbols (<,>,<=,>=, ..)



jmeno	prijmeni
Tomáš	Blbý
Miloslava	Blbá
Ferdinand	Velký
Kamila	Nováková
Laura	Benešová
Jarmil	Pražský
Jarmila	Jungová
Dalibor	Navrátil

```
SELECT name
FROM employees
WHERE pay > 20000;
```

It shows names of employees whose salary exceeds 20,000 CZK

Common simple query syntax

SELECT the list of columns from the WHERE table of the WHERE search terms to ORDER by the sorting columns.

The column list is either a comma-separated list of columns, or a * character to select all columns. The column name is either the column name only or the column-name.class_name. You can also use a table. * to select all columns from the table. The table can be any of:

- Relations (real database table)
- Result of another SELECT query
- View
- result of the JOIN operator (virtual table).

Search term is a condition that must be met for each record that is written in query results. If we want to write a record, first it must exist in the source table. Therefore, WHERE is a limiting condition. If it is not given, all the rows in the table are written. Sorting columns is a list of comma-separated columns according to which the resulting table will be sorted. The first column is the primary sorting criterion, the second column is a secondary sort - if you cannot decide according to the first column.

Example 1

Enter: Select all Radka and Tomas from the database.

Solution 1

```
SELECT name, surname, nickname FROM person
WHERE name = 'Radek' OR name = 'Tomas'
```

We can use common logical operators AND and OR, in most databases it can also be written as && and ||. You can use common = to compare strings, but you need to take a few things into account. Depending on the server settings, the letter size does or does not matter. The original (and therefore more frequent) default setting is that the letter size does not matter. In Akela.mendelu.cz server the letter size matters. Additionally, depending on the collation server (and database) setting, diacritics does or does not matter. In Akela.mendelu.cz diacritics matters, so the name must be written exactly as it is stored in the database.

Solution 2

```
SELECT name, surname, nickname FROM persons WHERE name IN ('Radek', 'Tomas')
```

An alternative to the previous query is to use the IN operator. IN is a set operator that tests whether the value is in the specified set, so for each row the value of the name column is tested if it is in the set ('Radek', 'Tomas'). The set can be defined either by a direct enumeration of values or by another SQL query.

Example 2

Task: Select all the people who live in Prague from the database.

Solution - step 1

```
SELECT id_address, FROM FROM WHERE address mesto = 'Praha'
```

In the first step we find all the addresses that are in Prague. It is now appropriate to recall the database schema from which we find that there is an `id_address` column in the `person` table that is related to the `id_address` column in the `address` table. The hypothesis is that if we select all the persons whose address is in the result of the above query we will get exactly those persons whose address is in Prague.

Solution - step 2

```
SELECT name, receive, nickname FROM people
WHERE id_addresses IN (

SELECT DISTINCT id_address FROM WHERE address
mesto = 'Praha' )
```

Now there are two `SELECT` constructs in the query. The second select is the so-called sub-query, and it is a slightly modified query created in the first step. The first adjustment is that it now only selects the `id_address` column. This is crucial, given that the parent SQL query looks for `id_addresses`, so the sub query has to return only the `id_addresses`. In addition, the `IN` operator only works with a scalar set and therefore a sub-query (to define a set for the `IN` operator) must always return just one column. If you use multiple columns in the sub query, the DB server will receive an error: `ERROR: subquery has too many columns`. The second change is to add the `DISTINCT` keyword to ensure that the return values are unique (see the following example). In this particular case, it is unnecessary (because the `id_address` is always unique), but because `IN` can behave very strangely with a sub-query in which the values are repeated (then it is not a set) as well as a precautionary universal measure `DISTINCT` is good.

6.SQL - MORE COMPLEX QUERIES

Exercises - SQL queries with multiple tables

6.1. Example 1

Task: write a SQL query that selects all people from the database who have an ICQ number; Select name, surname and ICQ number of the person; Sort by last name in ascending order.

Alternative 1

The easiest - the value of the `id_type_contact = 1` corresponds to the ICQ type in the `contact_type` table. The JOIN operator always works with two tables. The coupling condition is always the equality of values in some columns. The join condition must contain both tables that are in the JOIN section. The join condition must contain columns that link the tables. There is no reference to the contacts table in the person table. The contacts table is a single reference to the person table - the `id_object` column. INNER JOIN only selects from the table those records that meet the connection condition - that is, only those who have a contact (but because the search term is specified in the query, in this particular case it is also possible to use LEFT and RIGHT JOIN, but it is only a coincidence).

```
SELECT person.name, person.surname, contact.contact FROM
      INNER JOIN contacts
            ON contacts.id_osoby = person.id_osoby
WHERE contacts.id_types_contact = '1'
ORDER BY surname ASC
```

Alternative 2

It is the same as the previous one but uses USING. If the join condition is the equality of the equally named columns (`id_id`), its input can be simplified.

```
SELECT person.name, person.surname, contact.contact FROM
      Persons JOIN contacts USING (id_persons)
WHERE contacts.id_types_contact = '1'
ORDER BY surname ASC
```

Alternative 3

A better one - selects contacts by contact type name, so it does not rely on any `id_type_contact` value. The contact type name is in the `contact_type` table. It is important to think that the result of joining tables is a table and that the JOIN operator always works with two tables.

```
SELECT person.name, person.name, contact.contact FROM
    (INNER JOIN contacts
        ON contacts.id_osoby = person.id_osoby)
    INNER JOIN contact_types
        ON contact_types.id_type_contact =
            contacts.id_types_contact
WHERE contact_name.nazev = 'icq'
ORDER BY by ASC
```

SELECT is a command that can have many other words, it can combine several tables into each other, etc. But the full base of the syntax is: SELECT, then the list of columns we want to get, or aggregate or other functions FROM and the table name from which we want Date get, WHERE, and the expression with the limitations that must apply to the rows you want. In our case, we want the column of the city to contain exactly the value of "Chomutov".

