



INTERNATIONAL EUROPEAN UNIVERSITY
Scientific and educational institute
"European School of Business"

Falovskyi O. O.
Nesterenko O. V.

BASICS OF DATABASE DESIGN AND USING

SECTION I

Tutorial

Kyiv - 2023

BBK 74.48
UDK 372.8:004.04
O75

Recommended for publication by the Academic Council of the International
European University (Protocol № 10 from 2022.12.29)

Reviewers::

I. V. Kazachkov, Doctor of Sciences (Phys.-Math.), Professor, International
European University

V. L. Shevchenko, Doctor of Sciences (Tech.), Professor, Institute of Software
Systems of the National Academy of Sciences of Ukraine

O. I. Shiyani, Candidate of Sciences (Phys.-Math.), Associate Professor, Vinnytsia
National Technical University

Falovskyi O. O., Nesterenko O. V. Basics of database design and using: Tutorial.
Section I. Kyiv: Tropea - 2023. 83 p.

This Tutorial is designed to support the study of the discipline "Databases" for students enrolled in undergraduate programs in the specialty "Software Engineering". In addition, students of higher education institutions studying the disciplines of computer science, information technology and systems analysis can use the Tutorial.

Database technologies, taking into account current trends in improving the management of enterprises and the capabilities of information and communication technologies, is a promising area of automation of management work and other areas of activity. The manual contains basic information about the information factors of creating databases in enterprises, the organization of databases and knowledge bases, the use of data models, the features of the architecture of database management systems. An overview of some database maintenance software products available on the market is provided.

In addition to students and teachers, this publication can be useful for managers and professionals in the field of economics, as well as for scientists who are interested in the practical problems of creating databases in the field of management.

ISBN 978-617-7894-79-6

© O.O. Falovskyi, 2023

© O.V. Nesterenko, 2023

© International European University, 2023

CONTENT

INTRODUCTION	6
1. DATABASE CONCEPT	11
1.1. The concept of data, information and database	11
1.2. Database organization technologies	17
1.3. Databases models	22
Test questions and tasks	26
2. THE CONCEPT OF DATA MODEL	28
2.1. Properties of relations	28
2.2. Normalization of relations.....	30
2.3. Relations operations.....	34
Test questions and tasks	41
3. SEMANTIC MODELING	42
3.1. Basic concepts of semantic modeling.....	42
3.2. Construction of semantic models	45
3.3. Diagrams of entity relationships.....	48
Test questions and tasks	51
4. INTRODUCTION TO DATABASE PROGRAMMING	52
4.1. The concept of SQL language.....	52
4.2. Data types in SQL	55
4.3. Transact-SQL.....	58
4.4. Special database objects.....	60
4.5. Development of user applications in the ‘client-server’ environment	63
Test questions and tasks	65

5. COMMERCIAL AND FREELY DISTRIBUTABLE DBMS	66
5.1. General characteristics of the DBMS market.....	66
5.2. Ms SQL Server.....	68
5.3. Freely distributable DBMS	71
5.4. NoSQL DBMS	77
Test questions and tasks	81
LITERATURE	82



LIST OF ABBREVIATIONS

AI	- Artificial Intelligence;
API	- Application Programming Interface;
BI	- Business Intelligence;
CGI	- Common Gateway Interface;
DB	- Database;
DBMS	- Database Management System;
DW	- Data Warehouse;
HMI	- Human Machine Interface;
IoT	- Internet of Things;
IS	- інформаційна система;
IT	- інформаційні технології;
ML	- Machine learning;
NIST	- National Institute of Standards and Technology (USA);
OLAP	- Online Analytical Processing;
OLTP	- Online Transaction Processing;
SQL	- Structured query language;



INTRODUCTION

The current stage of society development is closely connected with the rapid technological growth and openness of activity, which are becoming determinants of economic, science and education development. At the global level, the development of the *information society* in the direction of *digital transformation* in all spheres of activity has been proclaimed. Due to the development of the Internet and means of communication, the widespread use of *information and communication technologies* significantly increases the intensity of information exchange, and the main type of activity is information processing and generating new knowledge.



Information society – the concept of post-industrial society; a new historical phase of civilization, in which the main products of production are information and knowledge. Features that distinguish the information society are: increasing the role of information and knowledge in society; increasing the share of information products and services in gross domestic product; creation of a global information space that provides effective information interaction of people, their access to global information resources.

Information Technologies (IT), Information and Communication Technologies (ICT) – a set of methods, production processes and software and hardware integrated to collect, process, store, disseminate, display and use information of interest its users.

Digital transformation (DX) is the adoption of Information technology by a company. Common goals for its implementation are to improve efficiency, value or innovation.

In many countries, government regulations and documents provide for the widespread introduction of automation of information activities in various spheres of life in order to increase efficiency and

achieve a qualitatively new level in the management of enterprises (organizations, institutions) and regions and the country as a whole.

Determinants in achieving the efficiency of enterprises are management processes, an integral part of which are *information processes* and flows. Today, management tasks have become much more complex than ever. With the development of production technology, the scope of management functions and their complexity is growing rapidly. In these conditions, the achievement of management goals requires the creation of tools that allow for the storage and processing of large amounts of information. Therefore, since the early 70's of the twentieth century began to work intensively on the development of database tools, resulting in the creation and successful use of new information systems – *database management systems* (DBMS).

It should be noted that the historical path of developing the basics of storage systems begins with research on systems analysis, experience and methods of specialists and scientists of the early computerization era, who actualized the role of mathematical thinking and use of models in information processes.

Ukrainian experience in developing the basics and creating automated information systems is associated primarily with the name of the famous scientist Academician V.M. Glushkov. At his initiative and with the participation of the Kyiv Institute of Cybernetics headed by him, automated control systems were developed and implemented at many enterprises, as well as in non-manufacturing organizations.

Growing competition in the market, difficult business conditions and demanding consumer needs require constant modernization of business processes in enterprises, changes in basic approaches to management, methods of promotion and sale of goods and services. In these conditions, the capabilities of modern DBMSs allow companies to move to a completely new level of business development through the active introduction of innovative technologies.

The operation of any enterprise, its basic tactical and strategic plans are always related to information needs. That is why an active and effective information policy of the company is the key to success. Using information technology, specialists and managers can quickly exchange information, obtain up-to-date data on sellers, buyers, consumers, find

the necessary information in databases. The possibility of conducting a comprehensive analysis of data using mathematical methods and efficient algorithms is essential. All this helps to make informed management decisions.

Thus, among the main tasks facing students majoring in "Information Systems and Technologies", "Software Engineering", "Computer Science" is an important place to learn the basics of information processes, understanding the need for models and their choice, mastering the features of presentation and organization of data and knowledge, as well as gaining skills in the use of DBMS tools, knowledge of the architectures of such systems, the features of the user interface.

Therefore, this tutorial is designed to support the study of the discipline "Databases" for students enrolled in programs in the above specialties. The manual contains basic evidence about the information factors of management and activities at the enterprise, the organization of data, the use of data models on the features of the DBMS architecture. An overview of some DBMS software products available on the market is given. A feature of the manual is a significant amount of methodical material from the laboratory workshop, which should help students gain practical skills in creating and using databases.

In this regard, the composition of the manual provides for two parts – theoretical and practical, and mainly focused on the number of hours in the curriculum allocated to the discipline for classroom sessions. Therefore, due to these limitations, many issues remained outside the scope of the manual. Also, due to page limitations, the manual is presented in two separate editions – «Section I. Basic concepts of database systems» and «Section II. Laboratory workshop».

To get acquainted with this material, a list of relevant literature is recommended.

At the end of the units, there are control questions and tasks that correspond to the areas of independent work of students. They are aimed at enhancing cognitive activity, independent creative work and gaining practical skills. An important feature of the tasks is that they are aimed at the complex gradual creation of the student during the course of his own database project, the functional purpose of which he can get as a task for the course work.

INTRODUCTION

The text of the manual uses icons that facilitate the orientation and search for certain structural elements, namely:



- basic concepts and definitions;



- deserves special attention;



- examples;



- questions for self-control;



- tasks;



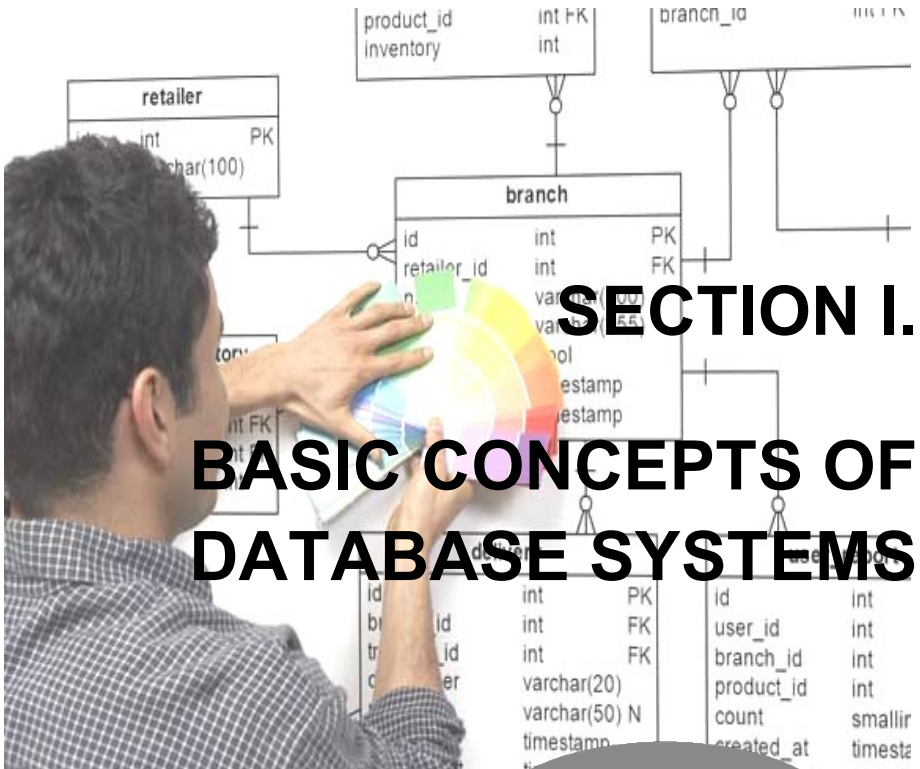
- tasks that require the use of a computer;



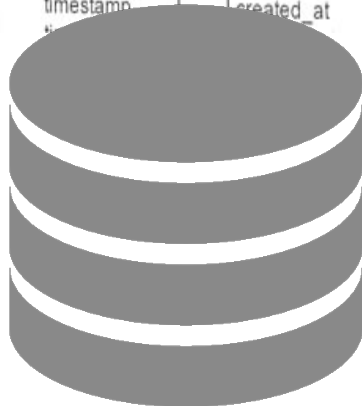
- tasks using Internet sources.

In preparing the manual used materials from leading companies providing solutions for database management - Microsoft, Oracle, etc.

Photo materials, background information are obtained from various open sources on the Internet, to which the authors express their respect and admiration.



SECTION I. BASIC CONCEPTS OF DATABASE SYSTEMS





1. DATABASE CONCEPT

1.1. The concept of data, information and database

The concept of information and data. DBMS, its functions. Database as a model of the subject area.

Data is an integral part of our world and surrounds us everywhere. Although the terms "data" and "information" have only recently been used, in fact these concepts have existed since ancient times. Most of the history of mankind is associated with the processing of data in one form or another (Fig. 1.1), for example, in the form of documents as data carriers. For storage of documents and other media, such as manuscripts, books, always used certain storage - shelves, cabinets, drawers. There, these media were usually placed in a certain order to make it easier for users to find the data they needed. Such repositories have become the prototype of modern databases (DB), which arose due to the need to store a huge amount of data processed by computers.

According to the well-known analytical company IDC (International Data Corporation), the amount of data in the world is estimated at 40 zetabytes (ZB). This means that each inhabitant of the Earth has more than 5,000 GB of data! To represent the amount of 40 ZB, analysts compared this value with the natural figure and concluded that humanity is dealing with the amount of data about 60 times greater than the amount of sand on the beaches across the Earth!



Fig. 1.1. Data (information) has always been an integral part of people's lives



Data is any signal received and processed either by a human using the senses or a technical device. In a broad sense, data are facts about objects and phenomena of the environment, their parameters, properties and state (numerical data, text, images, sounds, video segments). Data can be obtained from measurements, experiments, arithmetic and logic operations.

In other words, data is raw material that comes from data sources and is used to form *information* and *knowledge* based on them. To paraphrase the famous physicist Albert Einstein, we can say that everything is relative - except for the data, because they are absolute.

According to the international standard ISO / IEC 2382: 2015 "Data is a formalized presentation of information suitable for interpretation, transmission or processing with human participation or automatic means".

1. DATABASE CONCEPT

Thus, the relationship between the concepts of "data", "information", "knowledge" can be represented in Fig. 1.2.



Information is information that is perceived by information systems (living organisms, computers) in the process of data processing. **Knowledge** is a new value that emerges in the process of processing and analyzing information and is used for decision making.

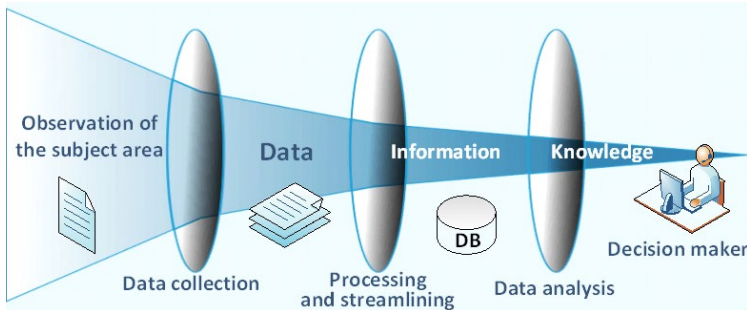


Fig. 1.2. The relationship between the concepts "data", "information", "knowledge"

An integral part of computer data processing and analysis are databases. The famous science-fiction writer Herbert Wells wrote back in 1940: "The vast, ever-increasing wealth of information and knowledge is scattered throughout the world today. This knowledge would probably be enough to solve all the enormous difficulties and problems of our time - but they are scattered and disorganized. We need to clear our minds in a kind of workshop where we can get, sort, summarize, assimilate, explain and compare information, knowledge and ideas".



A characteristic feature of our time has been the phenomenon of a significant increase in the amount of information processed, the mass flow of information flows and the influx of related problems of the "information explosion". The most important components of these processes are computerization and telecommunications.

Anticipating many technical inventions, Wells predicted the emergence of databases, those "workshops" that are present in all

information systems today. Any enterprise - large or small - today will not be able to work without a database, where the main value is concentrated - information (Fig. 1.3).

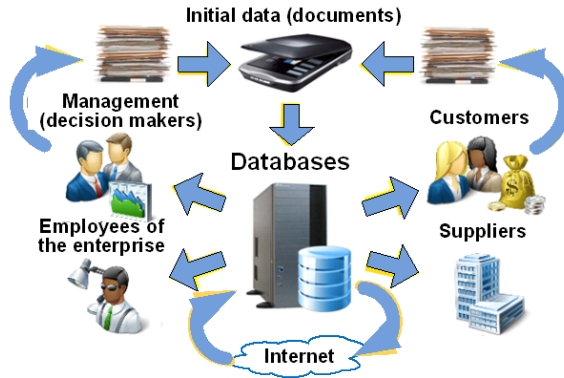


Fig. 1.3. Databases as part of information processes at the enterprise

So, what is a DB?



A **database** is a data storage system organized in a special (special) way. "Specially organized" means that the data is organized in a way that makes it easy to find and access data for one or more computer programs. Also, such data organization requires minimal data redundancy.

Databases are not only a form of data storage, but also a kind of information technology.

