Organization of Data bases

semester 4

lector: Lyudmyla Rozova, associate professor, phd

email: lyudmyla.rozova@khpi.edu.ua

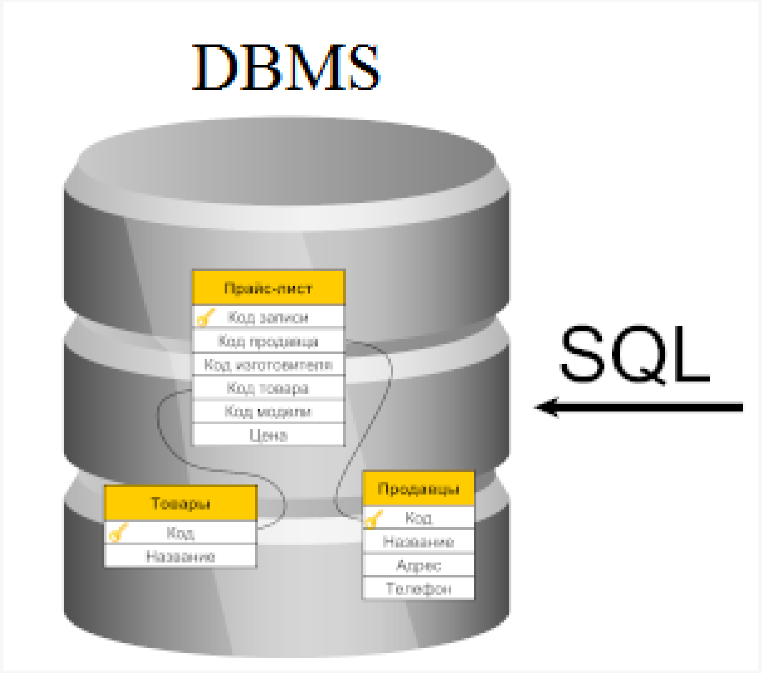
Lecture 3.

1 SQL language

2 Basic operators of relational algebra

SQL language

Relational algebra consists of different operators that use relations as operands. The result of operation is also relation.
Relational operations are implemented in **SQL (Structured Query Language).**
SQL is standard of ANSI. The Microsoft Access DBMS also uses Access SQL (Jet SQL), which is slightly different from the standard version, but the basic operators and rules are standard.



SQL language

Like any programming language, **SQL** has its own rules for writing instructions.

The main ones are:

- A comma is used to separate items in lists.
  For example, a list of field names.
- Use square brackets to specify field names that contain invalid characters (such as space). For example, [Date of Birth].
- If the query includes fields from multiple tables, the full field name is used, which consists of the table name and the field name, separated by a point. For example, Students.Last name.
- Character strings are enclosed in ‘***apostrophes***’ or quotation marks.
- At the end of the instruction, put a semicolon **(;)**.
- In SQL statements that are split across multiple lines, you can use indentation to indicate the continuation of the previous line.

SQL language

The main type of SQL query is SELECT - select query.

Its general view:

SELECT <list of field names> (if all fields, then *)

FROM <table name>

[WHERE <selection conditions>] (you can use <,>, =,

BETWEEN, AND, NOT, OR)

[GROUP BY <field names>] (grouping)

[ORDER BY <field names>] (sort)

SQL language

Examples:

Ex.1. Output the name, surname, address and telephone number from the table Students, sorted by last name:

```
SELECT Surname, First name, Address, Telephone
FROM Students
ORDER BY Last name;
```

Ex.2. Output all fields of the Students table, grouping by groups

```
SELECT * FROM Students
GROUP BY Group;
```

Ex.3. Remove all students from the table Students living on Park street

```
SELECT *
FROM Students
WHERE left ([Address], 9) = 'Park street'; 4)
```

Ex.4. Select students from the table Students who were born in September 2002

```
SELECT *
FROM Students
WHERE [Date of Birth] between 31.08.02 and 1.10.02;
```

2. Basic operators of relational algebra

Relational algebra defined by Codd consists of 8 operators.