With the aggregate function it is for example:- we want to find out how many rows, that is people are from Chomutov:

```
SELECT COUNT (*) FROM lide WHERE mesto = "Chomutov";
```

It will return one line in one column and it will contain number 2.

The conditions can be much more complicated with boolean AND and OR operators, brackets and SQL functions, the list of which is also in the documentation. Finding people from Chomutov over the age of 30 would be as follows:

```
SELECT * FROM lide WHERE mesto = "Chomutov" AND age> 30;
```

An asterisk means "all columns".

Linking other tables to a SELECT query is done using JOIN. This topic requires more than one table and some relationship between them. For example, the table people, where each person has a unique id, and then a table of ideas with a person_id column and an idea. In the person_id column, numbers would be people who invented the idea, and in the idea column there would be the text of the idea. Then we could make a query that would list all the ideas and add to each one a column with the name of the person. But details go beyond this introduction:

```
SELECT napady.napad, lide.jmeno FROM napady
LEFT JOIN lide ON lide.id = napady.clovek_id;
```

The result will be a table of ideas where the first column will be the idea and the second the name of the person who invented the idea.

When you insert "DELETE" instead of the "SELECT columns," we have a query for deleting rows. It works just like SELECT (there is a WHERE condition), but instead of getting the results, it deletes the lines corresponding to the condition. To delete Adam, just enter:

```
DELETE FROM lide WHERE name = "Adam";
```

Another one from the most frequently used SQL queries is UPDATE. This will edit the row. If Catherine becomes one year older, we will update it as follows:

```
UPDATE people SET age = 30 WHERE name = "Catherine";
```

But we can also use mathematical expression and references to existing column values, that is, to increase the age column by one can be done as follows:

```
UPDATE people SET age = age + 1 WHERE name = "Catherine";
```

These are all of the most commonly used SQL queries (clauses). Individual queries may have more words for more complex and more precise queries. For their understanding, however, it is necessary to study more individual documentation or to look for a specific problem. For example, when using the aggregate function in SELECT, we get the result of aggregating the entire table. But if we want for example to get temperature averages in

specific years and we have temperatures in months, we have to use GROUP BY and when we type in google group by years datetime mysql, we find that we need DATETIME YEAR:

```
GROUP BY YEAR (record_date)
```

In addition to GROUP BY, select has also ORDER BY command, where you specify which columns to use to sort the results and whether to sort them in descending or ascending order. Sometimes the word order is important. For SELECT it is again in the documentation and we can see that first we have to have GROUP BY ... and then ORDER BY

7. GRAPH DATABASES

Graph is a data structure consisting of vertices and edges. The graph database is a data storage and processing system in the form of a graph. In many cases, domain modeling using graph is very natural - such domains include human relationships, relation between genes and proteins, mobile networks, distribution networks of different kinds, neural networks, or ecological networks that capture interactions between organisms.

Graph databases often include different systems that manage data in the form of graphs. Based on such a definition, it would be theoretically possible to see relational databases, or their superstructures as graph databases. Relational databases, however, do not allow the effective storage and querying of graph data.

7.1. Relational databases

Moving through the graph in a relational database requires joins, and it is therefore inefficient for deep passing. One join using index requires a logarithmic time $O(\log n)$, without the index it is $O(n \log n)$.

For example, human relationships can be modeled using two tables: Man and Relationship. For each passage through the relationship between two people, we generally need one join. Traditional relational databases are optimized for join units. As the number of join moves to tens, relational databases are getting into trouble despite the use of indexes.

7.2. True graph databases

For the purposes of this article, true graph databases are such databases that require only a constant time $O(1)$ for the passage of the graph edge.

True graph databases, like other NoSQL databases, store data in a denormalized (already "joined") form. For true graph databases, you have to pay especially for writing, while querying (reading) is cheaper.

An example of a true graph database is Neo4j, which has been under development for more than a decade and is sufficiently sophisticated to be used for production, or, for example, for the newer OrientDB system.

Examples of databases that are able to store and query graph data but not necessarily efficiently browse the graphs are most triplestores (Sesame, Jena, Virtuoso) or FlockDB (a Twitter project).

As in the case of most NoSQL databases, there is no single query language for graph databases. However, many graph databases implement Gremlin language developed in collaboration with Neo4j authors.

Transition from the relational world to the world of graphs requires a shift in thinking and viewing data. Although graphs are often much more intuitive than spreadsheets, I've seen persistent mistakes at people who start with graph modeling. In this article we look at one of the most common mistakes - modeling two-way relationships, and finally we will show a real example in which we will continue.

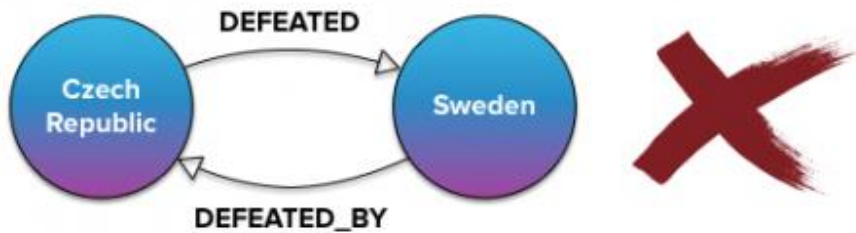
7.3. One-way relationships

Relationships in Neo4j must be of some type that gives the relationship its semantic meaning and also has a determined direction. This direction expresses meaning among entities. In other words, without determined direction, the relationship is ambiguous. .

For example, the following graph shows that the Czech Republic defeated (DEFEATED) Sweden in an ice hockey match. If the direction of the relationship changed, the Swedes would be much happier. Without the direction, we have no clue who the winner is, and that is why the relationship is ambiguous.



Note that from the existence of this relationship it results relationship in the opposite direction, as showed in the graph below. This is a very common case. Another example: Pulp Fiction was directed (DIRECTED) by Quentin Tarantino also means that Quentin Tarantino is the director of Pulp Fiction (IS_DIRECTOR_OF). And this could result in many pair relationships.