The main purpose of the database, in addition to saving data, to provide users with access to data, as well as means of extracting and modifying it. These functions are provided by special software - Database Management System (DBMS) (Fig. 1.4). DBMS is a shell through which a database is created by organizing the structure of the database and filling it with data. Thus, DBMS provides basic data management functions - Search, Create, Read, Update, Delete (SCRUD).



Database Management System – special software which controls the organization, storage, integrity, modification, reading and security of information in the database.

1. DATABASE CONCEPT

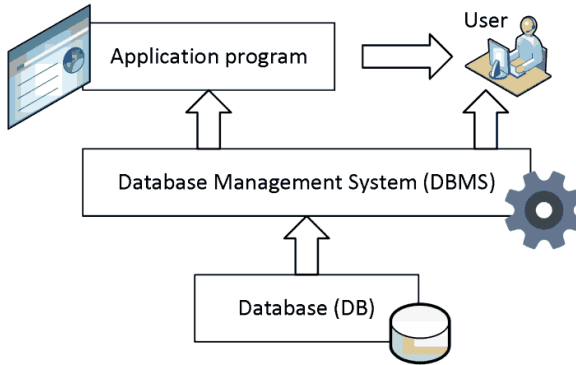


Fig. 1.4. Database management system in information processing

With regard to computer data processing, information is understood as a sequence of symbolic denotation (letters, numbers, encoded graphic images and sounds, etc.), which carries a semantic load and is displayed in a computer-readable form. To store data, they are usually represented by objects and their attributes (Table 1.1).



Object - record, fact, table row, etc.

Attribute - a property that characterizes the object. An attribute is also called a variable, table field, measurement, characteristic.

Table 1.1

Example of data presentation

	attributes				
	customer code	customer class	age	marital status	income
objects	01	1	18	singleton	12500
	02	1	22	married	10000
	03	2	30	singleton	7000
	04	1	32	married	12000
	05	2	24	divorced	9500
	06	1	25	married	6000
	07	2	19	singleton	8500



Information objects - items, processes, phenomena of tangible or intangible essence, considered in terms of their information properties.

Information process - the processes associated with certain operations on information.

The subject area is the part of the real world that is to be studied in order to organize management and, ultimately, automation. The subject area is represented by a large number of fragments, for example, the university it faculties, departments, groups, individual students. A large number of objects and processes that use objects, as well as a large number of users who are characterized by different views of the subject area characterize each fragment of the subject area.

The subject area is considered in the form of three representations: representation of the subject area in the form as it really exists, as it is perceived by the person (for example, the designer of a DB), and as it can be described by means of formal methods. That is, they say that we are dealing with reality, the idea of reality and the data that reflect (describe) this idea (Fig. 1.5).

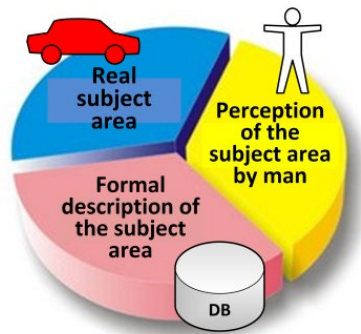


Fig. 1.5. DB as a model of the real world

Any database is an integral part of an information system (IS), which aims not only to store data, but also their processing to support specific decisions. Thus, when creating a database, there is usually a transition from the representation of a specific subject area to a specific implementation of the database by means of a specific database in a specific IS (Fig. 1.6).

1. DATABASE CONCEPT

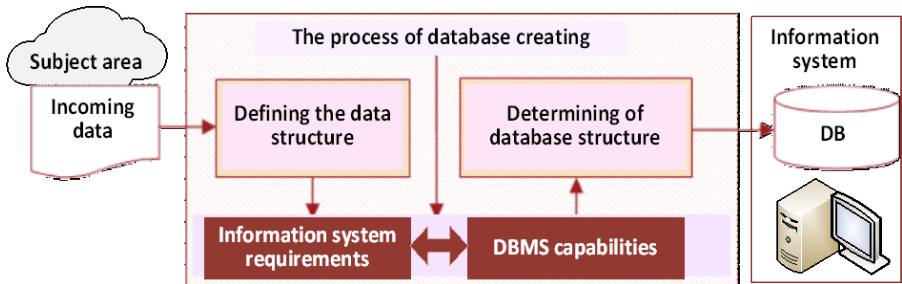


Fig. 1.6. The process of database creating



Computer system – a complex of software and hardware designed to process information

Automated system – organizational and technical system that implements information technology and integrates the computer system, physical environment, personnel and information processed

Information system - a complex of organizational, technical, software and information means combined into a single system for the purpose of collecting, storing, processing and issuing the necessary information designed to perform specified functions

Information and telecommunication system - a complex of information and telecommunication systems, which in the process of information processing act as a whole

1.2. Database organization technologies

The main components of the database system. Data warehouse. Knowledge base.

From the very beginning, database systems were *centralized*, in which the database, DBMS and application program that processes data from the database were located on one computer. Such systems are used to solve local tasks. With the advent of personal computers in the 1980s, such systems

became available to many small and medium-sized firms, enterprises and organizations, as well as individuals.

With the development of computer networks, it has become possible not only to organize access to the DB from another computer, but also to create a *distributed system* in which the database consists of several parts located on different machines.

Modern ISs are characterized by the use of *client-server* architecture. This term defines primarily the logical distribution of data processing functions.



Client – user application that retrieves data from DB. It is also called a frontend program.

Server – software that provides access to DB (backend).

The client-server technology assumes that, in addition to storing the database, the central computer (database server) provides the execution of the main amount of data processing. With this technology, a request to perform a data operation (for example, a regular selection) issued by a client (workstation) forces the server to search for data. The received data are transported on a network from the server to the client (fig. 1.7). A network bridging the gap between users and information resources.

In the client-server architecture, the database server is responsible for performing the main functions: working with database files, maintaining the integrity of links, backup, providing authorized access to data, logging operations. Of course, the server is also responsible for executing user requests to select and modify data. Client applications, which are the source of these requests, run on personal computers on the network.

In addition to two-tier client-server architectures, three-tier architectures are now popular (Fig. 1.8). They have another link – the *application server*, which is all data processing (calculations, analysis, scheduling, etc.). Only the results of data processing are displayed on the PC of the user named "thin client".

Applications running in the client-server architecture access the server and use client programming interfaces either (APIs) or one of the universal data access mechanisms. Typically, when using such application architecture, the database server is also responsible for controlling the preservation of procedures and other database objects.

1. DATABASE CONCEPT

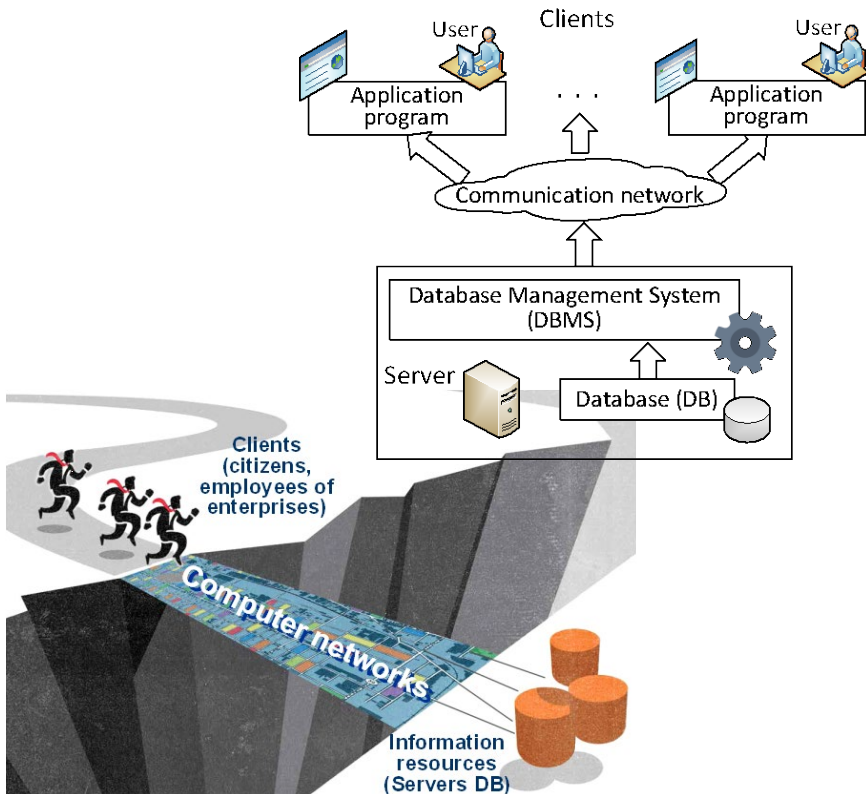


Fig. 1.7. Client-server architecture



Fig. 1.8. Client-server three-tier architectures

In the process of working with DB there is a need to protect data from situations where there is a possibility of loss, in particular related to the

simultaneous access to the database of several users. One way to solve this problem is the *transaction* mechanism.



Transaction – a logical sequence of operations on data (reading, deleting, inserting, modifying), indivisible in terms of impact on DB, which is either performed all together, or all together canceled

Information systems, in which the operational processing of information in database is called OLTP (On Line Transaction Processing). A typical example of OLTP systems is mass customer service, such as airline booking, banking systems, or paying for telephone company services.

During the operation of OLTP-systems in their database accumulated a lot of data. It turned out that in order to function effectively, such systems must be organized in a slightly different way than that used in OLTP systems. Therefore, there is a need to develop a methodology and systems for analyzing these large amounts of data. At this stage, operational-analytical analysis has been developed, in which the grouping and generalization of data in any form required by the analyst, and taking into account the multidimensionality of data. This approach is called OLAP (On - Line Analytical Processing).

To implement it, the question arose of using specialized databases that focus on analytical processing and meet the requirements for decision support systems. Such databases are called Data Warehouses (DW). W.H. Inmon described this concept in detail in 1992 in his monograph “Building the Data Warehouse”.

The concept of DW is based on the idea of dividing data into those used for operational processing and those needed to solve analysis problems. This allowed the use of data structures that meet the requirements of their storage, taking into account the use of OLTP-system and analysis systems.



Data warehouse – a subject-oriented, integrated, immutable data set that supports chronology and is organized for decision support purposes.

All data in the DW are divided into three main categories: detailed data; aggregated data; metadata (Fig. 9).

The data recorded directly by OLTP-systems are *detailed*. Based on detailed data the *aggregated* (generalized) data are formed.

1. DATABASE CONCEPT

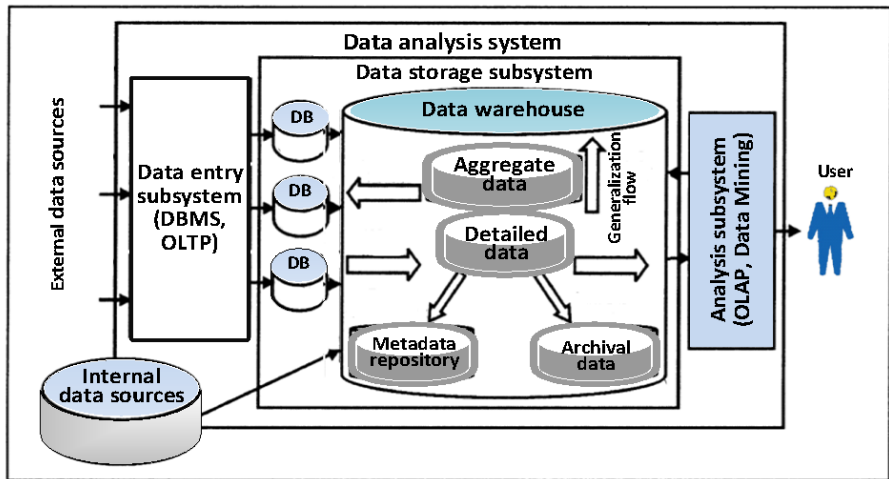


Fig. 1.9. An example of DW scheme in data analysis system

This is the data used in the analysis process. For convenience of work with DW the information on the data stored in it is necessary. Such data are called *metadata*. Metadata contains all the information necessary to extract, convert and download data from various sources, as well as for the subsequent use and interpretation of data contained in the DW. The creation of data analysis systems has developed in the direction of "intellectualization" of systems, the implementation of data processing technologies using the ideas of artificial intelligence. An intelligent system can be perceived as a computer analog of a person who is a specialist in a particular subject area. Therefore, in intelligent systems it is necessary to have a special kind of information called *knowledge*.



Intelligent System is an interactive computer system designed to support decision-making in various areas of activity on poorly structured and unstructured problems, which is based on the use of models and procedures for data processing and knowledge based on artificial intelligence technologies.

The most common type of knowledge-based systems are expert systems (ES). Its main elements are the knowledge base (KB) and a special

software module that performs logical operations (inferences machine) using the KB (Fig. 10).

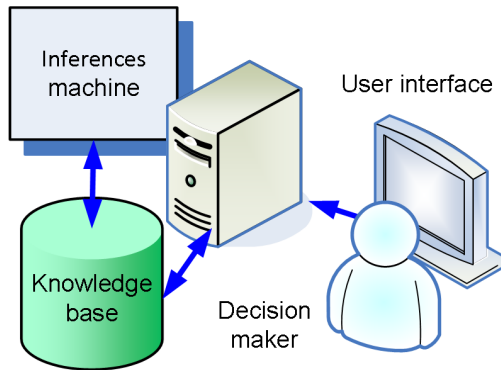


Fig. 1.10. Knowledge-based system



Knowledge base – a set of basic information that relates to a particular subject area. The main ones are *facts*, i.e., traditional data, and *rules* that reflect patterns and heuristic assumptions.

Thus, knowledge is presented in the form of data, for example, in the form of text in some formal language, in the form of a semantic network that defines the connections of various kinds between data elements. In other words, knowledge is some higher degree of data organization that allows for special interpretation.

1.3. Databases models

Data consistency. Hierarchical, network and relational models of data organization.

Data structuring problems have existed since the beginning of the use of computer technology. These problems in each case were solved individually. For example, the necessary additions to file systems were created by developing appropriate program libraries.

The development of information systems has led to more complex structural data. Additional individual data management tools have become an essential part of IS. It quickly became clear that it was impossible to provide complexed storage methods with a common library program and a standard file system. The key was the concept of *data consistency*.

The requirement to maintain data consistency across multiple files, and thus data integrity, requires that such a system have some of its own data (metadata) and even knowledge. Thus, the basis for the implementation of such opportunities should be some model of data organization.

Over time, *hierarchical*, *network* and *relational* models of data organization in the database were formed. The first two models are the historical predecessors of the relational model, which dominates modern databases and is supported by many DBMS.

Hierarchical database, the first version of which appeared in 1968, is a hierarchically organized set of record types. The model consists of an ordered set of tree-like structures. The tree type consists of one "root" record type and an ordered set of subtrees of record types (Fig. 1.11).

The organization of data in the database of the hierarchical type is defined by the following terms: element (attribute), record (group), group relationship.

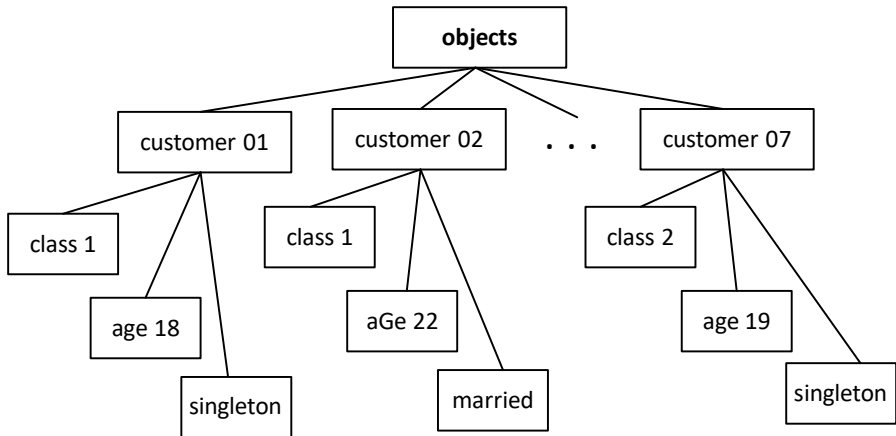


Fig. 1.11. Hierarchical database

A data *element* is the smallest unit of a data structure. A data element is also often called a field. Each element in the description of the database is assigned a unique name. He is addressed by this name during processing.

