

# **Database Solutions**

## **Basic terminology**

**Marzena Nowakowska**

**Faculty of Management and Computer Modelling**

**Kielce University of Technology**

**room: 3.21 C**

# The subject completion

**WWW lecture sources available via the page:**

<https://staff.tu.kielce.pl/spimn/>

then click the link *Materiały dydaktyczne* in the main menu and *Database solutions* from the bulleted list

**The *Database Solutions* syllabus:**

<https://wzimk.tu.kielce.pl/wzimk/studia/programy-obowiazujace-od-2019-20/inzyniera-danych/>

then click the link *Sylabusy* in the main window panel and find the document in the *Semestr 7* part

**Bibliography** – in the Internet, use Google (or other search engine) and look for the following terms:

- Database
- Database Management System
- Database relational model
- Normalisation / normalization
- SQL, select queries, action queries

## At the end of the semester

- **A test**
  - ✓ key terms understanding
  - ✓ review questions
- **A students' presentation**

# The main objective of the subject

- The main objective of the subject is to consolidate and develop a student's knowledge on database fundamentals as well as to teach the student a basic database terminology.
- Appropriate examples are given to illustrate each merit topic.
- Students are encouraged to participate in a discussion during the lecture.
- *The order of the lecture content does not have to be in accordance with the order of their presentation in the syllabus. However, all substantive issues will be considered.*

# Definition of a database

## **Database**

A self-describing collection of logically related data of various types that are registered onto and accessed from storage memory devices (disks, pen-drives, memory cards etc.).

## **Self-describing collection**

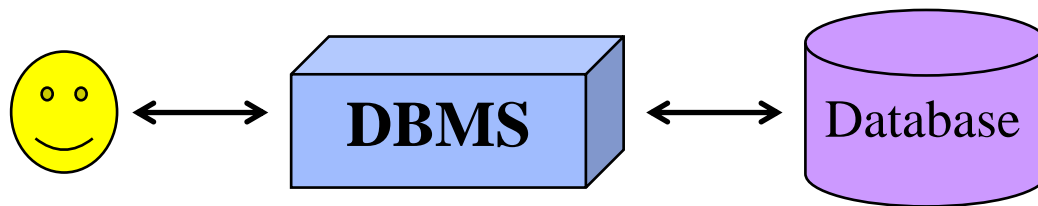
In addition to a users' data, a database contains the information about its own structure – this description is called *data dictionary* or *metadata*.

# Database Management System

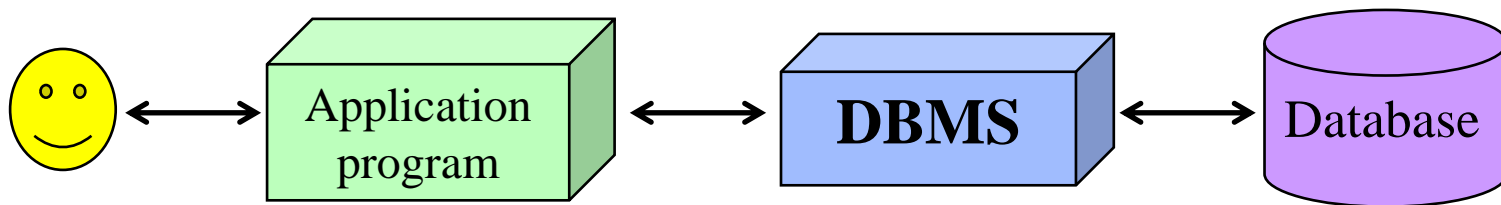
## Database Management System – DBMS

A computer program or a set of computer programs designed to enable a user to define, create, and maintain the database and also to provide controlled access to this database.

DBMS interacts with users or application programs. A general inquiry facility to user's data, built in the DBMS is a query language (most commonly used is SQL).