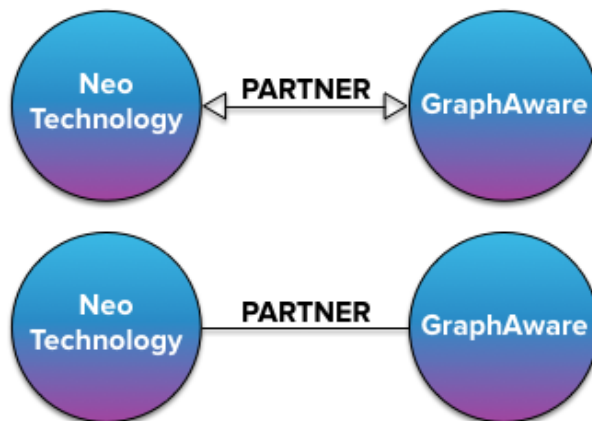


It is this mistake that people often make in graph modeling in the Neo4j and create bi-directional relationships. Since one relationship expresses the second one (symmetric relation), it is uneconomical both in terms of space and moving through the graph (traversing). Neo4j allows you to traverse the relationship in both directions, so also in the opposite direction of the edge. Moreover, due to the way Neo4j stores data, the speed of passing does not depend on its direction.

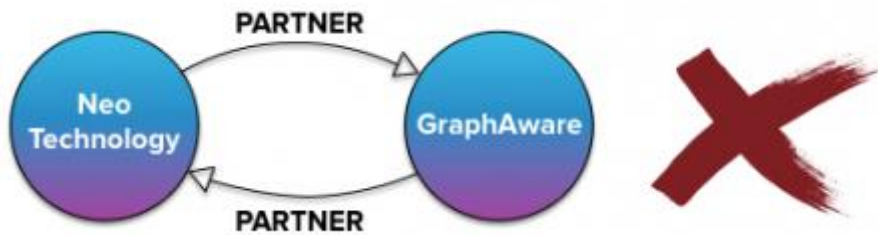
7.4. Two-way relationships

Some relationships are naturally two-way. A classic example is Facebook or a friendship. The relationship is mutual - when you're somebody's friend, so he's a friend of yours (probably). Depending on how we see the graph model, we might say that it is bidirectional or nonoriented.

An example of GraphAware and NeoTechnology are partner companies. Since it is a relationship, we could model a two-way or non-oriented relationship.



But this is not possible in Neo4j - each relationship must have an initial and a final node. Beginners often tend to work with the following model, which has exactly the same problem as the aforementioned ice hockey model with redundancy.



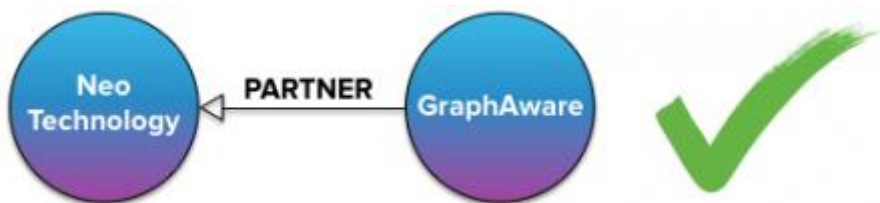
The Neo4j API allows developers to completely ignore the directivity of the relationships when writing queries if they want to. For example, in Cypher, the search for all partner companies with NeoTechnology could be as follows:

```
MATCH (neo) - [: PARTNER] - (partner)
```

The result would be the same as merging the results from these two different queries:

```
MATCH (neo) - [: PARTNER] -> (partner) and MATCH (neo)
<- [: PARTNER]
```

Therefore, the right (or at least the most efficient) way of modeling partner relationships is using a single PARTNER relationship with determining any direction.



7.5. Summary

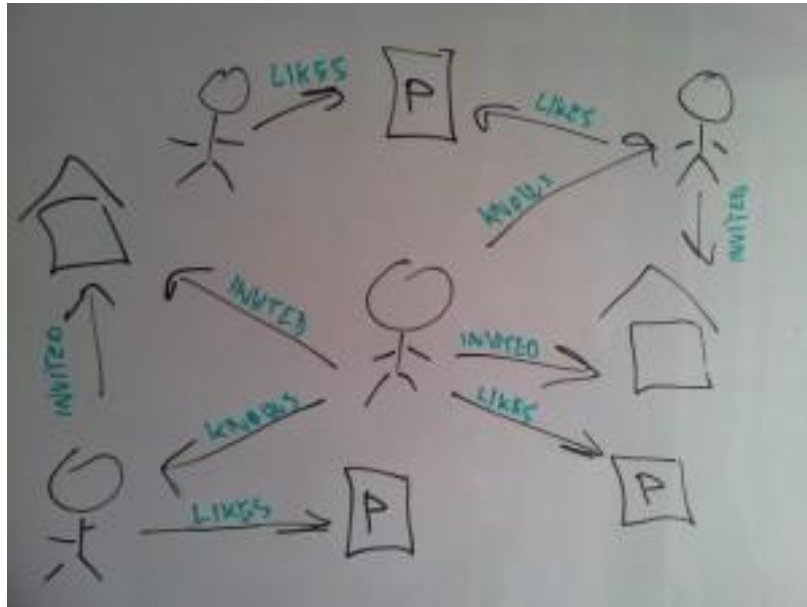
Relationships in Neo4j can be traversed in both directions at the same speed. In addition, directivity can be completely ignored. For this reason, it is not necessary to create two different relationships between entities, if the edges of the opposite direction have the same meaning.

7.6. Graphic Database Modeling Methodology

In the following articles, we will deal with the design, implementation and testing of the Neo4j graph database on the BestPartyToday project (this is a worldwide partylist that offers detailed statistics on events). Currently, the MySQL database uses tens of millions of records, so it will be necessary to move data from a relational database store to a graph one. However, now we will only model the graph database from selected tables and attributes of the current structure of the MySQL database.