A *record* is a named set of elements. The use of records allows for one access to the database to obtain some logically related set of data. In the database, the records themselves are changed, added and deleted. The type of record is determined by the composition of its attributes.

A *group relationship* is a hierarchical relationship between two types of records. The original record is called the parent record (owner of the group relationship), and the members of the group relationship are called child records (subordinates).

The root record of each tree must contain a *key* field with a unique value. Non-root write keys should only be unique in group relationships. Each record is identified by a complete *composite key*, which means a set of keys of all records starting from the root and further in a hierarchical path.

The hierarchical model meets the requirements of many real problems, but not all. The difficulties of the hierarchical model can be illustrated when performing data manipulation operations on the example shown in Fig. 1.11. For example, the field "class 01" is duplicated in different trees. If you need to change the class value, you must browse all the trees to find such a field.

Partly the shortcomings of the hierarchical model were eliminated in the network data model. The network data model was very popular at the time, and is still used in some systems. The network data model is defined in the same terms as the hierarchical one. However, the main difference between these models is that in a network model, a record can be a member of more than one group relationship. If in hierarchical structures the child record must have one parent, then in the network structure of these children can have any number of parents (Fig. 1.12).

As you can see, in a network model, any element can be associated with any other element. If in the hierarchical model the connection "one to many" (1: M) is realized, then in the network connection "many to many" (M: M). However, the network model is also not without its negatives, primarily due to its complexity and complicated.

Revolutionary changes occurred in 1970 when Edgar Codd proposed a *relational* data model. To date, almost all modern DBMS support this model. Databases built in accordance with this model are called relational databases (RDB).

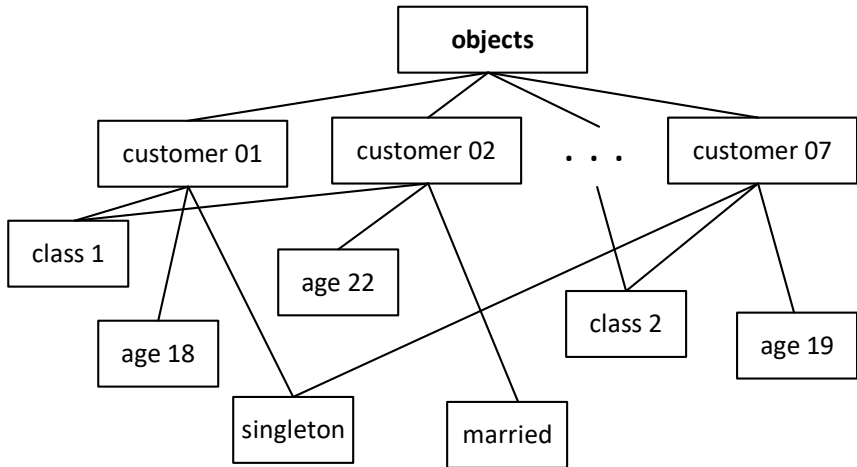
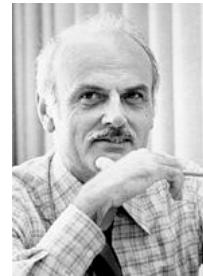


Fig. 1.12. Network database

The basic concepts of relational databases are relation, data type, domain, attribute, tuple, primary key.

The concept of *relation* underlies the relational model. The ratio is usually represented as a two-dimensional table. The table is clear, convenient and familiar to the person (see tab. 1.1). Codd himself proved that a set of relations (tables) could be used to store data about real-world objects and to model the relations between them.

The concept of "*data type*" in a relational model is adequate to the concept of data type in programming languages. Modern relational DB allow the storage of symbolic, numerical data, bit strings, specialized numerical data (such as "money"), as well as special "temporal" data (date, time, time interval).



Edgar Frank Codd, British researcher in the field of computer science, proposed a theoretical basis for relational databases and data warehouses

The concept of a *domain* is related to the data type, but is more specific to DB. The domain is defined by specifying some basic data type, which includes domain elements, and an arbitrary logical expression that is applied to the data type element. If the calculation of this logical expression gives the result "true", then the data element is a domain element. Interpretation of the concept of domain is understanding of domain as a valid potential set of values this type. For example, the domain "age" in our example table. 1.1 is defined on the basic type numerical data, but it may include only those data that correspond to age (in particular, such data may not contain zero values).

We are already familiar with the concept of an *attribute*, but a *tuple* is a set of pairs {attribute name, value}. "Value" is a valid domain value for this attribute. Tuples are also called "*relation - copies*". The set of tuples is *relation body*. The set of tuples must correspond to one *scheme of relations*, which is also called *relations title*. In other words, the representation of the relation is a table, the title of which is a relation scheme. Attribute names are called columns in this table, and rows are tuples.

Then a relational DB is a set of relations that have names and are displayed in the *database schema*.

Test questions and tasks



1. What are the features of the information society?
2. What is currently the most important component of the management decision-making process?
3. What current trends affect management decision-making processes?
4. What is data, information, knowledge?
5. What is a database?
6. What functions does the database management system provide?
7. Explain the concepts of "object" and "attributes".
8. What are the features of "client-server" technology?
9. What is a transaction?
10. What is the basis of the data warehouse concept?
11. What is the purpose and features of knowledge bases?
12. What are the models of data organization in the database?
13. Name the basic concepts of relational databases.

1. DATABASE CONCEPT



Search the Internet for information on the affiliation between decision-making processes, data availability, information operations, and read the materials received.



On the example of decision-making in the enterprise in any chosen field, give a description of the problem area, management functions, purpose (desired result), features of information activities and its impact on the decision-making process. Make the materials in the form of a report.



Laboratory work 1. The main features of database design tools.



2. THE CONCEPT OF DATA MODEL

2.1. Properties of relations

Objects and linked relationally. Concept of the key. Primary and foreign keys

Relational are divided into objects and linked.

In *objects relational* one of the attributes uniquely identifies a single object (tuple). This attribute is called the *primary key (PK)*.

For convenience, the primary key is written in first column of the table. In the example of the relation given in table 1.1 key can be attribute "customer code". If the primary key consists of more than one column, it is called a *composite primary key*.

There should be no rows with the same keys in the relation, ie there should be no duplication of objects. This is a major limitation of the relational model to ensure data integrity.

A *linked relations* stores the keys to two or more object relations. That is, the keys establish connections between objects of relation.

Consider, for example, the following object relation CUSTOMERS, object relation GOODS and linked relation BUY. A linked relation can have attributes other than the fields it binds that depend on that connection. An example might be «quantity» attribute to the BUY relation.

2. THE CONCEPT OF DATA MODEL

CUSTOMERS			
customer code	customer class	age	income
01	1	18	12500
02	1	22	10000
03	2	30	7000
04	1	32	12000
05	2	24	9500

GOODS		
good code	name	cost
0112	bread	12,50
0333	milk	25,00
0657	cookies	7,00
0778	sweets	32,70

BUY		
customer code	good code	quantity
01	0333	1
01	0657	2
02	0778	5
03	0657	3
04	0112	2
04	0333	2
05	0778	3

Keys in linked relations are called *foreign keys* (FK) because they are the primary keys of other relationships. The relational model imposes constraints on FK to ensure data integrity, which is called *reference integrity*.



Each FK must point to a string of some object relations, i.e., FK cannot refer to a non-existent object

In other words, the value of the FK must match the available values of the PK of another table (Fig. 2.1).

Such interconnections between tables are called a *relationship*. The relationship between two tables is established by assigning the value PK of one table to the value of FK another table.

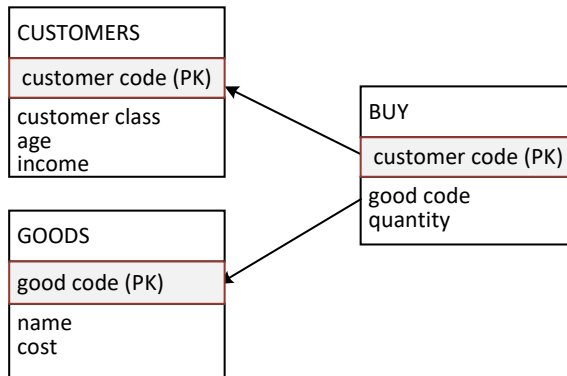


Fig. 2.1. Reference integrity

A group of linked tables is called a *database schema*.

The relationship can be *one-to-one* relationship, *one-to-many* relationship or *master-detail* relationship. In this case, the table containing FK is detail table, and the table containing PK is master table.

2.2. Normalization of relations

Normalization of relations. Normal forms. Системи бізнес-аналітики

One of the first complex problems faced by database developers is the task of gathering data elements into one tuple. Improper grouping of data elements can lead to various defects, such as duplication of data, inability to fulfill a query, etc. To solve this problem, the approach of relations *normalization* is used.

This approach is based on the fact that RDB relations contain both *structural* and *semantic* information. Structural information is defined by a relations scheme, and semantic information is expressed by functional relationships between attributes. Based on this, the composition of the attributes must meet two main requirements:

1) there can be no meaningless (undesirable) functional relationships between attributes;

2. THE CONCEPT OF DATA MODEL

2) when grouping attributes there should be minimal duplication of data.

Satisfaction of these requirements is achieved by *normalizing relations*.



Normalization of relations is a step-by-step reverse process of decomposing the initial relations into smaller and simpler ones. At the same time all possible functional relationships are established

The normalization procedure was developed by Codd. In this order, three normal forms are defined: 1NF, 2NF, 3NF. There are other forms, but it is usually considered sufficient to reduce to 3NF. Each of the normal forms limits the types of functional relationships of relations.



The relation in which all the attributes are simple is reduced to the **first normal form (1NF)**

The relation BUY2 is not normalized, because it contains a complex attribute "good":

BUY2			
customer code	good		quantity
	code	name	
01	0333	milk	1
01	0657	cookies	2
02	0778	sweets	5
03	0657	cookies	3
04	0112	bread	2
04	0333	milk	2
05	0778	sweets	3

Let's bring this relation to 1NF, that is we will get rid of a difficult attribute "good". In the received relation BUY3 the key is composite – from attributes "customer code" and "good code".

If the relation is in 1NF, then all non-key attributes are functionally dependent on the key. But the degree of dependence can be different. If the attribute depends only on part of composite key, then it is a partial dependency. For example, the non-key attribute "good name" depends only on the attribute "good code", i.e., functionally only on part of the key.

BUY3			
customer code	good code	good name	quantity
01	0333	milk	1
02	0778	sweets	5
03	0657	cookies	3
04	0112	bread	2
05	0778	sweets	3

If the non-key attribute depends on the whole composite key and is not partially dependent on its parts, then it is a complete functional dependence on the composite key. In relation BUY3 does not have attributes which are in full functional dependence on the composite key.

Partial dependence leads to the following anomalies:

- there is duplication of data;
- there is a problem of control of data redundancy;
- there is a problem with either not being able to include a new object in the relationship or removing an object from the relationship.



The relation is in the **second normal form** if it is in 1NF and each non-key attribute is functionally completely dependent on the composite key.

To eliminate the partial dependence and bring the relation to 2NF, it is necessary to decompose it into two relations. For example, to the type of the above relationship GOODS and BUY.

The attribute B of relation R functionally depends on the attribute A of same relation, if each value of attribute A corresponds to no more than one value of attribute B. The functional dependence is displayed as follows: $A \rightarrow B$. For GOODS, the “cost” attribute is functionally dependent on the “name” attribute.

In relationship between attributes there may be another type of dependence – *multi-valued* dependence. Attribute B multi-valued dependence on A ($A \twoheadrightarrow B$), if each value of A corresponds to a set of values of B, not related to other attributes in relation R.

Multivalued dependence is possible if there are at least three attributes in the relationship: a key and at least two independent attributes. For example, in BUY3 relations between a “consumer” and a “good” there is a

2. THE CONCEPT OF DATA MODEL

“many-to-many” relationship (M: M) because a consumer can buy one or more goods, and conversely, one product can be bought by several consumers. There is independent multi-valued relationship between “consumer” $\rightarrow\rightarrow$ “good” because the values of multi-valued attributes “good” and “consumer” are not related in any way, and it is possible to change their values in any which relation line.

Thus, the reduction to 2NF can be described as follows. Let the relation R (A, B, C, D), PRIMARY KEY {A, B} be given and let there be a functional dependence $A\rightarrow D$. The normalization procedure in 2NF involves replacing this relation with the following two projections R1 and R2:

R1(A, D) PRIMARY KEY {A}

R2(A, B, C) PRIMARY KEY {A, B}

FOREIGN KEY {A} REFERENCES R1.

Often, 2NF also causes inconvenience due to data redundancy and *transitive functional dependencies*. To eliminate them, the next step of normalization is 2NF to 3NF converts.



If for attributes A, B, C the conditions $A \rightarrow B$, $B \rightarrow C$ are fulfilled, and the inverse dependence is absent, then C depends on A **transitive**.

For example, in relation BUY3 the dependence “consumer” \rightarrow “good” \rightarrow “quantit” is transitive.

The presence of transitive functional dependencies causes the following anomalies (for example BUY3 relation):

- there is duplication of information (repetition of “good name” values);
- there is a problem of control of redundancy of data (change of “goods name” causes need of search and change of all corresponding at all consumers);
- it is not possible to add new data (about a new good if there are currently no consumers of this good. Conversely, when deleting a group of consumers, data on the good may disappear).



The relation is in 3NF if it is in 2NF and in it there are no transitive dependences of non-key attributes on a key

Thus, relation BUY3 cannot be included in database. This relation should be divided into separate relations, such as those listed at the beginning of this paragraph.

Reduction to 3NF is described as follows. Let the relation R (A, B, C), PRIMARY KEY {A} be given and be the functional dependence $B \rightarrow C$. The normalization procedure in 3NF involves replacing this relation with two projections R1 and R2:

R1(B, C) PRIMARY KEY {B}
R2(A, B) PRIMARY KEY {A}
FOREIGN KEY {B} REFERENCES R1

The third normal form eliminates redundancy and anomalies, if the relationp has one key, and other dependencies, including multivalued, it is absent. However, if there are dependencies other than the key dependency, then 3NF does not ensure the absence of transaction anomalies.

In this case, use enhanced 3NF - the so-called normal Boyce-Codd form. There are also 4NF and 5NF.



The level of normalization of a relation depends on its semantics, which are determined by functional dependencies

2.3. Relations operations

Relational algebra. Relational calculus.

In addition to the concept of relations, Codd proposed operations system of that allow to obtain some relations from others. This approach makes it possible to divide the data into two parts - which are stored and which are calculated. This saves memory by retrieving data by calculating some of the stored data.

In this sense, the most common interpretation of the relational data model belongs to Christopher Date. He reproduces this interpretation with various refinements in almost all his books, for example, in “An Introduction to Database System”.

2. THE CONCEPT OF DATA MODEL

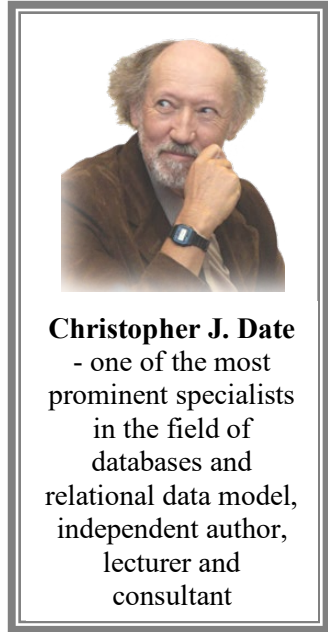
According to Data's teaching, the relational model consists of three parts that relate to different aspects of the relational approach: the structural part, the manipulation part, and the integral part.