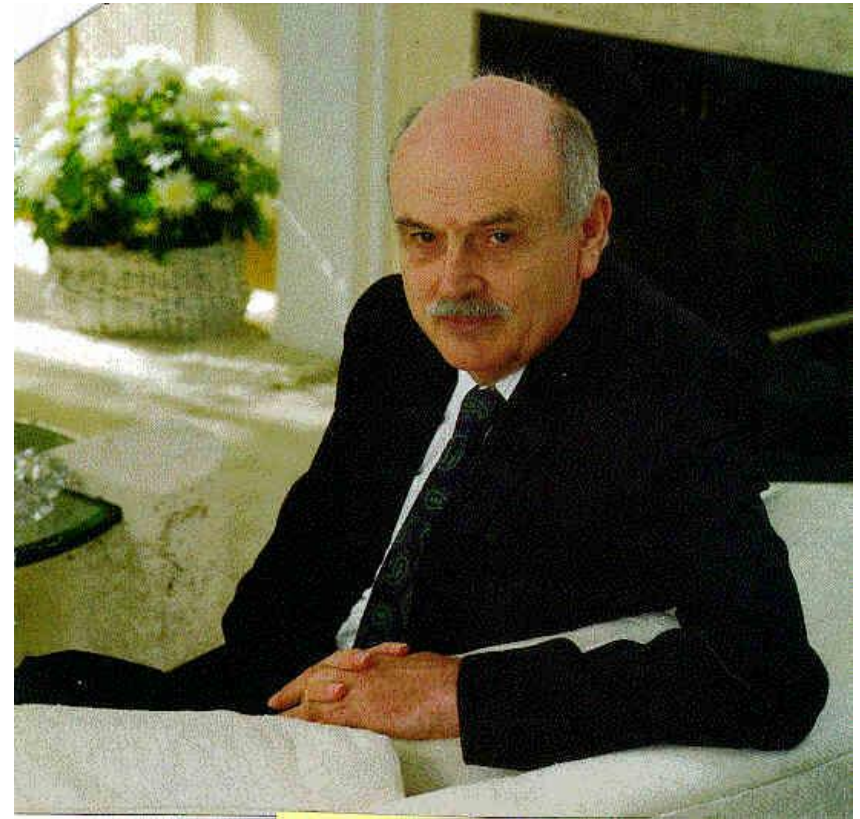
Using a database management system directly



Using a database management system from an application program

# The father of relational database model

A relational database is one that is built and operated in accordance with the relational model of data proposed by **Edgar Frank Codd** in 1970, an IBM engineer in the paper „**A Relational Model of Data for Large Shared Data Banks**” (*Communications of the ACM*, Volume 13, Numbr 5, June 1970, pp. 377-387). *Communications of the ACM* is a flagship magazine (published monthly) of the Association of Computing Machinery.



Codd (1923-2003), a trained English mathematician and computer scientist. While working for IBM, he invented the relational model for database management. Turing Award in 1981.

# A relational database

An excellent functionality and a commercial success of the relational model results from the fact that the model is based on strong mathematical foundations. The concept and terminology are taken from mathematics, principally from set theory and predicate calculus.

**Relation** is a basic term, from which the name of the database model is derived.

A **relational database** is a collection of relations logically connected. The connection is called a relationship.



Relational Database Model  
12 Rules  
by  
Edgar Frank Ted Codd

# A relation in a mathematical context

A **relation** in a mathematical theory is a subset of a Cartesian product of chosen domains. A **domain** is a set of certain values. Usually a domain reflects some aspect of reality, either mathematical (scientific) one or common one ("everyday's life").

## Example 1.

There are three sets:  $D1 = \{1, 2\}$ ,  $D2 = \{A, B\}$ ,  $D3 = \{1999-01-01, 2011-12-31\}$ .

Their cartesian product  $S$  is defined as follows:

$$S = \{(x, y, z) \mid x \in D1, y \in D2, z \in D3\} = \\ \{(1, A, 1999-01-01), (1, A, 2011-12-31), (1, B, 1999-01-01), (1, B, 2011-12-31), (2, A, 1999-01-01), (2, A, 2011-12-31), (2, B, 1999-01-01), (2, B, 2011-12-31)\}.$$

A relation is any subset of the  $S$  set. For example, there can be the following relations  $R1$  and  $R2$ , consisting of trios:

$$R1 = \{(x, y, z) \mid x \in D1, y \in D2, z \in D3, x = 1\} = \\ = \{(1, A, 1999-01-01), (1, A, 2011-12-31), (1, B, 1999-01-01), (1, B, 2011-12-31)\}$$

$$R2 = \{(x, y, z) \mid x \in D1, y \in D2, z \in D3, x = 2, y = A\} = \\ = \{(2, A, 1999-01-01), (2, A, 2011-12-31)\}$$