The conceptual design of the graph database is referred to as Whiteboard friendliness. In order to quickly reflect all the ideas and insights that were raised during the design brainstorming, a flipchart will do. The resulting layout is very simple to present to other project groups (retail department, account managers, sales managers, etc.) who do not have technical education. Understanding the design is no longer an obstacle for them, as the layout can be easily adapted to real life.

7.7. Implementation of the conceptual model



The photograph of the final design of graph database for the BestPartyToday projet shows five most important tables of the current relational database, represented by icons depicting the nodes and prolationships represented by directional arrows. Through the entities (nodes and relationships) we will traverse the graph and obtain the required data that will be processed and displayed to the user of the application. Their meaning is described in the table below.

8. NOSQL DATABASE

NoSQL database as the main topic of this work are given most attention to First, it is important to specify what the NoSQL database is. Next chapter is dedicated to CAP Theorem, which explains that one of the reasons why there are so many NoSQL products is that each of them can only provide certain properties at the expense of others, and then the user must decide which combination of properties is the most important for them and then to choose the product accordingly. The lecture focuses also on scalability, i.e. the capability to increase the performance of the database server, either by improving the server using more efficient hardware or by spreading the database to multiple servers. Besides scalability, we also need to mention sharding, a technique that divides databases into multiple parts that can work independently and faster than a whole. Next, the best-known NoSQL products will be mentioned, from which three representatives will be selected to make a detailed description. As already mentioned in the introduction, NoSQL databases are those that do not address the relational way of data control. They are not primarily based on tables and do not use SQL to work with data. Their realm is high search optimization at the expense of small functionality, which is often limited to a simple data storage. These deficiencies, compared to SQL, are offset by scalability and performance in certain data models. NoSQL is suitable for storing large volumes of data that do not need to be interrelated. Structured data are, however, allowed. For transactions, NoSQL databases cannot offer full ACID support, but only eventual consistency. This is a situation where, due to neglecting ACID, we gain more accessibility and better scalability. This approach without "strong consistency" is suitable for NoSQL, which often apply it too. NoSQL has a distributed architecture that is fault-tolerant. Some data may be located on several servers, so the failure of one can be tolerated. These databases usually scale horizontally and manage large volumes of data for which performance is more important than consistency.

NoSQL is literally a combination of two words: No and SQL. It's a technology that stands against SQL. The abbreviation may be confusing and there is no consensus on its meaning. However, the most widespread opinion is that "Not only SQL" is an acronym. Whatever the abbreviation means, NoSQL is now the umbrella name for all databases that do not follow RDBMS (relational database management system) but use their own way, often associated with large volumes of data.

What is it?

First of all, it is important to specify what NoSQL is NOT - it is definitely NO SQL, that is, rejecting relational databases. NoSQL means Not Only SQL, meaning that the relational database is not the only way to solve the persistence, that there are alternatives that may in some cases be more appropriate.

For larger applications, we can see that a relational database is suitable for some data, while for other NoSQL databases are more suitable. This pragmatic approach when more databases are used in one project is often referred to as polyglot persistence.

NoSQL databases have originated as a solution to real problems - it is not a crazy idea of academics living at a distance from everyday reality, but who have been torn apart from reality but a highly practical matter. The emergence of many of NoSQL databases is related to projects that have had to deal with a huge amount of data - Facebook (Cassandra), Google (BigTable), Amazon (Dynamo), LinkedIn (Voldemort).

The use of a NoSQL database can be meaningful even if we have less data (which a relational databases would be able to process) - because some NoSQL databases offer a data model that is more suitable for our application.

But back to the introductory question. NoSQL Database is data persistence software that is an alternative to classical relational databases and it definitely has its place on market.

9. TRANSACTION

Transaction is a logical unit of work consisting of one or more SQL statements that are atomic in terms of recovering from SQL transaction errors. SQL transaction automatically begins with the SQL initialization command (SELECT, INSERT). Changes made by one transaction are not visible to other competing transactions unless the transaction is not completed.

There are four ways to complete a transaction:

- COMMIT completes the transaction successfully and the changes are permanently recorded
- ROLLBACK interrupts the transaction and all changes are cancelled, the database returns to the pre-transaction state
- within the program - if the program ends successfully, SQL transaction
- within the program - if the program ends with an error, the SQL transaction is cancelled

SQL Access Database Management defines two commands to access table:

- GRANT and REVOKE. The security mechanism is based on - Authorization Identifiers - Ownership – Privileges.

9.1. Solution: Transactions

Transactions can be understood as a sequence of database operations that meet ACID:

1. **Atomicity** - Within a transaction, all operations or neither of them are executed.
2. **Consistency** - The database is consistent before and after the transaction. All integrity constraints must be met.
3. **Isolation** - Individual transactions are isolated. Changes made during the processing of the transaction are not visible in other transactions.
4. **Durability** - After successful completion of the transaction, the data are stored and cannot be lost.

9.2. Transactions in SQL

In SQL, there are the following commands to deal with transactions:

BEGIN	startet eine Transaktion ... befiehlt innerhalb einer Transaktion etwas ...
COMMIT	speichert und beendet die Transaktion
BEGIN	... befiehlt innerhalb einer Transaktion etwas ...
ROLLBACK	kehrt Veränderungen an einer Transaktion um und unterbindet diese

Isolation levels

ACID requirements are relatively strong and time-consuming. Therefore, these conditions can be reduced by so-called isolation levels. There are 4 levels (from the weakest to the strongest):

1. READ UNCOMMITTED

The transaction can read data recorded by another transaction without this other transaction being completed with COMMIT. Reading such data is referred to as dirty read. Such data may be inconsistent (time goes down from top to bottom):

2. READ COMMITTED

At this isolation level, dirty read cannot occur. The data we read are always recorded by a transaction that has been successfully terminated with COMMIT. However, non-repeatable read can occur. If we read data more times within a transaction, it can happen that within repeated readings not the same result appears every time. It is possible that between the individual data readings within a transaction, a parallel transaction has been completed successfully, which had recorded / edited / deleted some data.