The structural part of the model reports that the only data structure used in relational databases is a normalized n -ar relation. We have considered the very concept and properties of the structural component of the relational model since the beginning of this section.

In the manipulation part of the model, two fundamental mechanisms of manipulation of relational databases are declared - *relational algebra* and *relational calculus*.

Relational algebra is based mainly on classical set theory (with some refinements), and relational calculus is based on mathematical logic (the apparatus for calculating first-order predicates).

Therefore, the efficiency of a relational DB is determined by the ability to perform on the relations of eight operations of relational algebra. Such operations on relations are the following: Union, Intersection, Difference, Cartesian product, Selection, Projection, Connection, and Division. The first four are traditional operations on sets, the last are special relational operations. Another additional operation proposed by Codd is Rename, but by community decision it has become one of the main operations.



Relational algebra has ***the property of closedness***. This is because the result of a relational operation on a relation is also a relation. Therefore, the operations of relational algebra can be nested, that is, the argument of a particular operation may be the result of another operation.

When considering the operations of relational algebra, we will denote attributes in capital letters from the beginning of the Latin alphabet: A, B, ...,

and sets of attributes - in capital letters from the middle of the Latin alphabet: L, M,....

In addition, we introduce the concept of *compatibility of relations*. The relations R1 (A1,... An) and R2 (B1,... Bk) are said to be compatible if:

- 1) they have the same number of attributes, i.e., $k = n$;
- 2) it is possible to establish a mutually unambiguous correspondence between the domains of the attributes of the first and second relations.



The notion of relations compatibility is necessary because some operations (Union, Intersections, and Differences) are only defined for compatible relationships.

Because different relations can contain attributes with the same name, attribute names may end up being repeated when performing binary operations. To ensure the uniqueness of attribute names, they are specified by the names of the corresponding relations according to the following syntax: <relations name>. <attribute name>.

Binary operations also have a number of properties:

- operation ϕ is *commutative*, if $A \phi B = B \phi A$;
- operation ϕ is *associative* if $(A \phi B) \phi C = A \phi (B \phi C)$;
- operation ϕ is *distributive* with operation θ , if $A \phi (B \theta C) = (A \phi B) \theta (A \phi C)$.

$\theta (A \phi C)$.

Next, we consider the operations of relational algebra.

1. Union.

The union of compatible relations (denoted as $R1 \cup R2$) with schemes $R1(L)$ and $R2(L)$ is the following relation $R(L)$, which contains tuples of both combined relations, but without repetitions:



Fig. 2.2. Union operation $R1 \cup R2$

$$R(L) = R1(L) \cup R2(L) = \{r \mid r \in R1 \vee r \in R2\}.$$

In fig. 2.2 shows the interpretation of the Union operation on the example of two sets. The following is an example of a relations.

R1	
A	B
a1	b1
a1	b2
a2	b3

R2	
A	B
a1	b1
a2	b1

R = R1 \cup R2	
A	B
a1	b1
a1	b2
a2	b1
a2	b3

The operation is commutative, associative and distributive regarding Intersection operation.

2. Intersection.

The intersection of compatible relations (denoted as $R1 \cap R2$) with schemes $R1(L)$ and $R2(L)$ is the following relation $R(L)$, which contains tuples that are part of both operands:

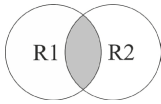


Fig. 2.3.
Intersection operation $R1 \cap R2$

$$R(L) = R1(L) \cap R2(L) = \{r \mid r \in R1 \ \& \ r \in R2\}.$$

In fig. 2.3 shows the interpretation of the Intersection operation on the example of two sets. The following is an example of a relations.

R1	
A	B
a1	b1
a1	b2
a2	b3

R2	
A	B
a1	b1
a2	b1

R = $R1 \cap R2$	
A	B
a1	b1

The operation is commutative, associative and distributive regarding Union operation.

3. Difference.

The difference of compatible relations (denoted as $R1 - R2$) with schemes $R1(L)$ and $R2(L)$ is the following relation $R(L)$, containing those tuples from the first operand $R1$, which are not in the second operand $R2$:



Fig. 2.4.
Difference operation $R1 - R2$

$$R(L) = R1(L) - R2(L) = \{r \mid r \in R1 \ \& \ r \notin R2\}.$$

In fig. 2.4 shows the interpretation of the Difference operation on the example of two sets. The following is an example of a relations.

R1	
A	B
a1	b1
a1	b2
a2	b3

R2	
A	B
a1	b1
a2	b1

R = $R1 - R2$	
A	B
a1	b2
a2	b3

The operation is not commutative, not associative and not distributive with other operations.

4. Cartesian product

The Cartesian product of relations R and S (denoted as $R \times S$) with schemes R (A1, A2,...An) and S(B1, B2,...Bm) is relation Q (A1, A2,...An, B1, B2,...Bm), which contains all possible tuples connections of relation R with relation S: $Q = R \times S = \{(r, s) \mid r \in R \ \& \ s \in S\}$. The following is an example of a relations.

R	
A1	A2
a11	a21
a11	a22
a12	a23

S	
B1	B2
b11	b 21
b12	b21

Q = R×S			
A1	A2	B1	B2
a11	a21	b11	b 21
a11	a21	b12	b21
a11	a22	b11	b 21
a11	a22	b12	b21
a12	a23	b11	b 21
a12	a23	b12	b21

The operation is commutative and associative.

5. Selection.

First, we define the *comparability* of attributes. Let θ be one of the comparison operators, for example: = , \neq , \geq , $>$, \leq , $<$. Attributes A and B of the same or different relations are called θ -comparable if for any values of $a \in A$ and $b \in B$ the result of the operation $a \theta b$ is definite (true or false). In turn, the sets of attributes $L = (A1, \dots Ai, \dots, Ak)$ and $M = (B1, \dots, Bi, \dots, Bn)$ are called θ -comparable if $k = n$ and Ai θ -comparable to Bi ($i=1,2, \dots k$). Then the expression $L \theta M$ is understood as follows:

$$L \theta M = (A1 \theta B1) \ \& \ \dots \ \& \ (Ak \theta Bk).$$

We can now define that Selection operation of the relation R under condition θ -comparable attributes $L \theta M$ of this relation (denoted by $R[L \theta M]$) is a relation whose tuples correspond to the condition $L \theta M$:

$$S = R[L \theta M] = \{r \mid r \in R \ \& \ r [L] \ \theta \ r [M]\}.$$

This operation is also written as $\sigma_{L \theta M}(R)$. The following is an example of a selection operation provided by the comparison operator =:

R		
A	B	C
a1	b1	c1
a2	b1	c2
a2	b2	c3

R[A=a2]		
A	B	C
a2	b1	c2
a2	b2	c3

6. Projection

The projection operation is done by removing from relation values that do not belong to the attributes on which the projection is performed. In the final, duplicate tuples are removed. That is, the projection operation is some vertical attribute filter.

If r is a tuple of relation R , and L is a subset attributes of relation R , then the above definition of the projection has the expression:

$$S = R[L] = \{r[L] \mid r \in R\}.$$

The projection operation is also written as $\pi_L(R)$. The following is an example of a projection operation provided by $L = A, C$:

R		
A	B	C
a1	b1	c1
a1	b2	c1
a2	b1	c2
a2	b2	c3

R[A,C]	
A	C
a1	c1
a2	c2
a2	c3

7. Connection

If relation R_1 has a scheme (L,M) , relation R_2 – scheme (N,P) , and the sets of attributes M and N are θ -comparable, then the connection of relations R_1 and R_2 under the condition $M \theta N$ (denoted as $R_1[M \theta N]R_2$) is the relation S with scheme (L,M,N,P) . The tuples of relation S are obtained by connecting those tuples of relations R_1 and R_2 on which the condition $M \theta N$ is fulfilled: $S = R_1[M \theta N]R_2 = \{(r, s) \mid r \in R_1 \ \& \ s \in R_2 \ \& \ r[M] \theta s[N]\}$.

As a result, the attributes on which the connection operation is performed are repeated in the final relation. The operation is commutative and associative.

If connection is made under the condition of equality, then the comparison attributes are removed from the final relations. Such a connection is called a *natural connection* and the symbol "*" is used to denote it. For example, the relations $R_1(A, B, C, D)$ and $R_2(C, D, E)$ are given. As a result of the operation $S = R_1 * R_2$ we obtain the relation $S(A, B, C, D, E)$. The following is an example of performing a natural connection operation:

R1			
A	B	C	D
a1	b1	c1	d1
a1	b1	c2	d1
a1	b2	c1	d1
a2	b3	c1	d1
a2	b4	c2	d3

R2		
C	D	E
c1	d1	e2
c2	d1	e3
c2	d1	e1

R1*R2				
A	B	C	D	E
a1	b1	c1	d1	e2
a1	b2	c1	d1	e2
a2	b3	c1	d1	e2
a1	b1	c2	d1	e3
a1	b2	c2	d1	e3
a1	b1	c2	d1	e1
a1	b2	c2	d1	e1

8. Division

First of all, we define the concept of *relation image*. Let the relation R with scheme R(M,N) be given. The relation image of R in the tuple $t1 \in R[M]$ is such a set of tuples $t2 \in R[N]$, for which the union $(t1, t2)$ belongs to the relation R. The image of R in the tuple t1 is denoted by $IR(t1)$ and is formally defined in as follows: $IR(t1) = \{t2 \mid t2 \in R[N] \ \& \ (t1, t2) \in R\}$.

Example:

R		
A	B	C
a1	b1	c1
a1	b1	c2
a1	b1	c3
a2	b2	c4

IR(a1)	
B	C
b1	c1
b1	c2
b1	c3

IR(a1,b1)
C
c1
c2
c3

IR(c1)	
A	B
a1	b1

The result of operation Division the relation R (M, N) by the relation S (K, L) by the sets of attributes N and K (denoted by $R [N \div K] S$) is the relation Q(M), consisting of such tuples $t \in R[M]$, the images $IR(t)$ which contain all tuples of projection $S[K]$, ie:

$Q = R [N \div K] S = \{t \mid t \in R[M] \ \& \ IR(t) \supseteq S[K]\}$. The condition for performing the Division operation is compatibility of projections $R[N]$ and $S[K]$.

The operation is not commutative and not associative. Example:

R		
A	B	C
a1	b1	c1
a1	b1	c2
a1	b3	c2
a2	b1	c4

S	
C	D
c1	d1
c1	d2
c2	d1
c2	d3

S(C)
C
c1
c2

R[C ÷ C]S	
A	B
a1	b1

Test questions and tasks



1. What is a primary key and its purpose?
2. What is a foreign key and its purpose?
3. What is the condition for data integrity?
4. What relationships between tables can exist?
5. What is the principle of normalization of relations?
6. What are the conditions for bringing the relationship to the first normal form?
7. What are the conditions for bringing the relationship to the second normal form?
8. What are the conditions for bringing the relationship to the third normal form?
9. Why was relational algebra proposed?
10. Name the basic operations of relational algebra.



Laboratory work 2. Inter-tabular relations and normalization.



Create a table in the rows of which enter the types of operations of relational algebra, and in the columns - their criteria. Evaluate, at your discretion, the benefits of each for maintaining databases and line them up in descending order of overall scores.

3. SEMANTIC MODELING

The semantic data model, like the relational model, includes structural, manipulative, and integral parts. But the main purpose of semantic models is to provide the ability to express the semantics (meaning) of data.

One of the most important and common semantic models is the Entity-Relationship (ER) model. The process of semantic modeling consists in the selection of objects of the subject area (entities), the establishment of the properties of selected objects and the identification of existing connections between them.

Graphical representation (notation) of this model is known as ERD (EntityRelationship Diagram) or ER-scheme (ER-diagram).

The basic concepts of the ER model are entitie, relationship and attribute. As in relational schemes, the concept of normal forms is introduced in ER-models. Their content is very close to the content of relational normal forms. This approach allows at the initial stage to correctly design the logical structure of the database.

The process of database design can be divided into two stages: logical and physical design. The result of the first of them is a logical (or conceptual) data model, which is usually expressed by an ER-diagram. The result of the second stage is a ready-made database or DDL-script (Data Definition Language) for its creation.

There are several types of tools (CASE-tools) that allow you to create ER-diagrams and design databases. Many of these tools are designed not only for data design, but also for other tasks, such as business process modeling, functional modeling, etc. In the table. 3.1 presents the most popular data design CASE-tools.



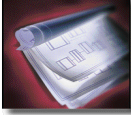





Peter Chen –
An American
originally from
Taiwan, he first
proposed the ER
model in 1976



CASE (Computer-aided software engineering) - a set of tools and methods for software design

Table 3.1

The most popular data design CASE-tools

CASE-tools	Producer	URL
 Designer 2000	Oracle	http://www.oracle.com/
 ERwin	Computer Associates	http://www.cai.com http://www.erwin.com
 PowerDesigner	Sybase	http://www.sybase.com
 ER/Studio	Embarcadero	http://www.embarcadero.com
 Visible Analyst	Visible Systems	http://www.visible.com
 Visio Enterprise	Microsoft	http://www.Microsoft.com

The most popular of these are ERwin, ER/Studio and PowerDesigner. The latter is considered by many experts to be more attractive due to its many advantages. Among them are support for more than 70 relational databases, including specialized BI-systems, an excellent relationship editor that simplifies visualization and clear graphical comparison of data sources and receivers, a combination of conceptual modeling and data modeling, business processes and applications (Fig. 3.1). Due to this, PowerDesigner is recognized as a universal tool for database design and organization of large data warehouses, as well as for the implementation of complex analytical programs.

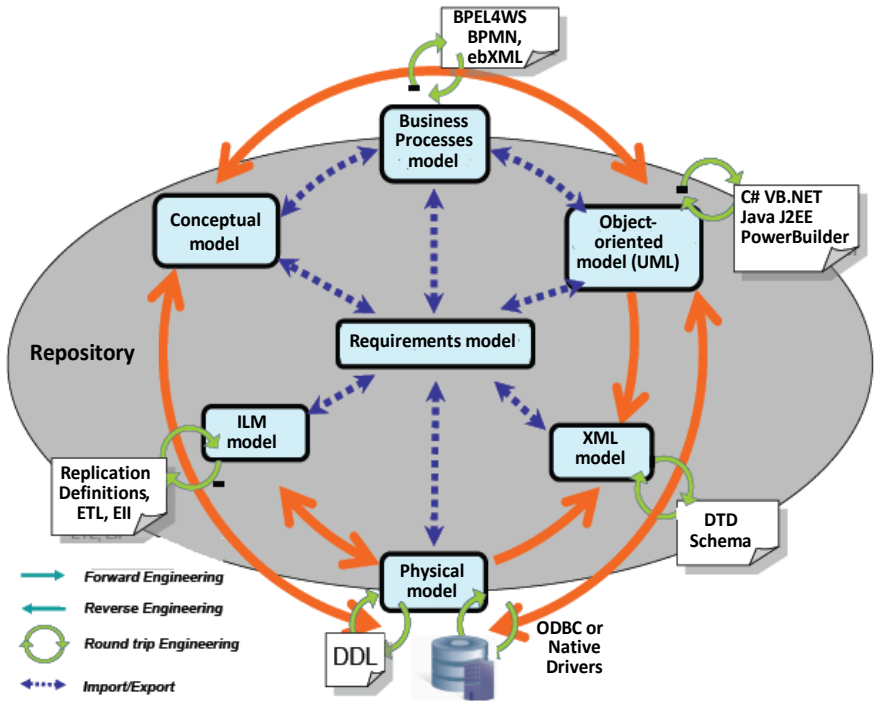


Fig. 3.1. PowerDesigner supports most of the commonly used standards modeling

3.2. Construction of semantic models

Basic concepts of the ER-model. Barker notation. Three types of relationships. Subtypes and supertypes of entities.