# From a mathematical relation to a database relation

Let there be the following three data sets:

- Surnames:  $SN = \{Nowak, Kowalski, Ferency, Podsiadło\}$
- Forenames:  $FN = \{Anna, Jan, Andrzej, Karol\}$
- Employment \_dates:  $D = \{2011-04-25, 2012-01-15, 2013-11-20, 2015-04-25\}$
- Salaries:  $S = \{7500.00, 5000.00, 4500.00, 3890.00\}$ .

The Cartesian product of the sets  $FN$ ,  $SN$ ,  $D$  and  $S$  consists of 256 four-element tuples (all possible quadruples containing one element from each set - each element of a given data set is combined with each element of remaining data sets).

A relation  $R$  in mathematical terms consists of quartets

$R = \{(Nowak, Anna, 2011-04-25, 7500,00), (Kowalski, Jan, 2012-01-15, 5000,00), (Ferency, Andrzej, 2013-11-20, 4500,00), (Podsiadło, Karol, 2015-04-25, 3890,00)\}$

The *Workers* relation in database terms

<u>Workers</u>			
<u>Surname</u>	<u>Forename</u>	<u>Employment_date</u>	<u>Salary</u>
Nowak	Anna	2011-04-25	7500,00
Kowalski	Jan	2012-01-15	5000,00
<u>Ferency</u>	Andrzej	2013-11-20	4500,00
Podsiadło	Karol	2015-04-25	3890,00

# A relation in a database context

A **relation** in a database theory is a table – a logical structure that consists of columns and rows.

There are the following properties that the table should satisfy:

1. The table has a name that is distinct from the names of all other tables in a database
2. The entries in the table are single-valued, i.e. there is only single value in a table cell
3. Each column has a distinct name within the table
4. All values in a column are values of the same meaning and type (values come from a single domain)
5. The order of columns is irrelevant
6. Each row is distinct
7. The order of rows is irrelevant.

## **Note**

Sometimes certain entries (represented by tables) contain repeating groups, which means that there are not single values in a table cell. Such structure is called unnormalised relation (unnormalized table).

# Database terminology I

**Entity** – a set of objects having the same properties; the entity can refer to a set of persons, a set of places, a set of things, a set of events etc. Each such set has an independent existence and is represented by a table (a relation) in a database. The term *entity* is used in database theory issues.

**Attribute** – a property of an entity; a field.

**Tuple** - a collection of values of related fields for one element in an entity; a record

attribute, column, field

entity, relation, table

<u>Workers</u>			
<u>Surname</u>	<u>Forename</u>	<u>Employment_date</u>	<u>Salary</u>
Nowak	Anna	2011-04-25	7500,00
Kowalski	Jan	2012-01-15	5000,00
Ferency	Andrzej	2013-11-20	4500,00
Podsiadło	Karol	2015-04-25	3890,00

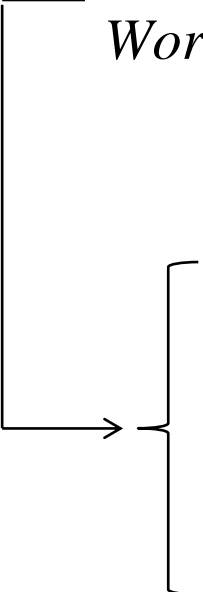
tuple, row, record

attribute values

# Database terminology II

**Relation definition** (structure) is the list of the relation attributes:

*Workers* = {*Surname*, *Forename*, *Employment\_date*, *Salary*}



Attribute	Attribute domain	Domain definition
Surname	People surnames	Text, up to 40 characters
Forename	People forenames	Text, up to 20 characters
<u>Employment_date</u>	Dates since 1950-01-01	Date
Salary	Employees' salaries	Currency

**Attribute domain** – the set of allowable values for the attribute.

**Domain definition** – the formal description of the attribute as regards its type and properties.

**Relation schema (table project)** is the **relation definition** along with the **relation domain definition**.

# Relational algebra

**Relational algebra** is a theoretical basis of manipulating a relational database.

In the relational algebra, operations act on existing tables to produce new tables. Retrieving data (in the form of a new table) from a relational database involves issuing relational algebra commands. Sometimes a series of commands are needed in order to obtain the final result.

