University of Paris-Saclay — **Master 1 - AI**
2022/23: first semester — **Relational databases**
Teacher : Carmelo Vaccaro

Tutorial 1 with answers

Exercise 1 What is a database management system?

Answer. A database management system is a software that is used to manage a database.

Exercise 2 List some of the drawbacks of using a file system for managing a database instead of a DBMS.

Answer. Some data could be in a file that is not easy to find.
The hard disk where the files are located could break and data can be lost.
Two users could modify the same files at the same time and some modifications could be lost.

Exercise 3 Give examples of things that could wrong when data are managed with a system (like a file system) that does not prevent isolation of actions by different users (two or more users access at around the same time the same file).

You can take as example a banking system and an airline reservation system or other examples of your choice.

Answer. Bank account data: suppose that at around the same time two operations take place, for example one where money is transferred and one where money is received. The modifications of one of these operations could be lost, so money could disappear from the bank account.

Airline reservation system: suppose that at around the same time the same airplane place is booked at two different travel agencies. Then the same place is booked twice for two different people.

Exercise 4 What is the main difference between early DBMS (hierarchical and network based) with relational ones?

Answer. Unlike early DBMS, relational databases did not show to the user the real data structure, but only an interface.

Exercise 5 Compare pros and cons of the first DBMS (hierarchical and network based) with relational ones.

Deduce the reason why by 1990's relational DBMS became the norm.

Today hierarchical based DBMS are still in use (like IMS of IBM): what is the reason in your opinion?

Answer. Hierarchical and network based DBMS were very simple and very efficient but they were very complicated to use, even for very simple queries.
Relational DBMS were much easier to use but were less efficient than hierarchical and network based ones.
The reason why relational DBMS became the norm by 1990's was that at that time all the computers were powerful enough to run efficiently relational DBMS.
The reason why today hierarchical based DBMS are still in use is due to the fact that they are part of legacy systems and the cost and difficulty of migrating to a more modern system can be too high.

acctNo	*type*	*balance*
12345	savings	12000
23456	checking	1000
34567	savings	25

Figure 1: The relation *Accounts*

firstName	*lastName*	*idNo*	*account*
Robbie	Banks	901-222	12345
Lena	Hand	805-333	12345
Lena	Hand	805-333	23456

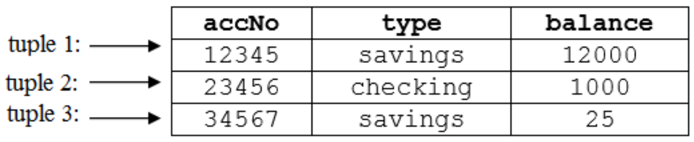
Figure 2: The relation *Customers*

Exercise 6 In Fig. 1 and 2 are instances of two relations that might constitute part of a banking database. Indicate the following:

a) The attributes of each relation.
b) The tuples of each relation.
c) The components of one tuple of your choice from each relation.
d) The relation schema for each relation.
e) The database schema.
f) A suitable domain for each attribute.

Answer.
a) The attributes of *Accounts* relation are: *acctNo*, *type*, *balance*.
The attributes of *Customers* relation are: *firstName*, *lastName*, *idNo*, *account*.

b) The data related to one account holder represents a tuple of *Accounts* relation.
The tuples of *Accounts* relation are:



	accNo	**type**	**balance**
tuple 1: →	12345	savings	12000
tuple 2: →	23456	checking	1000
tuple 3: →	34567	savings	25

The data related to one customer represents a tuple of *Customers* relation.
The tuples of *Customers* relation are

	firstName	lastName	idNo	account
tuple 1: →	Robbie	Banks	901-222	12345
tuple 2: →	Lena	Hand	805-333	12345
tuple 3: →	Lena	Hand	805-333	23456

c) The components of tuple 1 from *Accounts* relation are: 12345 for *acctNo*; 'savings' for *type*, 12000 for *balance*.