Consider the construction of semantic data models at the stage of infologic design using the ER-model. Subject area data modeling is based on the use of graphical diagrams that include a small number of disparate components.

As noted, the basic concepts of the ER-model are the entity, relationship and attribute.

An *entity* is an object of a subject area, information about which must be stored and made available. The names of entities are usually nouns, for example: Customer, Product, Account. There is an *entity type* and an *entity instance*.

The concept of entity type refers to a set of homogeneous objects, events, personalities that act as a whole. An entity instance refers to a specific thing in a set.

Account
AccNo
Balance
Type

In ER-model diagrams (in Barker notation), the entity is represented as a rectangle containing the entity name.

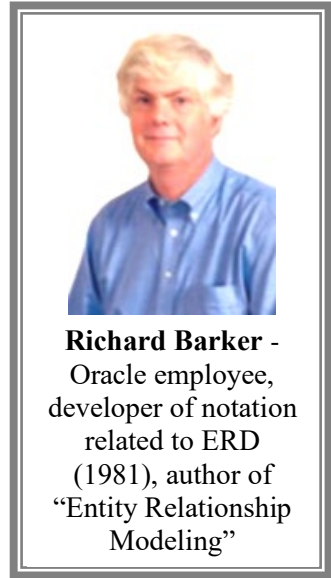
Barker's notation takes into account the properties of relations, in particular the use of abstraction (nested blocks), the designation of the number of elements (the so-called "goose foot"), exclusion (exclusion arc), recursion (cyclic structures).

An *attribute* is a named characteristic of an entity that defines its properties and takes values from a set of values. Each attribute is provided with a name that is unique within the entity. This concept is similar to the concept of attribute in relation.

Attributes can belong to one of three different types: descriptive, indicative, auxiliary. Descriptive attributes represent the facts inherent in each instance of the entity. Indicative attributes are used to name or denote instances of an entity. Auxiliary attributes are used to associate an instance of one entity with an instance of another.

A set of one or more attributes whose values uniquely identify each entity instance is called an entity *identifier*, or *key*. Each entity instance must have at least one identifier. If there are several identifiers, one of them is selected as privileged.

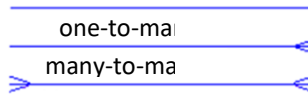
Relationship is a named graphically represented abstraction of associations between entities. Most relationships belong to the category of



binary and take place between two entities.

The name of relationship is usually represented by a verb. Example of relationship between entities: Customer-**Buys**-Goods.

Among the relationships there are three fundamental types of communication:



One-to-one relationship (1:1) when one instance of one entity is associated with a single instance of another entity. One-to-many relationship (1:M) when one instance of one entity is associated with one or more instances of another entity, and each instance of the second entity is associated with only one instance of the first entity. A many-to-many relationship (M: M) when one instance of one entity is associated with one or more instances of another entity, and each instance of the second entity is associated with one or more instances of the first entity.

Each relationship can have one of two modalities:



The "mayby" modality (conditional relationship) means that an instance of one entity may be associated with one or more instances of another entity, or may not be associated with any instance. The "must" modality (unconditional connection) means that an instance of one entity must be associated with at least one instance of another entity.

In accordance with the types of relationship entities belong to one of four classes: core; associative; characteristic; significant.

The core entity (core) is an independent entity. An associative entity is an entity that formalizes an M:M relationship between two or more entities or a 1:1 relationship between entity instances. Characteristic entity (characteristic) is an entity that formalizes the relationship of the form 1:M or 1:1. The purpose of the description within the subject area is to describe or clarify some other entity. A signifying entity is a entity that also formalizes a 1: M or 1:1 relationship between two entities, but differs from a characteristic in that it does not depend on the designated entity.

The elements of the ER-model also include *subtypes* and *supertypes* of

entities. The entity can be split into two or more mutually exclusive subtypes. Each of the subtypes has general (common) attributes and/or relationships, and can also define its own attributes and/or relationships. General attributes / relationships are explicitly defined once at a higher level.

3.3. Diagrams of entity relationships

ER-instances and ER-types diagrams. The degree of relationship. Normal forms. Obtaining relational schemes from ER-diagrams

The following graphical tools are used to build ER-models:

- diagrams of ER-instances;
- diagrams of ER-types, or ER-diagrams.

The following is an example of an ER-instance chart and an ER-type chart that corresponds to the ER-instance chart under consideration:

customer code	buys	good code
01	●	●0112
02	●	●0333
03	●	0456
04	●	●0657
05		●0778



In the ER-diagram, entities are indicated by rectangles with the name at the top. They can also show the attributes of the entity (Fig. 3.2). The diamond indicating the connection may not be indicated.

Based on the analysis of diagrams of ER-types, the relations of the projected DB are formed. This takes into account the degree of entities relationship and the class of their affiliation. The class is determined based on the analysis of ER-instances diagrams of the corresponding entities.

The degree of relationship is a characteristic of the connection between the entities, which can be of the type: 1:1, 1:M, M:1, M:M. The

class of affiliation (CA) of the entity can be: mandatory and optional. The CA of the entity is mandatory if all instances of this entity are necessarily involved in this relationship, otherwise the CA is optional.

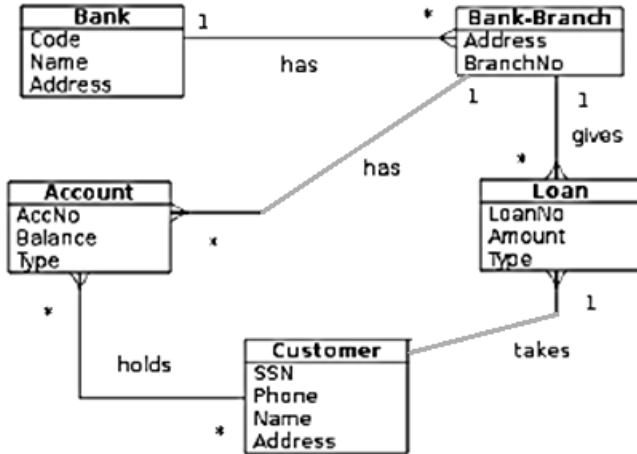


Fig. 3.2. Example ER-diagram of entity relationships

By varying the CA for each of the relationship types, several variants of ER-type diagrams can be obtained. Consider some examples.

In the previous diagram, the degree of relationship between entities is 1:1, and the CA of both entities is optional. Indeed, each customer buys no more than one good, and each good is purchased by no more than one customer (1:1). At the same time, some customers do not buy any goods; and there is a good that has not been purchased by either customer (the affiliation class of both entities is optional).

If each customer must buy no more than one good, and each good must be purchased by no more than one consumer, we have a 1:1 relationship and a mandatory CA. If each customer must buy no more than one good, and each good must be purchased, but more than one consumer, we have relationships of type 1:M and a mandatory CA.

Generally, several variants of CA are possible for 1:M relationship. Denote the mandatory class by the symbol "A", and the optional - by the symbol "O". Then the variants can be conditionally represented as: A-A, A-O, O-O, O-A. For M:1 relationship, 4 similar variants are also possible.

Relationship 1: M and variant O-A occurs when each customer can buy several goods or none, but each good is purchased by one customer. Relationship M:M and variant A-O is when each customer buys at least one good, and the good can be purchased by more than one customer; and there is a goods that no one buys.

ER-diagrams are convenient because the process of selecting entities, attributes and relationships is iterative. The developed first approximate version of the diagrams can be gradually refined by obtaining additional information about the subject area.

As in relational schemes, the concept of normal forms is also introduced in ER-diagrams. Their content is very close to the content of relational normal forms.

In the first normal form of ER-diagrams, repetitive attributes are eliminated, ie implicit entities are identified, which at first glance were considered attributes. The second normal form removes attributes that depend only on part of the unique identifier. This part of the unique identifier defines a separate entity. The third normal form removes attributes that depend on attributes that are not included in the unique identifier. These attributes are the basis of a separate entity.

From ER-diagrams it is possible to receive relational schemes. This happens in several stages. First, each simple entity (which is not a subtype or supertype) is transformed into a table. The entity name becomes the table name. Each attribute becomes a possible column in the table with the same name. Next, the components of the unique entity identifier are converted to the primary key of the table. If there are several possible unique identifiers, then the most used one is selected. M:1 and 1:1 relationships determine foreign keys. A copy of the unique identifier is made from the end of the "one" link, and the corresponding columns make up the foreign key. Next, indexes are created for the primary key, foreign keys, and those attributes on which the queries are supposed to be based.

Finally, the question of subtypes presence in the conceptual scheme is resolved. In this case, either all subtypes are collected in one table, or a separate table is created for each subtype.

Test questions and tasks



1. What is the main purpose of semantic models?
2. What is the process of semantic modeling?
3. Explain the basic concepts of ER-models and their graphical representation on models.
4. What tools do you know that allow you to create entity-relationship diagrams?
5. Describe the schemes of three fundamental types of relationship.
6. Name the four main classes of entities.
7. What is the difference between ER-instance diagrams and ER-type diagrams?
8. Explain the normal forms of ER-diagrams.
9. How to get relational schemes from ER-diagrams?



Using Internet resources, search for tools that allow you to create entity-relationship charts. Create a table in the rows of which enter the found tools, and in the columns give their main characteristics. Compare them and line them up in descending order of what you think is the level of their benefits.



Laboratory work 3. Creating databases and tables in Sybase Power Designer



4. INTRODUCTION TO DATABASE PROGRAMMING

4.1. The concept of SQL language

The basic concept of SQL language. The main operators.

With the databases dissemination, there was a need to create and standardize programming languages that allowed users to manipulate data. One such language that developed as a result of the relational data model is the Structured Query Language (SQL). This language has now become widespread and has in fact become the standard language of relational databases.



SQL is not a procedural language. This means that in this language you can specify what to do with the database, but you can not describe the algorithm of this process.

SQL is a relationally complete language. This means that the expressions of this language allow us to define each relationship using algebraic expressions of relational algebra.

The emergence of the SQL language is associated with the development in the early 1970s of the experimental relational DBMS IBM System R. For this DBMS was created a special language SEQUEL that allows relatively simple data management. This language was then renamed SQL.

The American National Standards Institute (ANSI) issued the first SQL standard in 1986. In 1987, the International

Organization for Standardization (ISO) adopted it as an international one. Further development of the language is associated with the adoption in 1992 of a new extended standard ANSI SQL-92 or simply SQL2. The next standard was SQL: 1999 (SQL3). The standard adopted in 2003 (SQL: 2003) is currently in force, with minor modifications made later.

The presence of clear basic principles of SQL leads to the compatibility of different implementations of the language helps to increase of software and databases portability and the versatility of database administrators. SQL can be used both to execute data queries and to build applications.

Implementation in SQL concept of operations focused on tabular data representation, allowed to create a compact language with a small set of operators (commands) (Fig. 4.1). The main categories (subsets) of SQL language commands: DDL – data definition language; DML – data manipulation language; DCL – data control language. These categories side data administration commands and transaction management commands.

Data Definition Language (DDL) is designed to create and modify database object structures, such as creating and deleting tables. The main DDL commands are as follows: CREATE TABLE, CREATE INDEX, ALTER TABLE, ALTER INDEX, DROP TABLE, DROP INDEX.

Data Manipulation Language (DML) is used to manipulate data inside relational DB objects using three main commands: INSERT, UPDATE, DELETE. Adjacent to this group is the SELECT command. This command,



along with its many options and expressions, is used to query DB.

Data control language (DCL) commands allow you to control access to data. Typically, they are used to create objects associated with access to data, and also serve to control the distribution of privileges between users. Data control commands: GRANT, REVOKE.

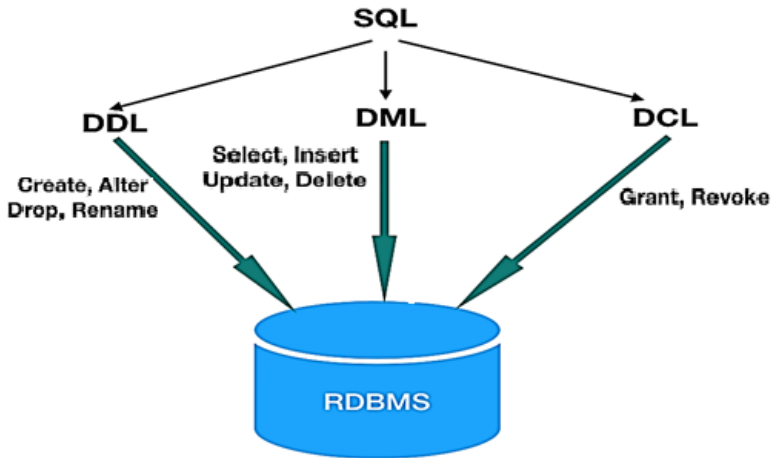


Fig. 4.1. The main categories SQL commands

With the help of data administration commands, the user monitors the actions performed and analyzes the database operations. These commands can also be useful in analyzing DBMS performance.

Database transaction management commands include: COMMIT, ROLLBACK, SAVEPOINT, SET TRANSACTION.

SQL language as a powerful tool that provides access to information contained in relational databases, is the basis of many DBMS. This is due to many advantages of the SQL language, such as standardization, independence from specific databases, the ability to transfer from one computer system to another, support for client-server architecture, and others.

However, the SQL language is not without its drawbacks, which requires researchers to think about new principles of data access languages for the next generation.

4.2. Data types in SQL

Strings. Numerical data. Logical data. Temporal data.

In SQL, as in all programming languages, the types of data supported are important. The data type determines the set of valid values, the format of their storage, the size of the allocated memory and the set of operations that can be performed on the data. SQL provides several types of data (Fig. 4.2).

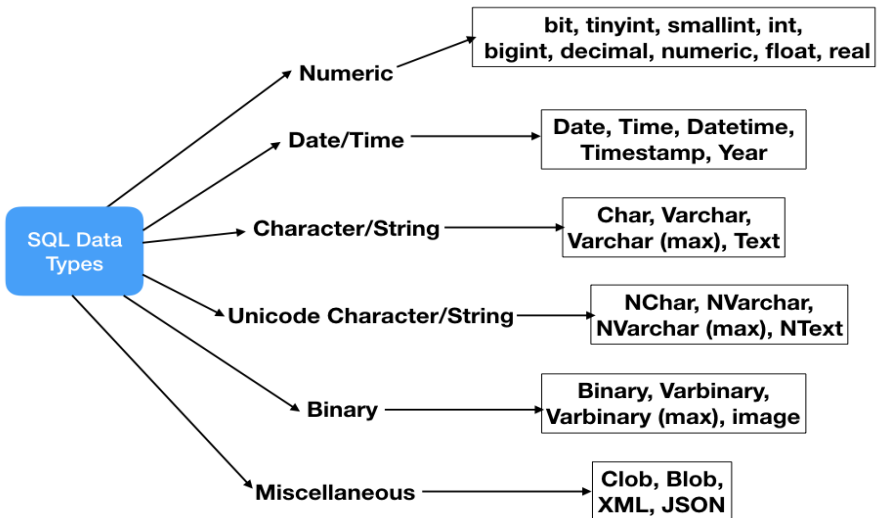


Fig. 4.2. Types of data in SQL

In most databases, the bulk of data is presented in numerical, symbolic and date/time forms (Table 4.1). This is natural, because such data are inherent in the subject areas of human activity.

Numerical data can be of two types - exact and approximate. The first allows you to accurately express the value of the number. At the same time, some values have a very large range of values, ie both very small (close to 0) and very large. For example, to present scientific data. In such cases, it is sufficient to limit themselves to some approximate representation (taking into account the technical capabilities of the computer).

