Organization of Data bases

semester 4

lector: Lyudmyla Rozova, associate professor, phd

email: lyudmyla.rozova@khpi.edu.ua

Lecture 3.

1. SQL language
2. Additional operators of relational algebra

Additional operators of relational algebra

Several additional operators have been added to the eight basic operators of relational algebra.

They are convenient to use for practical purposes, although they can be implemented through the main ones.

The most successful addition to the basic operators:

extend and summarize.

EXTEND operation

With its help, a new relation is created from the original relation, containing an additional attribute, the values of which are obtained by the use of some scalar calculations.

EXTEND <relation name> ADD (<scalar expression >) AS< new attribute name>

Additional operators of relational algebra

Examples:

1)Let there is a **relation Teachers** (***Teacher Code, Surname, First name, Date of Acceptance***). For each teacher, output the work experience.

EXTEND Teachers ADD (year (date ()) - year (*AcceptanceDate*)) AS *Experience*

2)A special case is the addition of a new attribute filled with the same values: in the relation of ***SportSections*** of the ***Faculty*** database, add the attribute ***Location***, filling it with the same values for all sections - " ***SportsComplex*** ".

EXTEND *SpotSections* ADD ("*SportsComplex* ") AS *Location*

3)A special case is renaming the attribute ***Address*** of the relation **Students** to ***RegistrationAddress***

EXTEND *Students* ADD (Address) AS *RegistrationAddress*

Additional operators of relational algebra

SUMMARIZE operation

If the extend operation allows "horizontal" computation, then the summarize operation alows "vertical" computation. This operation makes it possible to perform group operations by attributes (count the number, amount of records, etc.)

SUMMARIZE < relation name > BY (<list of attributes' name>)
ADD< group operation > AS < field name for the total value>

<u>Group operations can be:</u>

sum (<**attribute name**>) - the sum of numeric values;
count (<**attribute name** >) – the number of values;
min (<**attribute name** >) – minimum value;
max (<**attribute name** >) – maximum value.

Additional operators of relational algebra

Examples:

1)In relation **Students** (**GradebookNo, Surname, Name, GroupCode**) count the number of students in each group.

SUMMARIZE *Students* BY (*GroupCode*) ADD count (*GradebookNo*) AS *StudentsNumber*

2)Let there is an **Lessons** relation (***GroupCode, Discipline, Teacher***). Count the number of groups that study more than 10 disciplines

(SUMMARIZE *Lessons* BY (*GroupCode*) ADD count (*Discipline*) AS N) WHERE N> 10

Additional operators of relational algebra

Operations of updating include **insert, update, and delete** operations.

Operation: INSERT (<relational expression or list of attributes>) INTO < relation name >

Examples:

1)Insert a new record in the relation ***Students***.

INSERT (GradebookNo=123456; Surname = "Ivanov"; First name = "Ivan") **INTO** Students

2)In the relation ***StudentsGroup31*** insert those records from the relation ***Students*** that correspond to the students of the group INF-31.

INSERT (Students **WHERE** GroupCode = "INF-31") **INTO** StudentsGroup31

The purpose of relational algebra

The main purpose of relational algebra is to provide a **record of expressions**. And relational expressions aren't limited to queries.

Examples include the following uses of expressions:

• the definition of the *selection area* (WHERE ...);

• the definition the *updating area* (data to insert, change, or delete);

• the definition of *integrity rules*;

• the definition of *security rules* (data for which access control is performed).

So, we can say that expressions denote a symbolic record of the intentions of a database user, and this symbolic record exists in the SQL language.

Language is called relationally complete if its capabilities match the capabilities of relational algebraic operations.

Practice 4

Building queries in SQL mode using SQl-instructions.
Task 1 and Task 2. Create Queries in SQL mode
To select data from the database, the SQL (Structured Query Language) language is used.
SQL is a programming language that closely resembles English, but is designed for database management programs.
Every executed query is actually SQL based.

Practice 4

Task 3 and Task 4. Create form.

Introduction:

Form is a database object that can be used to enter, modify, or display data from a table or query. Forms are used to control access to data, such as determining which fields or rows of data should be displayed.

An efficient form makes working with the database faster because users do not need to search for the information they need. The attractive form makes working with the database not only efficient, but also more enjoyable.

In addition, forms can prevent invalid data from being entered.

Practice 4

Types of forms:

1. Form for entering and modifying data:
   - Form with one element;
   - Divided form;
   - Form for multiple elements (strip form);
   - Composite form (main and subordinate, with a one-to-many relation).
2. Summary table (chart).
3. Navigation form.
4. Dialog box.

A form can be created using a variety of tools found on the **Create tab** in the **Forms group** (depending on the type and complexity of the form you are creating).