The components of tuple 2 from *Customers* relation are: 'Lena' for *firstName*, 'Hand' for *lastName*, '805-333' for *idNo*, 12345 for *account*.

d) The relation schema for *Accounts* relation is:

$$Accounts\ (acctNo,\ balance,\ type)$$

The relation schema for Customers relation is:

$$Customers\ (idNo,\ firstName,\ lastName,\ account)$$

e) The database schema is as follows:

$$Accounts\ (acctNo,\ balance,\ type)$$

$$Customers\ (idNo,\ firstName,\ lastName,\ account)$$

f) A suitable domain for each attribute of *Accounts* relation is as follows:
- *acctNo*: Integer
- *type*: String
- *balance*: Integer

A suitable domain for each attribute of *Customers* relation is as follows:
- *firstName*: String
- *lastName*: String
- *idNo*: String
- *account*: Integer

Exercise 7 Let us consider the following two relations schemas

Movies(title: String, year: Integer, length: Integer, genre: String)

and

Movies(genre: String, year: Integer, title: String, length: Integer)

Are they related in some manner? Justify your answer.

Answer. The two schemas are indeed the same because the relations have the same name and their attributes have the same name and type. The only difference is the order of attributes, but since the attributes are a set and not a list the order is not important.

Exercise 8 Let us consider the following table

Actor	*Gender*	*MoviesPlayed*
Scarlett Johansson	F	Girl with a Pearl Earring - Match Point
Will Smith	M	Men in Black - Men in Black 2 - I, Robot
Leonardo DiCaprio	M	Titanic - Inception

What is wrong with it?

Answer. The attribute *MoviesPlayed* does not respect atomicity because it is a list.

Exercise 9 Let us consider the following table

Actor	*Film*	*Year*	*SalaryInDollars*
Bruce Willis	The Sixth Sense	1999	30,000,000
Sandra Bullock	Gravity	2013	20,000,000
Robert Downey Jr.	Avengers: Endgame	2019	20 millions

What is wrong with it?

Answer. The attribute *SalaryInDollars* has some values that are integer and some other that are string, while the type of the values of an attribute must be the same for all the tuples.

Exercise 10 Which data type would you choose for an attribute in the following cases:

1. the attribute can take only two possible values and you want to save memory to store this attribute;
2. the attribute is a single character or a string of always the same length (like airport codes);
3. the attribute is a floating point number that you want to record with a high degree of precision;
4. the attribute is a string and you want the width of the column of the attribute to be the same for all tuples by adding extra spaces at the end;
5. the attribute is a small value integer and you want to save memory to store this attribute;
6. the attribute is a floating point number that you want to be approximated always with the same number of digits after the dot;
7. the attribute is a string of varying length and you do not want to waste memory.

Answer.

1. *BOOLEAN* or *BIT(1)*
2. *CHAR(n)*
3. *DOUBLE PRECISION*
4. *CHAR(n)*
5. *SHORTINT*
6. *DECIMAL*
7. *VARCHAR(n)*

Exercise 11 Write in SQL the following:

1. the date of today;
2. the date of beginning of next summer;
3. the time this course has started today;
4. the time of a quarter to midnight plus 27 seconds and 65 hundredths of second.

Answer.

1. DATE '2022-09-16';
2. DATE '2023-06-21';
3. TIME '09:00:00';
4. TIME '23:45:27.65'.

Exercise 12 Let us consider the following database schema consisting of four relations, whose schemas are:

Product(maker, model, type)
PC(model, speed, ram, hd, price)
Laptop(model, speed, ram, hd, screen, price)
Printer(model, color, type, price)

The *Product* relation gives the manufacturer, model number and type (PC, laptop, or printer) of various products. We assume for convenience that model numbers are unique over all manufacturers and product types; that assumption is not realistic, and a real database would include a code for the manufacturer as part of the model number.

The *PC* relation gives for each model number that is a PC the speed (of the processor, in gigahertz), the amount of RAM (in megabytes), the size of the hard disk (in gigabytes), and the price.

The *Laptop* relation is similar, except that the screen size (in inches) is also included.