3. REPEATABLE READ

At this level, it is ensured that there is no non-repeatable read. The data we read once cannot be changed during repeated reading. However, phantom read can occur. If we read data in a transaction more times, it may happen that the repeated reading result will contain new rows that did not appear in the result of the previous reading. It is possible that between the individual data readings within a transaction, a parallel transaction has been completed successfully that has recorded new data.

4. SERIALIZABLE

ACID enforces no above mentioned phenomenon can occur.

10. PROCEDURES AND FUNCTIONS, TRIGGERS AND SEQUENCES

Functions, methods and procedures - what's the difference between them? Functions, methods, and procedures are fairly similar to each other - it's the DEFINED sequence of commands, a part of the program that can be repeatedly called for from other parts of the program.

10.1. Formal view

From the formal point of view, we follow the nomenclature of the given programming language.

- Function is a term of functional programming, which occurs in languages like JavaScript or Haskell.
- Functions in Object-Oriented Programming (OOP) are referred to as methods, where the data structures and functional code are linked to objects. These methods can be found in Java, Smalltalk, etc.
- Procedure is a term known from procedural languages. They can be found, e.g. in Pascal and in some database systems - so-called stored procedures.

However, we can see functions, methods and procedures from a factual point of view (according to their properties and meaning) and deviate from the terminology of a particular programming language.

Function

Functions in programming are very close to the functions as we understand them in mathematics. Function has a certain definition field (type of input parameters) and a certain range of values (type of return value).

Example of a simple JavaScript function:

```
Function sum (a, b) {return a + b};
```

In object languages like Java, there are methods, not functions, but it does not prevent us from writing functions in them:

```
Public static int sum (int a, int b) {return a + b};
```

Although `sum ()` is formally a method (public and static), it is actually a function and the class here only plays the role of the namespace (together with the name of the package) and we do not create its instances (or not). This is a design pattern Library Class.

In Java, such functions are for example in the `java.lang.Math` class. The following function returns the larger of two numbers:

```
Int x = java.lang.Math.max (a, b);
```

Procedure

The procedure is a special case of a function - it does NOT have a return value and does not have to have even input parameters. They are often used in batch processing - for example, every hour we call the procedure that processes the orders that have accumulated in the database and passes them to another system.

An example of a very simple procedure in PostgreSQL that sums the values of columns a and b to the unaddressed rows and stores the result in column c:

```
CREATE OR REPLACE FUNCTION add ()  
RETURNS void AS  
$ BODY $  
UPDATE table of summaries  
SET c = a + b  
WHERE c is NULL  
$ BODY $  
LANGUAGE sql VOLATILE;
```

This procedure is written in SQL - besides, we can write in `plpgsql`, which allows us to create very complex procedures, or we can use C language or some other.

The above mentioned procedure (formally function) has no input parameters or return value, it is simply a sequence of commands of a language we have named and saved.

It is interesting that the procedures can have parameters - not only the classic input parameters but also the output (or input / output) ones. The output parameter procedure use is similar to function - although it does not have a classical return value. For example, we can pass some data to the procedure, and it will modify it.

We can also simulate these procedures in Java.

A structure:

```
Public class Person {public String name;  
Public String Surname; Public String fullname};
```

A procedure:

```
Public class StoredProcedures {public void reportNameName  
(Person o) {o.celName = o.name + " " + oName}};
```

We will pass the data (the person) to the procedure and make the required adjustment. However, if you are programming this way in Java, you are probably doing something wrong and do deprive yourself of the main benefits of PPE.

Note: beware of value transfer - if you assign a new person to the variable within the Java procedure, this change will not be reflected in the procedure - the original person remains untouched. But we can work with the attributes of the original person - in this case, the code of the procedure and the code that called it "sees" the same person's instance.

10.2. Method

Methods are part of the class and (if they are not static) are closely related to the given class instance (object). Therefore, they are not functions working with any global variables and data, but they have access to variables of a given instance (in this case, individuals).

The previous example can be overwritten "more objectively" as follows:

```
Public class Person {public String name; Public  
String Surname; Public String getCeléName ()  
{return name + " " + surname}};
```

We do not need a class with stored procedures and move the required functionality closer to the data (to the Person class). Note that we do not have to save even the calculated value fullName - we will calculate it in case somebody needs it, i.e. when somebody calls the getNameName () method.

What is a trigger?

Trigger is a procedure that is automatically triggered when something happens. It can be defined for both DML (modification) and DDL (creation) operations. Plus, it can be said whether it is supposed to be done before or after the operation. (BEFORE, AFTER) It is interesting that the trigger can be named as a table (however, it is not recommended) and that any number of triggers can be defined for the same table and event. However, it is not possible to determine the order the triggers will be called, and the trigger must not depend on each other. In triggers, recursive calling must not be used. Within trigger, it is possible to perform any SQL statements – INSERT, UPDATE, DELETE. However, if you want to use SELECT, it must be used in combination with a key word INTO that ensures storing the data into the local variables.

II. ANALYTICAL TOOLS - OLAP

II.1. OLAP - Information Under Control

Which customers bring 80% of the profit? How has the success of retaining customers changed after launching the loyalty program? How do different business events (campaigns, introduction of new products, changes in company processes, etc.) and the market (competition activities, changes in economic conditions, etc.) affect sales fluctuations? What is the result of comparison with the previous period? These and similar questions are asked by every realistic manager. Every day, they try to understand the behavior of the company based on the analysis of the data they have at disposal. In well-functioning companies, analysts are included in selected departments, who only take on this role and provide reasoned recommendations to managers. All of these people need to have prepared infrastructure and tools which would enable them to do this activity for the company.