There are the following fundamental relational algebra operations:

**UNION**

**DIFFERENCE**

**INTERSECTION**

**PRODUCT**

**PROJECTION**

**SELECTION**

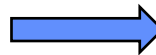
**JOIN**

# UNION

The **union** of two relations  $R$  and  $S$  with  $nR$  and  $nS$  tuples, respectively, is a relation formed by adding the tuples from one relation to those of a second relation to produce a third relation with a maximum of  $(nR+nS)$  tuples, duplicates being eliminated.

For this operation to make sense, the relations must be **union-compatible**, i.e. they must have the same number of attributes with matching domains.

Student_interest		
Sid	Name	Interest
B1	Nowak Jan	History
B2	Kowal Anna	Management
B3	Bratek Szymon	History
Student_engagement		
Sid	Name	Engagement
B1	Nowak Jan	History
B7	Cichosz Elwira	Mathematics



UNION: students with their scientific interest OR help engagement		
Sid	Name	Activity
B1	Nowak Jan	History
B2	Kowal Anna	Management
B3	Bratek Szymon	History
B7	Cichosz Elwira	Mathematics

# DIFFERENCE

The **difference** of two relations  $R$  and  $S$  is a relation containing tuples that appear in the first relation ( $R$ ) but not in the second relation ( $S$ ).

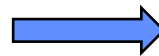
Both the relations must be **union-compatible**.

The **order of the difference matters**, therefore the difference of  $R$  and  $S$  is not the same as the difference of  $S$  and  $R$ .

Student_interest		
Sid	Name	Interest
B1	Nowak Jan	History
B2	Kowal Anna	Management
B3	Bratek Szymon	History

Student_engagement		
Sid	Name	Engagement
B1	Nowak Jan	History
B7	Cichosz Elwira	Mathematics



DIFFERENCE: students active in their scientific interest AND NOT in help engagement		
Sid	Name	Interest only
B2	Kowal Anna	Management
B3	Bratek Szymon	History

# INTERSECTION

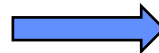
The **intersection** of two relations  $R$  and  $S$  is a relation containing tuples that appear both in the first relation ( $R$ ) and in the second relation ( $S$ ).

Both the relations must be **union-compatible**.

Student_interest		
Sid	Name	Interest
B1	Nowak Jan	History
B2	Kowal Anna	Management
B3	Bratek Szymon	History

Student_engagement		
Sid	Name	Engagement
B1	Nowak Jan	History
B7	Cichosz Elwira	Mathematics



INTERSECTION: students active in their scientific interest AND in help engagement		
Sid	Name	Activity
B1	Nowak Jan	History

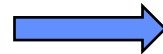


# PRODUCT

The **product** of two relations  $R$  and  $S$  (called also the **Cartesian product**) is a relation that is a concatenation of every tuple of the first relation ( $R$ ) with every tuple of the second relation ( $S$ ).

The product of a relation  $R$  having  $nR$  tuples and a relation  $S$  having  $nS$  tuples has  $nR$  times (multiplied by)  $nS$  tuples ( $nR * nS$ ).

Student_interest		
Sid	Name	Interest
B1	Nowak Jan	History
B2	Kowal Anna	Management
B3	Bratek Szymon	History



Hostel		
Shs	Name	Manager
H1	Laura	Herus Edmund
H2	Filon	Milik Karol
H3	Bartek	Konek Elwira
H4	Misiek	Bartos Julia