The *Printer* relation records for each printer model whether the printer produces color output (true, if so), the process type (laser or ink-jet, typically), and the price.

Do the following:

- for each table find the attributes and their domain, find a primary key if any and write a statement to create the table;
- write the statement to alter the *Printer* table to delete the attribute *color*;
- write the statement to alter the *Laptop* table to add the attribute *od* (optical-disk type, e.g., cd or dvd). Let the default value for this attribute be 'none' if the laptop does not have an optical disk.

Answer.

a) *CREATE TABLE Product (maker CHAR (25), model CHAR (15) PRIMARY KEY, type CHAR (25)).*

b) *CREATE TABLE PC (model CHAR (15) PRIMARY KEY, speed DECIMAL(4,2), ram INTEGER, hd INTEGER, price DECIMAL (7,2)).*

c) *CREATE TABLE Laptop (model CHAR (15) PRIMARY KEY, speed DECIMAL (4,2), ram INTEGER, hd INTEGER, screen DECIMAL (3,1), price DECIMAL (7,2)).*

d) *CREATE TABLE Printer (model CHAR (15) PRIMARY KEY, color BOOLEAN, type CHAR (10), price DECIMAL (7,2)).*

e) *ALTER TABLE Printer DROP color.*

f) *ALTER TABLE Laptop ADD od CHAR (10) DEFAULT 'none'.*

Exercise 13 Let us consider the following relations:

Classes(class, type, country, numGuns, bore, displacement)
Ships(name, class, launched)
Battles(name, date)
Outcomes(ship, battle, result)

Ships are built in "classes" from the same design, and the class is usually named for the first ship of that class.

The relation *Classes* records the name of the class, the type ('bb' for battleship or 'bc' for battlecruiser), the country that built the ship, the number of main guns, the bore (diameter of the gun barrel, in inches) of the main guns, and the displacement (weight, in tons).

Relation *Ships* records the name of the ship, the name of its class, and the year in which the ship was launched.

Relation *Battles* gives the name and date of battles involving these ships, and relation *Outcomes* gives the result (sunk, damaged, or ok) for each ship in each battle.

Do the following:

- for each table find the attributes and their domain, find a primary key if any and write a statement to create the table;
- write the statement to alter the *Classes* table to delete the attribute *bore*;
- write the statement to alter the *Ships* table to add the attribute *yard* giving the shipyard where the ship was built.

Answer.

a) *CREATE TABLE Classes (class CHAR(25) PRIMARY KEY, type CHAR(10), country CHAR(25), numGuns INT, bore DECIMAL(3,1), displacement INT).*

b) *CREATE TABLE Ships (name VARCHAR(50), class CHAR(25), launched INT).*

c) *CREATE Battles (name VARCHAR(50) PRIMARY KEY, date DATE).*

d) *CREATE TABLE Outcomes (ship VARCHAR(50), battle VARCHAR(50), result CHAR(15)).*

e) *ALTER TABLE Classes DROP bore.*

f) *ALTER TABLE Ships ADD yard VARCHAR(50).*

Exercise 14 Consider the table below which is an instance of a table called *Students*.

ID	*name*	*login*	*age*	*gradeAvg*
50000	Dave	dave@cs	19	3.3
53650	Smith	smith@math	19	3.8
53666	Jones	jones@cs	18	3.4
53670	Ron	NULL	21	3.5
53688	Smith	smith@ee	18	3.2
53831	Madayan	madayan@music	11	1.8
53832	Guldu	guldu@music	12	2.0

1. Find all sets of attributes that are not unique based on this instance.
2. Find all sets of attributes that can be unique but not primary key based on this instance.
3. Find three sets of attributes that can be a primary key based on this instance. Which ones would you not choose as primary key for the whole relation? And why?

Answer.

1. All sets of attributes that contain *name* and/or *age*.
2. All sets of attributes that contain *login*.
3. $\{ID\}$, $\{name, age\}$, $\{gradeAvg\}$. I would not choose $\{name, age\}$ and $\{gradeAvg\}$ because there can be different students with the same name and age or with the same grade average.