II.2. What and how it simplifies work

The path to successful data analysis begins when they are being prepared. The first important step is to gather data from enterprise systems and arrange them into a form suitable for analysis and reporting. During this preparation, it is necessary to ensure that the various data and the quality of the data are correctly aligned. We must always document the links to the source systems so that the results are always credible and verifiable - in the case handling errors that occur in the original data. The resulting data are stored in a relational database where they are available to the extent necessary for operational analyses. Ready-to-use data are used to generate detailed reports on company performance and simpler analyses based on detailed data. However, in the following simplified example, we will require a different type of information: A nationwide company sells different products. The manager divides individual branches into regions. The category of products sold has three levels - categories, subcategories, specific product. The company has a logical definition that sales must follow a plan. The real fulfillment of the plan is then given by the sum of all products and all branches.

II.3. How to get this information?

For these needs, advanced analyses are referred to as multidimensional. The manager can monitor measurable business parameters in contexts, angles, and various detail levels. Such analysis requires OLAP (on-line analytical processing) technology to ensure that data are stored and pre-calculated in such a way that subsequent queries last for a reasonable amount of time. OLAP technology brings you the ability to work with data on a summary level, identify the problem or an interesting area here, and proceed to the level of detail that is of interest to our decision making. A typical area where OLAP benefits are demonstrated is sales analysis.



Fig. 1: OLAP in Oracle Business Intelligence – Discoverer

Through OLAP tools, the manager can control how reality evolves against the plan. If the plan is not fulfilled, the trend curve shows this information in time, and the end of the year may not end with a fiasco. Summarized numbers can be easily broken down by a level below, then you can see a summary fulfillment of the plan in each region. An analysis can show that the problem lies in only one of them, the other regions fulfill the plan. By further drilling, they get to the branch level, identifying five of those who are significantly

behind in the problematic region. The system will allow splitting to the level of categories and subcategories to the level of individual products that are not sold successfully at that location. Based on such information, it is possible to design and implement corrective actions to improve the situation. The example shows that OLAP data are actually stored in a tree structure - a cube. Here, for each depth level of this tree, values for the subtree are calculated. Such a hierarchical structure is definable on most enterprise data. It simplifies analysis when evaluating them. The price for this simplification is the time and place needed to calculate and save the tree. However, if the cube is filled, it can accurately respond to very complicated queries, in particular to make comparisons of time slots limited by complex conditions. A typical complicated query is, "Which ten customers saw the highest increase in turnover with my business compared to last year, in the districts that rank among 20 % of my least profitable ones?" The result can then be used to provide extra care to these customers. With OLAP, it is possible to query data from many angles - many dimensions - and arbitrarily parameterize these angles. As we have shown, it is not appropriate to store data in relational form for these types of queries.

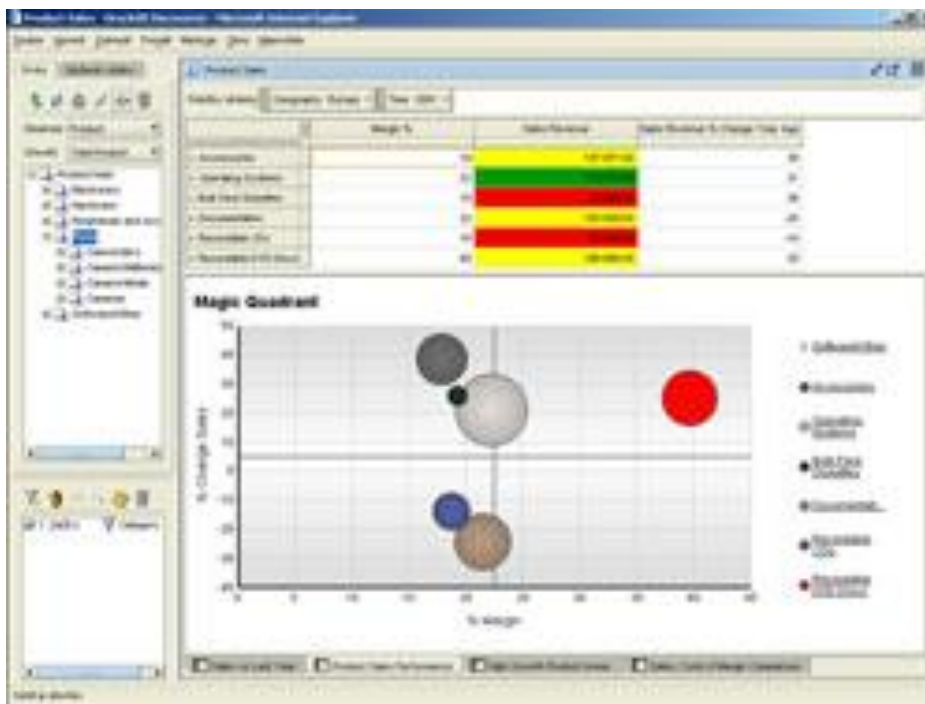


Fig. Figure 2: Sales Analysis in Oracle Business Intelligence – Discoverer

II.4. Analytical tools

Up to now, we have described ways and technologies of storing data. To work with them (performing analyses) requires appropriate tools. Choosing the right analytical tool is a key issue for effective information gathering. In particular, it is important to take into account the experience of analysts who are already working with the resources at their disposal - very often, tools from the Microsoft Office family are used. Therefore, it is important that the solution chosen allows sufficient co-operation with the tools people are used to - this leads to lower costs of adopting a more efficient way to work with information. At the same time, security of access to information, automated distribution of reports, high-quality graphic output for presentation, or integration with company intranet must also be ensured.

The most important thing are people

Despite the great number of technology, we must not forget people. One of the most valuable people the company can get is experienced analysts because only an experienced analyst can get essential information from the huge amount of data and formulate recommendations for management.

II.5. OLAP cube

OLAP cube (online analytical processing) is a method of organizing data that extends a two-dimensional table arrangement so that each data dimension is stored in one axis of the cube. This overcomes some limitations of relational databases.