They can be divided into two groups:

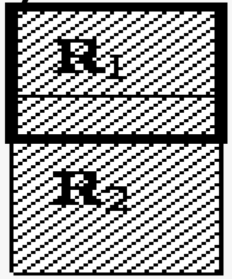
- relational operations, similar to traditional **set operations**,
- and relational operations actually.

For all relational operations, **the closure property** must be satisfied,

→The result of every operation on relations must be a relation.

2. Basic operators of relational algebra. Relational operators similar to traditional set operations

1) **Union.**



The result of the union of relations $\mathbf{R_1}$ and $\mathbf{R_2}$ is a relation $\mathbf{R_3}$ containing all corteges that belong to at least one of $\mathbf{R_1}$ and $\mathbf{R_2}$.

Unlike union of sets, the result is not a set of corteges, but a relation. Therefore, corteges must be homogeneous. The relations that used in Union operation must be compatible by type.

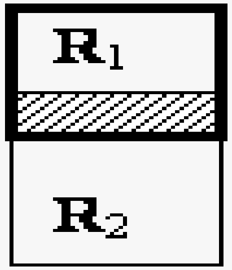
It means that: each of relations has the same set of attributes; the corresponding attributes are defined in the same domain.

In SQL: **R_1 UNION R_2**

For example: let's say there are separate relations between Laboratory assistants and Teachers in the Faculty database. These two tables can be combined, the result is a relation containing all the data on faculty members: both teachers and laboratory assistants.

2. Basic operators of relational algebra. Relational operators similar to traditional set operations

2) **Intersection**



The intersection of the relations $\mathbf{R_1}$ and $\mathbf{R_2}$ results in the relation $\mathbf{R_3}$ containing corteges which belong to both $\mathbf{R_1}$ and $\mathbf{R_2}$.

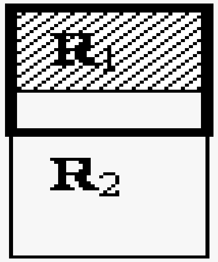
This operation must also satisfy the ***type compatibility condition***.

In SQL: R1 Intersect R2

For example: let that there are relations in the Faculty about Laboratory assistants and Students. With this operation, you can find those students who work as laboratory assistants.

2. Basic operators of relational algebra. Relational operators similar to traditional set operations

3) **Subtraction**



The result of **subtraction** the relation **R_2** from the relation **R_1** is the relation **R_3**, all corteges of which belong to **R_1** and do not belong to **R_2**. .

This operation must also satisfy the ***type compatibility condition***.

In SQL: R1 Minus R2

For example: from the same relations *Laboratory Assistants* and *Students* in the *Faculty* database, you can find out which students are not working as laboratory assistants in the *Faculty* and vice versa.

Relational operators similar to traditional set operations

4) **Multiplication** (Cartesian)

R_1		R_2		R_3	
a		x		a	x
b		y		a	y
c				b	x
				b	y
				c	x
				c	y

The multiplication of the relations **R_1** and **R_2** is the relation **R_3** .

Relation **R_3** contains ***all possible corteges***, which are the combination of two corteges, belonging to the relations **R_1** and **R_2**, respectively.

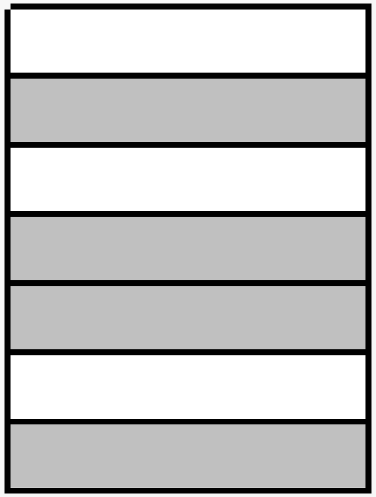
In SQL: **R_1 TIMES R_2**

The result is a relation, not a set of pairs of corteges. The initial corteges are concatenated, the result is a new cortege.

For example, if you perform the product of the relations between *Students* and *Disciplines,* then we will get a relations that will contain information that each student studies each discipline.