Table 4.1

Example of data composition in a typical DB

Database clients							
Customer	Type	Country	City	Contract Number	Date	Limitation years	Contact Manager
Intersection	com.network	USA	New York	2314589	12.12.2012	2	Aaron
Nevsky comp.	com.network	Russia	Moscow	2337735	15.04.2013	1	Caroline
Perspective korp.	enterprise	Belarus	Minsk	2361112	17.08.2013	2	Daniel
In touch	warehouse	USA	San Francisco	2384723	20.12.2013	2	Alex
Driveway	enterprise	USA	New York	2408570	24.04.2014	5	Aaron
Magnet	com.network	USA	New York	2432656	27.08.2014	3	Alex
Perspective korp.	warehouse	Belarus	Minsk	2456983	31.12.2014	2	Ashley
near	enterprise	USA	Los Angeles	2481553	06.05.2015	2	Ashley
Nori	warehouse	Japan	Tokyo	2506369	09.09.2015	2	Blake
Nardis	com.network	Japan	Tokyo	2531433	14.01.2016	3	Blake

The exact numerical types include the following:

- integer - integer (without fractional part) number. Usually, numbers of this type in the range from - 2147483648 to 2147483647 (four-byte integer);

- smallint - a small integer. Usually, numbers of this type in the range from - 32 768 to 32 767 (two-byte integer);

- bigint - a large integer. The number of bits depends on the SQL implementation and exceeds the number of bits of the integer type;

- numeric (x, y) - a number in which all x digits (accuracy), of which y digits (scale) is assigned to the fractional part;

- decimal (x, y) - a decimal number in which all x digits, of which y digits are assigned to the fractional part. This type is very similar to numeric.

Approximate numerical types include the following:

- real - a real number of single precisions with a floating dividing point (this point "floats", appearing in different places of the number);

- double precision - a real number of double precisions with a floating dividing point;

- float (x) - a real number with a floating dividing point and a minimum accuracy of x, which occupies no more than 8 bytes.

Integer types (integer, smallint, or bigint) are used for table columns that contain different kinds of identifiers (for example, customer codes, products, orders), or quantitative data (for example, number of boxes, packages, pieces). If the column of the table should contain numbers with a fractional part, then for it it is possible to set not integer type. If you are not sure what to use, it is advisable to choose exact (numeric, decimal). They

require fewer resources and give accurate results. If the column is intended to store data from a very wide range (both very small and very large numbers), then use approximate data types (float, real).

String data (character sequences) has three main string types:

- character (n), or char (a string of fixed length), where n is the maximum number of characters contained in the string. If less than n characters are entered, the remaining positions are filled with spaces. If n is not specified, it is assumed that the string consists of one character;

- character varying (n), or varchar (variable length string) is used when the entered data have different lengths and it is undesirable to supplement them with spaces. In this case, the indication of the maximum number of characters is mandatory.

- character large object (clob) is used to represent very large character strings (for example, texts of articles, books, etc.).

The data type is intended for storing date values, the elements of which are arranged in the following order: year (4 digits), hyphen (-), month (2 digits), hyphen, day (2 digits). Thus, the date values occupy 10 positions, for example, 2021-12-22.

There are two types of time representation:

- time without time zone is designed to store time values, the elements of which are arranged in the following order: hour, colon, minute, colon, seconds. Hours and minutes are represented by two digits, and seconds can be represented by two or more digits (if a fractional part is required), for example 19:45:21.653. The length of the fractional part of seconds depends on the implementation, but the internal representation of time must have at least 6 digits;

- time with time zone is the same data type as time without time zone. The only difference is that the time value is supplemented by information about the difference between local and world time (Universal Time Coordinated, UTC, or Greenwich Mean Time). The length of this data type is equal to the length of type time without time zone plus 6, because additional information about the time difference occupies 6 positions (hyphen, sign (+) or (-), 2 digits for hours, colon, 2 digits for minutes).

In SQL, you can represent the date and time at simultaneously. Two types of data are used for this:

- timestamp without time zone (x), where (x) fractional part of seconds. If there is no fractional part, then the data occupy 19 positions: 10

positions for the date, one space and 8 positions for time. If a fractional part is defined, then the data length is 20 plus the number of digits in the fractional part of seconds;

- timestamp with time zone - the same type of data, but to the value of time is added information about the difference between local and world time. Additional information occupies 6 positions.

The difference between two date-time values is interval. SQL supports two interval types: year-month and day-time. A year-month interval is the number of years and months between two dates, and a day-time interval is the number of days, hours, minutes, and seconds between two points within a month.

The interval can be set in two ways: in the form of initial and final moments or in the form of the initial moment and duration. Example:

- interval set by the start and end moments:

time '14:05:50', time '14:35:50';

- interval set by the starting point and duration in hours:

time '14:05:50', interval' 30 ' minute.

In SQL it is also possible to process logical data (data type boolean). This type has three values - true, false and unknown. The value unknown is entered to indicate the result obtained when compared to the value null (indefinite). If the user has not yet entered any value in the table element, then this "empty" element contains a null value, which is interpreted as an unknown or undefined value.

In SQL statements, logical values are enclosed in quotation marks, such as 'true'.

4.3. Transact-SQL

SQL and T-SQL. The SELECT command.

Despite the existence of SQL standards, almost every DBMS from different vendors uses its own dialect of SQL. For the popular Microsoft SQL Server, this language extension is Transact SQL (T-SQL).

T-SQL supports most of the features that are also available in the original SQL version. However, there are some differences, as T-SQL adds: control statements; support for Microsoft Windows authentication; global

and local variables; additional functions for processing dates, strings, etc. As a conclusion, you can embed SQL in T-SQL, but you cannot embed T-SQL in SQL.

A distinctive feature of T-SQL is the use of the SELECT command. This command is a tool that completely abstracts from data presentation issues. It helps focus the user's attention on data access issues. Such approach clearly demonstrates one of the fundamental principles of large (industrial) DBMS: the means of storing and accessing data should be separated from the means of presenting data.

The SELECT command does not change the data in the database, but only selects them in accordance with the specified criteria. The point of the SELECT command is to select rows from one or more tables. In this sense, the command is closed - the result of its execution will also be a table. The command replaces all the relational algebra operators and allows you to form the resulting relation that matches the query.

The SELECT command has the following format:

```
SELECT [predicate]
{* | [column_name [AS new_name]]} [, ... n]
FROM table_name [[AS] alias] [, ... n]
[WHERE <selection_condition>]
[GROUP BY column_name [, ... n]]
[HAVING <group selection criteria>]
[ORDER BY column_name [, ... n]];
```

The parts of the SELECT command shown are called a sentences or phrase.

The SELECT command defines the fields (columns) to be included in the query result. In the list, they are separated commas and are given in the order in which they should be presented in the query result. If the name is used fields containing spaces or delimiters, it must be enclosed in square brackets.

If several tables are being processed, then the full field specification is used in the field list, i.e. Table_name.Field_name.

The SELECT command is executed in strict order, so the sequence of sentences and phrases in the command cannot be changed.

First, the names of the tables used FROM are determined. Next, records are selected from the specified table that meet the specified WHERE conditions. Then the received records are grouped. Using GROUP BY, groups

of rows are formed that have the same value in the specified column. **HAVING** allows you to select a group of rows that meet the specified conditions. **ORDER BY** sorts the records in the specified order. Finally, **SELECT** sets which columns should appear in the output. Only two sentences - **SELECT** and **FROM** - are required, all others can be omitted.

The following **SELECT** example returns all rows (**WHERE** sentence not specified) and all columns (uses an asterisk *****) of the Works database Product table, grouping records by **ASC** name:

```
USE Works;
GO
SELECT *
FROM Production.Product
ORDER BY Name ASC;
GO
```

For a more detailed study of T-SQL and **SELECT**, in particular, it is advisable to refer to special sources.

4.4. Special database objects

Store procedures. Triggers. View.

In a client-server model, business logic is distributed between client and server. On the client's computer, the business logic is implemented in the form of automated operations for processing data of the domain area. On the server, business logic is implemented as *stored procedures*, *triggers* and *views* (Fig. 4.3). These are named blocks of SQL code that are precompiled and stored on the server to quickly execute frequently called query functions.



Stored procedure is a named set of precompiled SQL commands that can be called from a client application or from another stored procedure.

Trigger is a procedure that is executed automatically in response to an event (for example, inserting, changing or deleting data in a table).

View is a virtual (logical) table, the content of which is dynamically calculated based on the data in real tables.

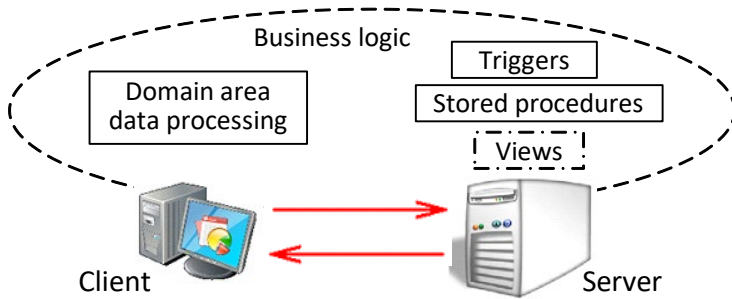


Fig. 4.3. Distribution of elements of business logic between client and server

Stored procedures are pieces of code written in SQL to perform a specific task. The user can explicitly call stored procedures. It's like a subroutine, it can take some input as a parameter, then it can do some processing and return values.

On the other hand, a trigger is a stored procedure that runs automatically when various events occur. Triggers are more like an event handler that fires on a specific event. A trigger cannot accept input and cannot return values. DML triggers run when a user tries to modify data through a data manipulation language (DML) event. DML events are INSERT, UPDATE, or DELETE statements on a table or a view.

Storing and executing code on the server allows you to create code only once, and not in every application working with a database, which saves time when writing and maintaining programs. This ensures that data integrity and business rules are maintained regardless of which client application is accessing the data. Triggers and stored procedures do not need to be forwarded over the network from the client application, which significantly reduces network traffic.

It is advisable to use triggers to prevent certain changes to the database schema, to configure the execution of certain actions in DB in response to changes in the DB schema, to write changes or events to the DB schema in the log. In addition, triggers can be bound not only to a table, but also to a view.

Triggers may be created on views, as well as ordinary tables, by specifying `INSTEAD OF` in the `CREATE TRIGGER` statement. If one or more `ON INSERT`, `ON DELETE` or `ON UPDATE` triggers are defined on a

view, then it is not an error to execute an INSERT, DELETE or UPDATE statement on the view, respectively.

Regarding the view, it should be noted that, unlike ordinary relational database tables, it is not an independent part of the dataset stored in the database. However, changes to the data in the real database table are immediately reflected in the contents of all views built on the basis of this table (Fig. 4.4).

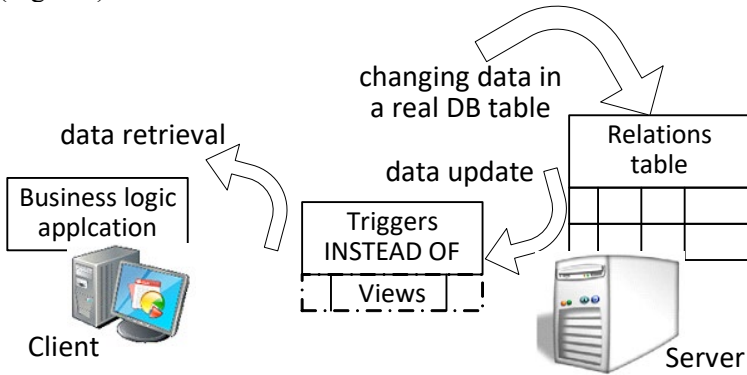


Fig. 4.4. Reflection on changing data in a real database table

The typical way to create views for a DBMS is to link the view to a specific SQL query. Accordingly, the content of the view will be the result of this request. Views are used in database queries in the same way as regular tables. The view name can be in the place of the table name in the SQL query (in the FROM clause).

For many DBMS, such as PostgreSQL, Ms SQL Server, Oracle, a view can contain a subset of records from the database table that meets certain conditions, or a subset of the database table columns required by the program, as well as the result of processing the table data by certain operations. For example, if you have one table "University", you can create two views "Students" and "Teachers", in each of which there will be records only about people of the corresponding position. Or from the real table "Sales" the view can contain for each product only the name and article.

Using views does not provide any completely new possibilities for working with a database, but it can be very convenient. First, views hide the complexity of the queries and the very structure of the database tables from

the application program. The use of views allows you to separate the application schema of data presentation from the storage schema. From the point of view of the application program, the data structure corresponds to the representations from which the program extracts this data. In reality, data can be stored in a completely different way.

In addition, views provide another layer of data protection. The user can be granted only view rights, so that he will not have access to data that is not intended for him, located in the same tables.

4.5. Development of user applications in the 'client-server' environment

Client/server application. Database connection. Database access architecture.

Database connection is one of the most important functions that modern information systems perform. Preparation of SQL queries from the client side to DB on the server can be performed using a dedicated utility. However, to provide the user with great opportunities and convenience in preparing and executing requests, as well as in processing the received data, client applications are created.

In principle, a client/server application consists of a client program that consumes services provided by a server program. The client requests services from the server by calling functions in the server application. Thus, the application always selects a part of the code, or a module responsible for sending requests to the database and processing the responses received from it.

However, this is not enough. To ensure interaction with the database server (with the DBMS), additional tools are required, among which the main ones are:

- DB-LIB interface (DataBase-Library);
- ODBC technology (Open DataBase Connectivity);
- OLE DB interface (Object Linking and Embedding, DataBase);
- DAO technology (Data Access Objects);
- ADO technology (ActiveX Data Objects).

The general modern architecture of access from an application to a database looks like in Fig. 4.5.

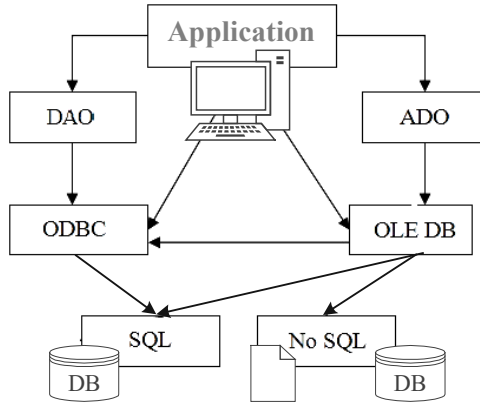


Fig. 4.5. Modern database access architecture

DB-LIB is an optimized application program interface specially designed for database access. This method allows for the fastest access to information. This is because DB-LIB directly uses the SQL-query.

ODBC technologies are designed to provide interconnection between various DBMS. ODBC's capabilities consist in receiving requests from the application to retrieve information, translating them into the kernel language of the addressed DBMS for accessing the information stored in the database. Thus, the main purpose of ODBC is to abstract the application from the features of the back-end database engine.

The OLE DB interface is recommended for building applications and utilities that require high performance. The core capabilities of the OLE DB specification provide complete data access functionality.

When using DAO technology, great convenience in working with database objects is provided. DAO abstracts the entities of the application domain and makes them map to the database, defines the general methods of using a connection, getting and closing it.

Currently, DAO technology is gradually being replaced by ADO technology, which developed by Microsoft. ADO allows you to represent data from a variety of sources (relational databases, text files, and so on) in

an object-oriented manner. In general, ADO technology can be described as the most modern technology for developing Web applications for working with distributed bases of client-server technology.

Test questions and tasks



1. Name the main categories (subsets) of SQL commands.
2. Name the types of data supported by SQL.
3. What is Transact SQL used for?
4. Name the main features using the SELECT command.
5. How is the business logic distributed between the client and the server in the client-server model?
6. Explain also triggers and stored procedures.
7. What is a view and what is it used for?
8. What is required to ensure interaction between the application and the database server?
9. Explain what the ODBC technology is for.



Laboratory work 4. SQL-queries

Laboratory work 5. Complicate SQL-queries for a few tables

Laboratory work 6. SQL-queries for insert, modify and delete rows in database tables

Laboratory work 7. Creation of views, triggers, indexes and store procedures

Laboratory work 8. Transactions.



5. COMMERCIAL AND FREELY DISTRIBUTABLE DBMS

5.1. General characteristics of the DBMS market

DBMS classification. Rating assessments.