Arranging data into cube vectors allows to access them from different points of view (dimensions) in the same way. This way there is no need for difficult combining of RDBMS tables, which is demanding for the system performance. However, physical data storage in cubes does not allow rapid editing; in such a case the entire cube needs to be reprocessed. The cube is made up of values that are categorized into dimensions. The structure is implemented by relational tables in the star or the snowflake scheme. It is typically a parent-child structure where parent elements represent consolidation of children and they themselves can be aggregated into their own parent elements.

Basic data cube operations

For analytical needs, the following data cube operations are performed to enable processing and data projection to facilitate their understanding.

Slicing a Cube: Limiting one or more dimensions to a subset of one element.

Cubing: Limiting one or more dimensions to a subset of two or more elements.

Roll up and drill down: Navigation through the data hierarchy up and down.

Pivoting: Rotating the cube in order to gain a different perspective on data relationships.

Aggregation: Consolidation according to relationships defined by formulas.

12. SPECIFICS OF DATABASE SYSTEMS. ACCESS TECHNOLOGY TO DATA- BASES. GEOGRAPHIC INFORMATION SYSTEMS.

The database (or data base) is a file system with a fixed records structure. These files are interconnected using keys. In the broader sense, database includes software resources that allow you to manipulate and access the stored data. This software is called database management system (DBMS) in Czech scientific literature. Database is normally perceived - depending on the context - as both stored data and software (DBMS).

12.1. Data Access Overview in ASP.NET Technology

Visual Studio 2010

Web applications commonly access data sources for storing and retrieving dynamic data. You can write data access code by using the classes in the System. Data namespace (commonly referred to as ADO.NET) and the System.Xml namespace. This approach was common in the previous versions of ASP.NET.

However, ASP.NET technology also allows data bindings to be done declaratively. In most cases, no code is required:

- Selecting and displaying data.
- Sorting, paging and caching of data.
- Updating, inserting and deleting data.
- Data filtering using runtime parameters.
- Creating master-detail scenarios using parameters.

ASP.NET includes several types of server controls that are a part of a declarative data-binding model, including data source controls, data binding controls, and an expanding query control. These controls handle the basic tasks that are required by the stateless site model to view and update the data on the ASP.NET Web site. Controls allow you to add data binding functionality to the page without having to understand the page life cycle details.

13. DATABASE OBJECTS

The term "database" is often simplified to what actually a database system (database engine) or a database management system is. It does not only include tables - they are only one of many so-called database objects (sometimes also database entities). More advanced database systems include:

- **Views**- SQL commands named and stored in the database system. You can select from them (apply the SELECT statement).
- **Indexes for each table.** Keys are defined above the individual columns of tables (one key may include more columns), and their function is to keep LUT tables at the columns over which they were defined, to avoid records duplicity or to ensure fulltext search.
- **Triggers** - a mechanism over individual rows of a table (or the table itself) that is called for after a change, deletion or addition of a line, or deletion of a row or deletion of a table and that performs a pre-programmed action (for example, checking the integrity of data, ...)
- **User-defined procedures and functions** - some database devices support storing of named pieces of code that perform a specific sequence of commands (procedures) in the database above the given tables, or return some result (user functions). They can have parameters that are mostly divided into input (IN), OUT (OUT) and I / O (INOUT).
- **Events** - procedures triggered on a specific (user-defined) date and time or repeatedly with a defined period. They can serve for maintenance, deleting temporary data, or reference integrity checking.
- **Forms** - Some database systems such as Microsoft Access allow users to create entry forms for visually friendly input. For example, the user can define the layout of each input field from a given table, labels, etc.
- **Reports** - similarly to forms, reports allow the user to define the layout with the fields of the given table, to which the current values are added. They are used for data output (printing, presentation, or plain view). Reports can be supplemented, for example, by filters that filter out only the desired records.
- **User Permissions** - better database systems offer the possibility to separate individual levels of access to other database objects for their users. There are tens of options, with a distinction to the individual types of commands that the user will or will not have permission to use.
- **Partitioning** - a way to divide the data in a table into multiple hard disks and thereby break down the load placed on it

- **Processes** - Database tools can give an overview of the processes that are currently using their services .
- **Variable settings** - typically dozens of variables that can be reshaped and thereby affect the behavior and performance of the database tool itself.
- **Collation** - MySQL has advanced options for setting up several dozens of character sets and comparisons, collectively called collation. The collation setting can be done on individual text columns, entire tables, and entire databases (with cascading inheritance). Collation also affects sorting, for example, the value `utf8_czech_ci` ensures correct sorting by Czech (ie including diacritics and including `ch`).
- **Visual E-R Schema** - (in MySQL `INFORMATION.SCHEMA`). Visual representation of relationships (s) of dependent fields (foreign keys) between tables.

13.1. Basic concepts of database technology

Introduction to database technologies

With the appearance of microcomputers and their entry into the life of each one of us, there seems to be an increasing interest in computer-processed data, not only in the sense of computing something, but above all in the sense of knowing something. The explanation is quite understandable considering that even the most sophisticated computer games are tiring over time, and that programming in Basic is a way of calculating the type of solution of a system of two equations of two unknowns.

One would like to organize and search for some of the information frequently used and changed while using it. Assuming appropriate means of storing such information in the form of data, there would be no problem to spend several evenings sitting at the computer and make, buy or exchange the relevant data himself with another similar enthusiast. An example might be a library, an office, flight ticket processing, a city museum, a hospital, a business,

The completed object of the type described can be called an information system (IS). The IS is a set of elements in the mutual information and process relationships (information processes) that process the data and ensure the communication of information between the elements. From a user's point of view, the IS should have such features so that manipulating with them (input of data, query formulation, use of application programs) is as easy as possible, and at the same time that its functions are fast and strong enough. Providing a flexible "empty" IS to a user today can be quite fast, but only with sufficient knowledge.