2. Basic operators of relational algebra. Relational operators exactly

5) **Selection** (by limiting operation).



The result of the selection applied to the relation **R_1** is the relation **R_2**, which contains all corteges of the relation **R_1** that satisfy certain conditions.

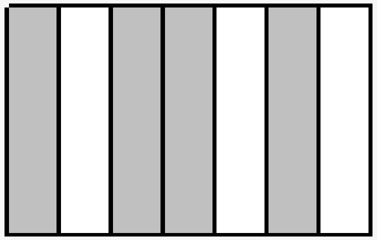
We can say that this is a ***"horizontal" subset*** of the initial relation.

In SQL: **R_1 WHERE xθy** (θ (theta) – any comparison operator: <,>, =, ...)

For example, Output data about students who study in the INF-31 group from the relation Students (Gradebook No., Surname, Name, Patronymic, Group Code)

2. Basic operators of relational algebra. Relational operators exactly

6) **Projection**



The result of the **projection** applied to the relation **R_1** is the relation **R_2,**
R_2 contains all **R_1** corteges after excluding some of the attributes from it.

Such corteges are called ***subcorteges***. We can say that this is a ***"vertical" subset*** of the initial relation.
There is no special SQL operator for projection. It is enough to indicate in the standard ***SELECT query*** which attributes of the relation are taken.
Let's pay attention to special cases:
it is possible to specify a list of all attributes of the original relation - this is an **identical** projection;
it is possible to specify an empty list of attributes - this is a **zero** projection.
For example, Output information for all students only about the last name, first name, and group from the relation Students (Gradebook No., Surname, First name, Date of birth, Address, Phone, Group).

2. Basic operators of relational algebra.

Relational operators exactly

7) Join

Common case						Private case				
R_1			R_2			R_1			R_2	
a	x		x	l		a	x		x	l
b	y		y	m		b	y		y	m
c	z		z	n		c	z		y	n
		R_3						R_3		
	a	x	l				a	x	l	
	b	y	m				b	y	m	
	c	z	n				b	y	n	

The result of **joining** of relations $\mathbf{R_1}$ and $\mathbf{R_2}$ is the relation $\mathbf{R_3}$, the corteges of which are the concatenation of two corteges (belonging to $\mathbf{R_1}$ and $\mathbf{R_2}$, respectively) that have a common value for one or more common attributes $\mathbf{R_1}$ and $\mathbf{R_2}$.

Moreover, these common values appear only once in the resulting relations

In SQL: $\mathbf{R_1}$ **JOIN** $\mathbf{R_2}$

2. Basic operators of relational algebra. Relational operators exactly

The **joining** has the properties:

- associativity: (R1 JOIN R2) JOIN R3 = R1 JOIN (R2 JOIN R3);
- commutability: (R1 JOIN R2) JOIN R3 = R1 JOIN R2 JOIN R3.

This operation has several varieties, but the most common is natural connection (in the diagram). There is also the θ (theta) compound. It is intended for cases where two relations are connected based on some conditions (**xθy**) other than equivalence.

In this case, in SQL: **(R_1 TIMES R_2) WHERE xθy,** combination of multiplication and selection.

An example of a natural join: joining of relation between Students and Groups by the Group attribute. The result is a relationship containing information about students and, for each student, about his group.

An example of a θ-join: joining of relation Students and Groups by the Group attribute but only for 5th year groups.

2. Basic operators of relational algebra. Relational operators exactly

8) **Division**

R_1			R_2
a	x		x
a	y		y
a	z		
b	x		R_3
c	y		a

The definition for a particular case:
for the relations **R_1** (binary) and **R_2** (unary), the result of ***division*** is the relation **R_3**, which contains all the values of one attribute **R_1** that correspond to all the values of **R_2** in the other attribute.

For relations with a large number of attributes, the same is true.

In SQL: **R_1 DIVIDEBY R_2**

For example, in the Faculty database there is a relation Lessons (Group, Discipline, Teacher). To get a list of groups that study a given set of disciplines, you can apply the division of this relationship into a specially created unary relationship containing a given set of disciplines.