The current DBMS market can be classified in three directions - 1) commercialization, 2) relativity of the model and 3) by size (Fig. 5.1).

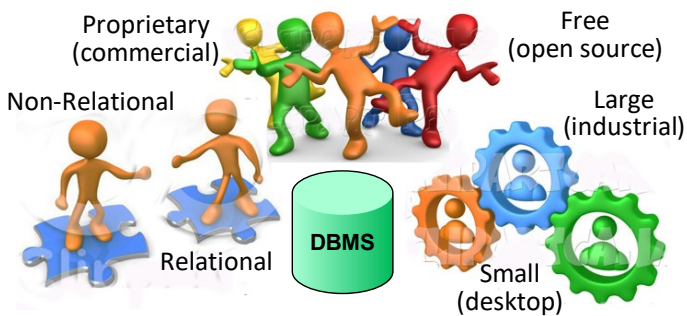


Fig. 5.1. Current DBMS classification

With regard to commercialization, it should first be reminded that proprietary (commercial) products are software for which property copyrights are retained by restricting the right to use them. For freely distributable software, authors and owners allow you to study, modify, and distribute a modified product, often by opening the raw code. The main representatives of the proprietary DBMS are Oracle, Ms SQL Server, DB2. MySQL and PostgreSQL stand out among the freely distributable DBMS.

Among large (industrial, or server) DBMS first of all we see already familiar names - Oracle, Ms SQL Server, DB2, MySQL, PostgreSQL, and also Sybase, MongoDB and others. To date, more than two dozen desktop DBMS formats are known. The most popular are Paradox, FoxPro, Microsoft Access, SQLite.

Most of the above DBMSs are relational using SQL to access data. Recently, however, there has been an emergence of non-relational DBMSs ("No RDBMS", more called "NoSQL"). This is due to the need to create parallel distributed systems for highly scalable Internet applications, such as, for example, search engines. These include MongoDB, Redis, Casandra and others.

A wide range of DBMS provokes their rating assessments. At the same time, from year to year, the top positions are occupied by practically the same products (Fig. 5.2).



Fig. 5.2. TOP popular DBMS

The database management systems market is one of the most conservative in the IT industry. The DBMS is the "heart" of the enterprise IT

system, and it is not changed unless absolutely necessary. Therefore, in the "old" systems, proprietary solutions of the giants of the IT market will prevail for a long time. But new corporate systems are likely to be based on open source DBMS. They are now the most popular with developers.

At the same time, cloud technologies brought a revival to the market. Companies producing "native cloud" DBMS (Amazon Web Services, Alibaba, Google) have achieved great success and in a matter of years were able to harshly increase their market share. However, traditional leaders, primarily Microsoft, Oracle, SAP, are not missing the trend. They also rely on cloud-based versions of their products, incorporating new capabilities into their DBMSs.

5.2. Ms SQL Server

General Information. Services and components. Creating a database.

Familiarity with database management systems should be carried out on the example of Ms SQL Server. First, this database is widely used to support small and medium-sized databases, as well as large databases of enterprise scale. In addition, a lightweight version (SQL Server Express Edition) is used for teaching at many universities.

Ms SQL Server was launched in 1989. At the time, its base code was based on Sybase SQL Server code. This allowed Microsoft to enter the enterprise database market. To eliminate Sybase's claims of copyright infringement, all legacy code in the seventh version DBMS was rewritten by Microsoft.

Ms SQL Server is constantly evolving to work with critical business applications and business intelligence in both traditional data centers and in private, public, and hybrid cloud environments. Each release is marked with a corresponding year and has its own code name. For example, the popular SQL Server 2012 is called Denali.

In addition to development versions, Microsoft releases SQL Server in a variety of versions, which vary in affordability and feature sets depending on end-user goals.

SQL Server has built-in support for the .NET Framework for application development. Due to this, database access procedures can be written in any language of the .NET platform using a complete set of

libraries.

Ms SQL Server consists of individual components that run on Microsoft Windows operating systems as services. A service is a special type of program that runs in OS background.

The Ms SQL Server Installation Wizard lists all the components for a custom installation:

- ✓ Database Engine - for data storage, processing and protection;
- ✓ Analysis Services - for interactive analytical processing;
- ✓ Reporting Services - to create reports;
- ✓ Integration Services - for data integration, transformation and transformation;
- ✓ Full-Text Search - for full-text search;
- ✓ SQL Server Replication - for copying and distributing synchronized data;
- ✓ Service Broker - to create complex applications for communication between disparate databases;
- ✓ Notification Services - to send e-mails;
- ✓ software and utilities;
- ✓ documentation and samples.

The Database Engine component is the main service for storing, processing and protecting data. This component is used to create a database, for online transaction processing or interactive analytical processing. When you start SQL Server, the Database Engine service starts. Users can then establish new connections to the server.

Microsoft SQL Server software tools and utilities allow users, programmers, and administrators to perform the functions that the components described above provide. For example, SQL Server Management Studio is designed to manage the database, used to write Transact-SQL code, multidimensional expressions and XML. In Management Studio, you can develop and manage Database Engine solutions, manage deployed Analysis Services solutions, run and manage Integration Services packages, manage Reporting Services servers, reports, and report models, and process emails.

Thus, SQL Server Management Studio is the main tool for SQL Server administration. It is an integrated environment that provides connectivity, configuration, management, administration, and development for all SQL Server components. Management Studio combines a large set of

graphical tools and a number of powerful script editors, as well as provides access to SQL Server for developers and administrators at all levels.

The database creation procedure is usually only done the database administrator. This procedure includes such steps as directly creating a database, as well as creating a transaction log belonging to the database. The first stage creates a file with *.mdf extension for main files and a file with *.ndf extension for secondary files. At the second stage, a file with the *.ldf extension is created.

To create a new database SQL Server Management Studio is used. In the Object Explorer window, the server destination is selected and the New Query item is selected in the menu that appears (Fig. 5.3).

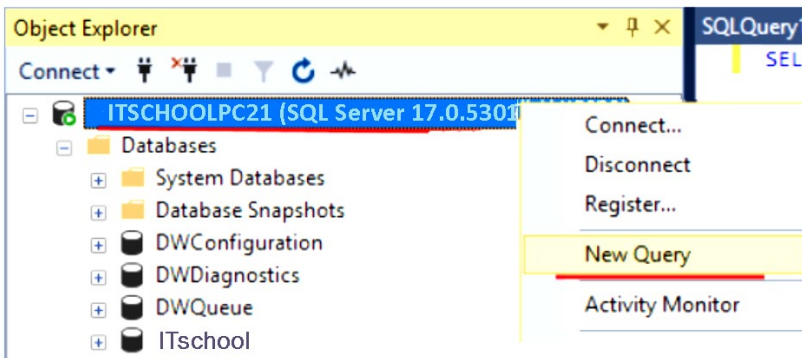


Fig. 5.3. Using SQL Server Management Studio to create a database

To create a database named "usersdb" in the central field for entering SQL expressions, enter the CREATE DATABASE command (Fig. 5.4) and click on the Execute button on the toolbar (or press the F5 key). After that, a new database will appear on the server.

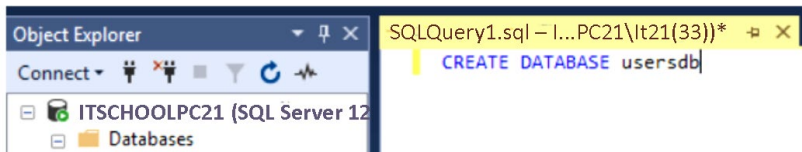


Fig. 5.4. Using the CREATE DATABASE command

The database file (*.mdf) can be transferred between computers. However, if you copy it to a computer with MS SQL Server installed, the database will not appear just like that. To do this, you need *to attach* the DB to the server. In this case, the CREATE DATABASE command also applies:

```
CREATE DATABASE database_name
ON RIMARY (FILENAME='path_to_mdf_file_on_local_computer')
FOR ATTACH;
```

As a directory for the DB, it is advisable to use the directory where the rest of the server databases are stored. On Windows 10, the default directory is C:\Program Files\Microsoft SQL Server\MSSQL13.MSSQLSERVER\MSSQL\DATA.

To drop a database (or several databases at once), use the DROP DATABASE command:

```
DROP DATABASE database_name1 [, database_name2] ...
```

If the database to be removed was attached, all files of this database will be deleted anyway.


Most of the actions to change the database configuration are done using the command ALTER DATABASE database_name.

The command syntax assumes several parameters. For example, the ADD parameter adds new files to the database, and the REMOVE parameter is used to remove files or groups of files from the database.

5.3. Freely distributable DBMS

MySQL. Creating a database. PostgreSQL. Client-server architecture.

The most famous of the freely distributed DBMSs that support SQL are MySQL, PostgreSQL, and Firebird. Apache Cassandra, CouchDB, MongoDB are popular among "non-SQL systems".

 MySQL is a very popular, fast, multi-threaded, multi-user relational database server that supports SQL. MySQL is free software and open source. MySQL is licensed under the GNU General Public License (<http://www.gnu.org>). Up-to-date information on MySQL is provided by <https://www.mysql.com>.

MySQL was developed by the Swedish commercial company MySQL AB. In 2008, MySQL AB was bought for a billion dollars by Sun Microsystems and is currently owned by Oracle Corporation.

MySQL is one of the components of LAMP technology (Linux, Apache, MySQL, PHP).

MySQL was originally designed to handle very large databases much faster than existing solutions. MySQL has long been used successfully in large industrial environments. Further development of MySQL in the areas of connectivity, performance and security have made this server very convenient for supporting DBs on the Internet.

MySQL is a "client-server" system that supports various client programs and libraries, administrative toolkits and multiple programming interfaces. MySQL also includes some extensions to the SQL92 ANSI standard.

The attractiveness of MySQL is due to three main advantages.

First of all, it is flexible and easy to use. The user can change the source code to meet his own expectations, without having to pay anything for this level of freedom. The server installation process is relatively simple and usually takes no more than 30 minutes.

Secondly, it is high productivity. MySQL supports a wide range of servers. Whether you want to store large amounts of e-commerce data or perform heavy business analytics, MySQL performs these tasks at optimal speed.

In the end, this is security, because data security is the main criterion for choosing a DBMS software. With an account access and management system, host-based authentication and password encryption, MySQL provides a high level of security.

If you carry out a large detailed analysis, then it will appear that MySQL has a number of features that bring this DBMS to the forefront. This includes the ability to run on many different platforms, and support for interfaces for different programming languages, and the ability to mix tables from different databases in one query, and processing tables in memory, and much more. We should also mention the support for truly huge amounts of data. The latest MySQL versions allow a maximum table size of up to 8 million terabytes (2^{63} bytes). There is a case of using MySQL for 60,000 tables that store about 5,000,000,000 rows.

The following system requirements are required to install MySQL on

a local computer: 128 MB of RAM; 200 MB of free disk space; Windows (x86 / x64) or Linux.

When installing MySQL to support full-fledged administration, table filling, and data access, you must select the following components:

mysql (directly mysql itself);

mysql gui tools (for administration and creation of new users);

mysql connector odbc (component through which the programming environment can access data);

mysql-front (for filling, creating, and viewing user tables).

To create a new database and the user to continue working with it, first configure the administrator account in MySQL-Front (Fig. 5.5), and then proceed to create a new database. Next you need to create one or more users to access the database. Users are entered by the administrator. To do this, he uses mysql gui tools (Fig. 5.6).

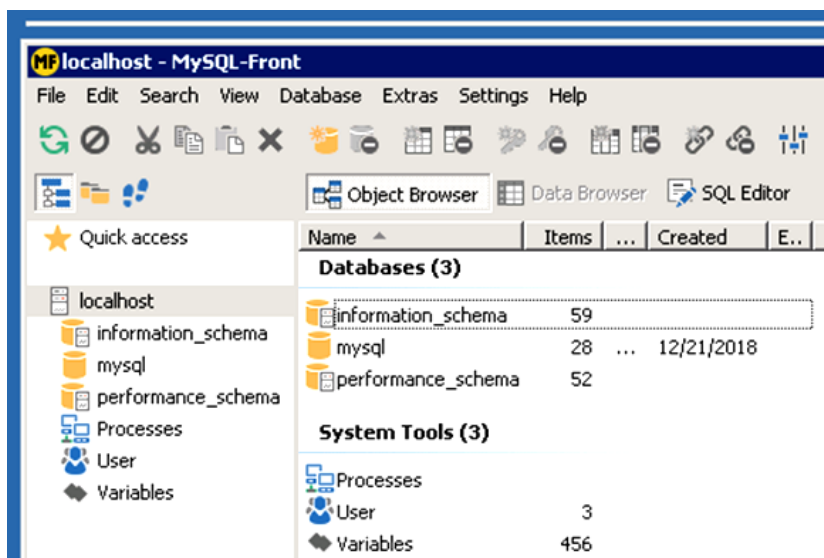


Fig. 5.5. Configure the administrator account in MySQL-Front

Then the user can create tables in the database using the graphical interface of the CREATE TABLE command (Fig. 5.7) as well as perform other operations.

S1. BASIC CONCEPTS OF DATABASE SYSTEMS

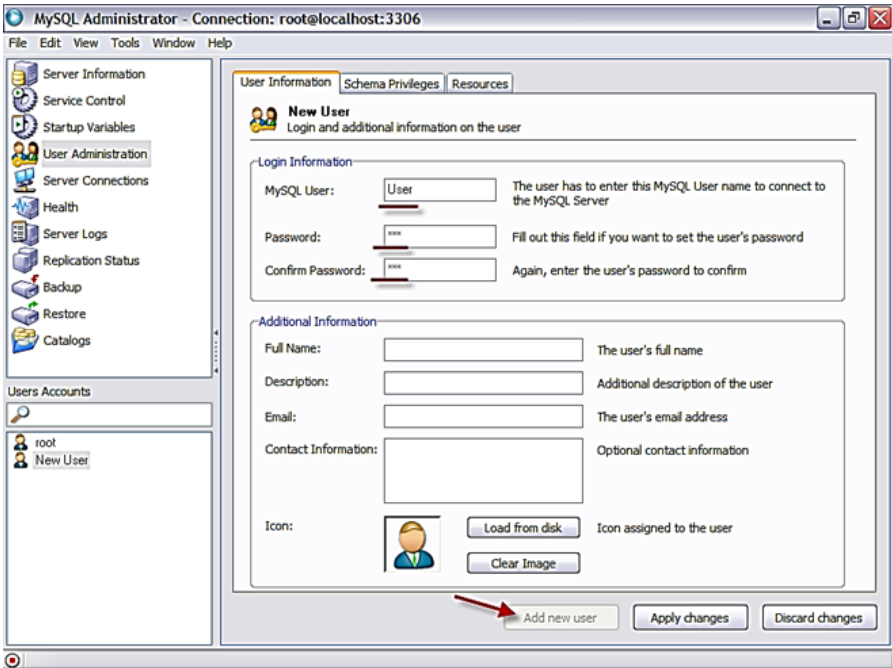


Fig. 5.6. Admission by the database user

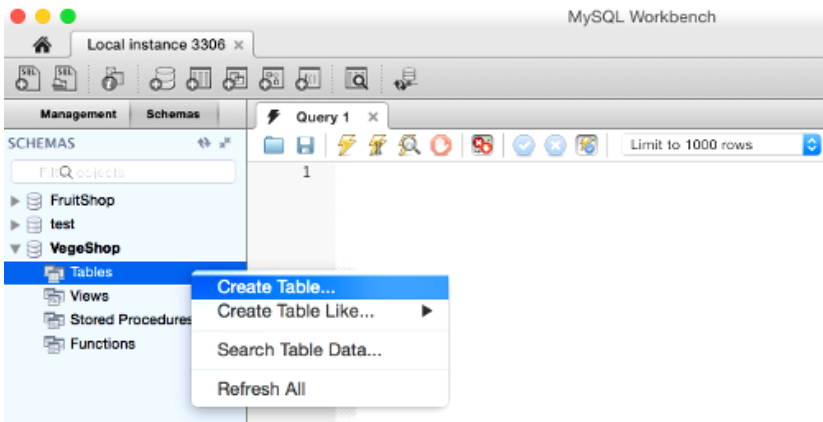


Fig. 5.7. Creating a table in the database using the graphical interface of the CREATE TABLE command

Another example of the success of freely distributed software is PostgreSQL. Unlike other similar DBMS, a single company does not control PostgreSQL. PostgreSQL development has been made possible by the collaboration of many people and companies. A prototype called POSTGRES was developed at the University of California, Berkeley in 1987 with the participation of Michael Stonebreaker, after which it was actively developed and supplemented.