Our goal will be to show the so-called database processing technology. Database technology is a set of concepts, resources and techniques used to create information systems (IS). At the most basic level, we can visualize the IS architecture with the database as follows: Data is organized in a database (DB), controlled by a program package called Database Management System (DBMS). The database and the DBMS form the so-called database system (DBS). In our terminology, we can simply write $DBS = DB + DBMS$. IS uses data from DBS either directly or processes it with application programs.

13.2. Geographic information system

(GIS, Geographic Information System) is a geographic information system that allows you to store, manage and analyze spatial data.

Most real-world objects and phenomena occur at some place on the earth's surface (e.g. tree, house, river) or relate to a place on the earth (the citizen has a permanent residence somewhere, the product was manufactured in a particular factory). At the same time, these objects occur in the space together with many other objects and interact with each other. Therefore, knowing the location and spatial relationships between objects is very important and can play an important role in a number of human activities, from the design of the nuclear power plant to the design of the business network and the evaluation of its success.

In practice, this means that our data in the computer must include both, i.e. both the actual data on the object and its location. This data type is called geographic (or spatial) data and a computer system that allows us to store and use such data, is called the geographic information system, GIS.

Basic components of geographic information systems

Basic Components of Geographic Information Systems as Vít Voženílek in his book Geographical Information System I. The Concept, History, Basic Components describes them, are:

- Hardware
- Software
- Data
- Service staff

For the efficient operation of systems, their balance is essential. Hardware, or technical equipment, is the technical basis of geographic information systems. Software is a set of

programs executing all system operations. Data is a key element of any geographic information system. GIS is a real, actual system from the organizational structure point of view. Its operation is a collection of activities that ensure the functions of the system.

13.3. Geographic data

Dimensions of geo objects

The basic division of geo objects is a division by the number of dimensions. Real objects on the Earth's surface are always three-dimensional. They are, however, transformed (transformed) into the GIS by the required level of simplification.

- 0D geo objects – non-dimensional objects, points defined only by their location. An example can be a bus stop in a GIS modeling transport or a GSM transmitter in a GIS mobile carrier modeling signal coverage.
- 1D geo objects - one-dimensional object, line sections (edges, lines), finite length and zero area. 1D geoobjects are most often modeled by roads, rivers,
- 2D geo objects - two-dimensional, polygons (polygons), with finite circumference and finishing surface.
- 3D geo objects - three-dimensional object, geometric body. In GISs, they are used exceptionally, in specific cases. The third dimension is most often modeled in the GIS by means of the so-called digital terrain model (DMT, English DEM) by the "Surface" - linked topological surfaces (2.5D).

The basis of each GIS is geographic data. Geographic data is information about the Earth's surface and the objects on it. Data can be represented as a certain layer of information that is called a theme. Each topic represents a certain element (e.g. roads, lakes, towns, etc.). Geographic data can be organized in GIS by two basic models:

13.4. Data Collection

Getting data and moving information into the system takes a lot of time. There are a number of methods used to enter GIS data stored in digital format. Exit data printed on paper or PET film can be digitized or scanned to produce digital data. The digitizer produces vector data as points, lines, and polygons. Scanning maps in the form of raster that could be further processed to produce vector data.

Collected data can directly access GIS using a technique called Coordinate Geometry. Positions from the Global Navigation Satellite System (GNSS) can also be collected and subsequently imported into GIS. The current trend in data collection that allows users to use computers in the field with the ability to edit live data by connecting to a wireless network or editing without an Internet connection. This eliminates the need to import and update office data once it has been collected by fieldwork. This includes the ability to incorporate positions obtained by laser rangefinder. New technologies enable users to create maps as well as field analyzes, making projects more efficient and mapping more accurate. Remote-sensed data also plays an important role in collecting data and consists of sensors connected to the platform. Sensors include cameras, digital scanners and LIDAR, while platforms usually consist of aircraft and satellites. In England in the mid-1990s, hybrid balloons called Helikites first promoted the use of compact digital airborne cameras as Geo-Information Systems. Airplane software measuring 0.4 mm precision was used to link photographs with ground measurements. Helikites are cheap and collect more accurate data than planes. Recent development of miniature unmanned drones. For example, the Aeryon Scout drone was used to map a 50-acre plot with a sampling density of 1 inch (2.54 cm) in just 12 minutes.

13.5. Multidimensional cube

The basic building unit in the multidimensional database is cube (cube, data cube, multidimensional cube, hypercube). The cube in a multidimensional database consists of a set of dimensions and measures.

Cube dimensions are the categories against which we want to aggregate and analyze data. Dimensions arise from relational database tables. Typical dimensions in multidimensional databases are time, position, product. Dimensions can consist of a number of levels that further refine the data.

Cube measures are the quantitative data we want to analyze. As dimensions, measures are derived from relational database tables. Typical rates are sales, expense, prices, but almost every quantitative figure can be a measure of a multidimensional dice.

13.6. Saving data in the OLAP database

The source data from the relational database can be stored using one of three basic methods:

Data cubes contain aggregated data that are the basis of multidimensional analysis. Comprehensive multidimensional applications (such as informatory) contain data related to different contexts, but some dimensions can be shared. It is therefore more understandable to create a single data cube for one context than to use a data cube embracing all contexts. The data cube consists of only those dimensional attributes that share all of its numeric attributes. This means that dimensional attributes form the basis of a data cube. If we want to get information from more data cubes, these attributes can be linked at the level of one or more common dimensions. Navigation between data cubes is not seen by the user.

Dimension tables describe specific dimension properties. For each dimensional attribute, there may be no more than one dimension table. Unlike the original star model, in our model, dimension of data cubes can be without dimension tables. On a schematic level, the dimension table consists of attribute names, and at the level of the examples it is based on the values of these attributes.

In informatory, aggregate data must often be analyzed on the basis of complex criteria that depend on the properties of the dimensions. In order to specify these criteria, the information should be used in several dimension tables. It results from this that semantic relationships between dimension tables are created based on the shared values of some attributes in dimension tables.