PRODUCT: all possible combination of tuples					
Sid	Name	Interest	Shs	Name	Manager
B1	Nowak Jan	History	H1	Laura	Herus Edmund
B2	Nowak Jan	History	H2	Filon	Milik Karol
B3	Nowak Jan	History	H3	Bartek	Konek Elwira
B4	Nowak Jan	History	H4	Misiek	Bartos Julia
B2	Kowal Anna	Management	H1	Laura	Herus Edmund
B3	Kowal Anna	Management	H2	Filon	Milik Karol
B4	Kowal Anna	Management	H3	Bartek	Konek Elwira
B5	Kowal Anna	Management	H4	Misiek	Bartos Julia
B3	Bratek Szym	History	H1	Laura	Herus Edmund
B4	Bratek Szym	History	H2	Filon	Milik Karol
B5	Bratek Szym	History	H3	Bartek	Konek Elwira
B6	Bratek Szym	History	H4	Misiek	Bartos Julia

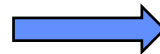
# PROJECTION

The **projection** operation works on a single relation  $R$  and defines a relation that contains a vertical subset of  $R$ , extracting the values of specified attributes and eliminating duplicates.

A **projection chooses columns** from a relation.

Because the result of projection is a relation and because relations cannot contain duplicates, redundant tuples are eliminated.

Teacher_subject		
Tid	Name	Subject
B1	Nawrot Klara	History
B2	Legutko Ewa	Management
B3	Bak Elwira	History
B1	Nawrot Klara	Sociology
B2	Legutko Ewa	History
B4	Bak Tadeusz	Microeconomy



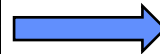
PROJECTION: projecting theTeacher_subject relation over Tid and Name attributes	
Tid	Name
B1	Nawrot Klara
B2	Legutko Ewa
B3	Bak Elwira
B4	Bak Tadeusz

# SELECTION

The **selection** operation works on a single relation  $R$  and defines a relation that contains a horizontal subset of  $R$ , i.e. those tuples that satisfy a specified condition.

A **selection** chooses rows from a relation.

Teacher_subject		
Tid	Name	Subject
B1	Nawrot Klara	History
B2	Legutko Ewa	Management
B3	Bąk Elwira	History
B1	Nawrot Klara	Sociology
B2	Legutko Ewa	History
B4	Bąk Tadeusz	Microeconomy



SELECTION: selecting Subject = 'History' from the Teacher_subject relation		
Tid	Name	Subject
B1	Nawrot Klara	History
B3	Bąk Elwira	History
B2	Legutko Ewa	History

# JOIN

The **join** of two relations  $R$  and  $S$  is a relation containing attributes from both the relations that are to be joined. Tuples in this new relation will be the concatenation of a tuple from the first relation ( $R$ ) and a tuple from the second relation ( $S$ ) that match on the common attributes (often called **join attributes**).

Two relations are joined on join attributes (in particular on one join attribute).

The operation is a **natural join** (**default meaning of join**) if the common attributes appear once in the resulting relations. If the common attributes appear twice the operation is the **equi-join**.

The (left) **outer join** of two relations  $R$  and  $S$  is a join in which tuples from  $R$  that do not have matching values in the common attributes of  $S$  are also included in the result relation. Missing values in the second relation ( $S$ ) are set to null.

# Join examples

Student		
Sid	Name	SHs
B1	Nowak Jan	H1
B2	Kowal Anna	H2
B3	Bratek Szymon	H5
B7	Cichosz Elwira	H1

Hostel		
Shs	Name	Manager
H1	Laura	Herus Edmund
H2	Filon	Milik Karol
H3	Bartek	Konek Elwira
H4	Misiek	Bartos Julia

EQUI-JOIN: students and their hostels					
Sid	Name	Shs	Shs	Name	Manager
B1	Nowak Jan	H1	H1	Laura	Herus Edmund
B2	Kowal Anna	H2	H2	Filon	Milik Karol
B7	Cichosz Elwira	H1	H1	Laura	Herus Edmund

(NATURAL) JOIN: students and their hostels				
Sid	Name	Shs	Name	Manager
B1	Nowak Jan	H1	Laura	Herus Edmund
B2	Kowal Anna	H2	Filon	Milik Karol
B7	Cichosz Elwira	H1	Laura	Herus Edmund

OUTER JOIN: students and matching hostels				
Sid	Name	Shs	Name	Manager
B1	Nowak Jan	H1	Laura	Herus Edmund
B2	Kowal Anna	H2	Filon	Milik Karol
B3	Bratek Szymon	H5		
B7	Cichosz Elwira	H1	Laura	Herus Edmund