Currently, PostgreSQL is used to implement large systems, such as financial data analysis systems, asteroid tracking, medical information, geographic systems, and more. PostgreSQL is also used as a learning tool in universities.

A PostgreSQL server written in C is usually distributed as a set of open-source text files. To install, you need to compile the files and collect them to some directory. The whole process is detail described in documentation. By the way, the documentation on PostgreSQL is quite a lot and is issued by many universities.

PostgreSQL is focused on most of the SQL standard and supports many modern DBMS functions. At the same time, users can expand the capabilities of PostgreSQL, creating their own data types, functions, operators, procedural languages, etc.

PostgreSQL is implemented in a client-server architecture. The working session includes the following interacting processes (programs):

- the main server process (called postgres), which manages DB files, accepts client application connections and performs various client requests to databases;
- a client application of a user who wants to perform operations in DB.

Clients can be very diverse. Some client programs come with the PostgreSQL distribution, but of course, most are created by third-party developers.

A PostgreSQL server can handle multiple client connections at the same time. For each connection, the main postgres process generates a separate server process. In this way, the client and this process interact without affecting the postgres process. One connection cannot access more than one DB. However, the client is not limited in the number of connections to the same or to different databases.

The PostgreSQL server can manage multiple databases, which allows

you to create separate databases for different projects and users running a single DBMS. However, DBs must have unique names that begin with a letter and be no longer than 63 characters. It is convenient to use the user's name as the database name.

Before a DB creating, the administrator must create a PostgreSQL account for the user who will create the database, and give the user the appropriate permission (Fig. 5.8).

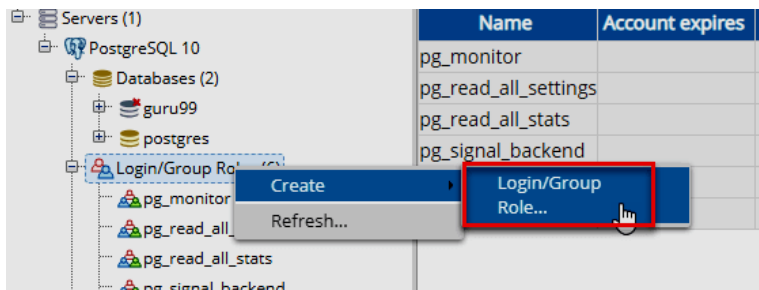


Fig. 5.8. Creating a PostgreSQL account for the user

Having created a database, you can access it in different ways, namely:

- run a terminal program called `psql`, in which you can interactively enter, edit and execute SQL commands;
- use existing graphical tools, such as pgAdmin;
- use an office suite with ODBC or JDBC support, which allows you to create and manage databases;
- write and use your own application through one of the many available language interfaces.



ODBC (Open Data Base Connectivity) is an open-source database application programming interface developed by the X/Open consortium.

JDBC (Java Data Base Connectivity) - connection to databases in Java

If to connect by means of `psql` it is necessary to enter the command:
`$ Psql [db_name]`

If you do not specify a database name, it will be selected by username.

5.4. NoSQL DBMS

Fundamental characteristics. Advantages and disadvantages.

At the end of 2010, the term “NoSQL” became very popular. All kinds of software solutions began to actively develop and promote under this flag. These decisions were associated with huge amounts of data, linear scalability of the database, clusters, fault tolerance, and also non-relationality of the database. However, the term “NoSQL” rather characterizes the vector of IT development away from relational databases.

It should be emphasized that the term “NoSQL” has an absolutely spontaneous origin and does not have a generally accepted definition. It stands for “Not Only SQL”, although there are supporters of the direct definition of “No SQL”. The NoSQL label now hides a lot of heterogeneous systems, but there are few common characteristics for all NoSQL (Fig. 5.9). Many characteristics are unique to certain NoSQL databases.



Fig. 5.9. Variety of NoSQL databases

What are the main distinguishing features of a NoSQL database?

First, it is not using SQL (meaning ANSI SQL DML). Many DBMSs try to use query languages similar to SQL syntax, but no one has yet been able to fully implement it. The rejection of SQL is related to the second feature – the unstructured database (schemaless). In NoSQL databases, unlike relational databases, the data structure is not regulated (weakly

typed). In a separate line or document, you can add an arbitrary field without first declaratively changing the structure of the entire table.

The consequence is the third feature – the presentation of data in the form of aggregates. The relational model stores the logical business entity of the application in various tables that are normalized. NoSQL storages operate with these entities as with integral objects - aggregates (Fig. 5.10).

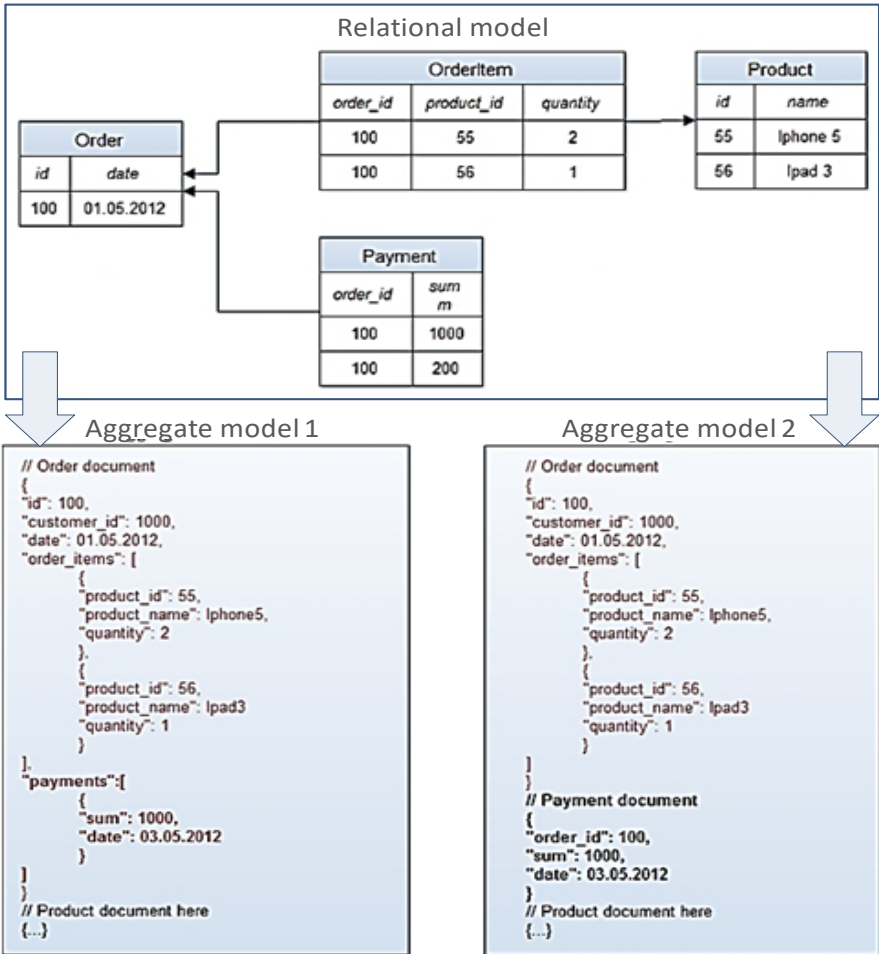


Fig. 5.10. Using aggregates in a NoSQL database

The example shows a standard relational trade model (“order - order items - payment - product”) and two aggregate models. In both models, the order is combined with the order items into one logical entity. Moreover, each position order items stores a link to the product and some of its attributes. In one aggregate, payment is combined with the order and are an integral part of the object, in the other, they are placed in a separate object. This demonstrates the main rule of designing a data structure in NoSQL databases – it must obey the requirements of the application and be maximally optimized for the most frequent queries.

The aggregate model, like the relational model, is not without its drawbacks. However, its benefits appear in a distributed environment and in large data, providing faster read speeds. It is in such environments of fast distributed clusters - financial, online shopping, ticket booking - that high requirements are imposed on the atomicity and consistency of operations. Transactions must be executed at the highest rate so that the maximum interval during which the user can see inconsistent data is no more than a second. The fulfillment of the conditions of “eventual consistency” should prevent cases when, for example, client B orders the same ticket that client A had already ordered a moment ago.

This is perhaps the main theme in the development of NoSQL databases. With the explosion of information and the need to process it in a reasonable amount of time, the issues of speed, scalability, and throughput have come to the fore. The only way out of the situation is to connect a fast network of several independent servers, and each server processes only a part of the data or only a part of requests for reading-updating the database. The NoSQL DBMS itself deals with sharding procedures, replication, ensuring fault tolerance, redistributing data in case of adding nodes.



Sharding – separation of data by nodes (servers) in the network.

Replication – copying data to other nodes when updating the database.

Thus, NoSQL databases are optimized for applications that need to quickly processing large amounts of different data structures with low latency. Taking into account today's trends, non-relational storages are directly focused on Big Data (Fig. 5.11).

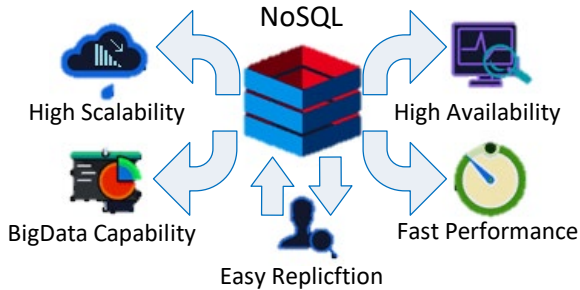


Fig. 5.11. Features of NoSQL DBMS important for Big Data

All NoSQL solutions are usually divided into 4 types.

Key-value is the simplest option, using a key to access a value within a large hash table. Such DBMS are used to store images in scalable Big Data systems, in Internet of Things (IoT) projects. The most famous representatives are Oracle NoSQL Database, Berkeley DB, MemcacheDB, Redis, Riak, Amazon DynamoDB.

In **document-oriented** storage, data is represented as a semi-structured document of tagged elements, like XML and others. The most common use of documentary NoSQL is in content management systems (CMS), publishing, and documentary search. The most prominent examples are CouchDB, Couchbase, MongoDB, eXist, Berkeley DB XML.

Column storage contains information in the form of a sparse matrix, the rows and columns of which are used as keys. In the world of Big Data, “Column Family” databases are referred to as column storages. In such systems, values are stored in columns presented in separate files. Thanks to this data model, it is possible to store a large number of attributes in a compressed form, which speeds up the execution of queries to the database. This makes it possible to use such DBMS for registering and processing events in exchange analytics systems, IoT applications, etc. The most famous columnar database is the Google Big Table, as well as Apache HBase and Cassandra based on it.

Graph storage is a networked database that uses nodes and edges to display and store data. Since the edges of the graph are stored, traversing it does not require additional calculations (like a join in SQL). Such DBMSs are used in communication-oriented tasks – social media, fraud detection,

public transport routes, road maps, network topologies. Examples of graph bases: InfoGrid, Neo4j, Amazon Neptune, OrientDB, AllegroGraph, Blazegraph, InfiniteGraph, FlockDB, Titan, ArangoDB.

Just because the NoSQL movement is gaining popularity at an enormous rate does not mean that relational databases are becoming archaic. Most likely they will still be used actively, but more and more in symbiosis with NoSQL databases. The choice of storage will come from the nature of the data itself, from what amounts of data and how we want to manipulate them.

Test questions and tasks



1. In what areas can the modern DBMS market be classified?
2. Describe Ms SQL Server.
3. Describe the purpose and capabilities of SQL Server Management Studio.
4. What are the stages of creating a database in a SQL Server environment.
5. Name the most famous freely distributable DBMS.
6. Describe the MySQL DBMS.
7. Describe the PostgreSQL DBMS.
8. What are the main distinguishing features of a NoSQL database?
9. What types are customary to divide NoSQL solutions into?



Laboratory work 9. Access rights management

Laboratory work 10. Upload and download database. Replication.

Laboratory work 11. Creation and using of non-SQL database (in Redis environment)



LITERATURE

1. Hlushkov V. M. Osnovy bezbumazhnoi informatyky. M.: Nauka, 1982. 552 c.
2. Nesterenko O.V. Informatsiini systemy upravlinnia pidpriumstvamy. Navch. posib. Kyiv: UkrNTs RIT, 2019. 135 c.
3. Nesterenko O.V., Savenkov O.I., Falovskyi O.O. Intelktualni systemy pidtrymky pryiniattia rishen. Navch. posib. / Za red. Bidiuka P.I. Kyiv: Natsionalna akademiia upravlinnia, 2016. 188 c.
4. Rudenko V. D. Bazy danykh v informatsiinykh systemakh. Navch. posibnyk / za zah. red. V. Yu. Bykova. Kyiv: Feniks, 2010. 240 c.
5. Pasichnyk V.V., Reznichenko V.A. Orhanizatsiia baz danykh ta znan. Kyiv: Vydavnycha hrupa BHV, 2006. 384 c.
6. Pasichnyk V. V., Shakhovska N. B. Skhovyshcha danykh. Navch. posib. / za red. V. V. Pasichnyka. Lviv: Mahnoliia, 2008. 492 c.
7. Christopher Date. An Introduction to Database System. Pearson Education, 2004. 1034 p.
8. Gruber Martin. Understanding SQL. San Francisco: Sybex, 1990.
9. Viescas John L., Hernandez Michael J. SQL Queries for Mere Mortals: A Hands-On Guide to Data Manipulation in SQL, 3rd Edition. 2014, Addison-Wesley Professional. 800 p.
10. Ward Bob. SQL Server 2019 Revealed: Including Big Data Clusters and Machine Learning. 2020, Apress. 480 p.
11. Kleppman Martin. Designing Data Intensive Applications. O'Reilly, 2014. 144 p.
12. Simkovics Stefan. Enhancement of the ANSI SQL Implementation of PostgreSQL. Department of Information Systems, Vienna University of Technology. 1998. Vienna, Austria.

13. A. Yu and J. Chen. The Postgres95. User Manual. 1995. University of California. Berkeley, California.
14. Zelaine Fong. The design and implementation of the POSTGRES query optimizer. University of California, Berkeley, Computer Science Department.
15. www.mysql.com
16. www.postgresql.org
17. www.mongodb.com
18. redis.com

Навальний посібник

**Фаловський Олександр Олександрович
Нестеренко Олександр Васильович**

Основи пректування та використання баз даних

(англійською мовою)

Підп. до друку 17.01.2023 Формат 60x84/16.
Гарнітура Futura Book. Папір офс. Друк офсет.
Обл.-вид. арк. 4,10, Ум.-друк. арк. 5,25.
Наклад 350 прим.

Видавництво ТОВ «Тропеа».
Свідоцтво про внесення суб'єкта видавничої справи
до Державного реєстру — серія ДК № 7025 від 24.12.2019 р.
01001, Київ, вул. Мала Житомирська 10, нежиле приміщення 60, літ. А.

Віддруковано у друкарні ТОВ «Про формат».
Україна, 04080, м. Київ, вул. Кирилівська, 86.
Свідоцтво про внесення суб'єкта видавничої справи до державного
реєстру ДК № 5942 від 11 січня 2018 р.