Designing a Relational Database
Problems and Constraints

SAPIENZA
UNIVERSITÀ DI ROMA

The content of the slides is authored by Prof. Maria de Marsico, Sapienza University of Rome
Part of the material is taken from the book: Atzeni, Ceri, Paraboschi, Torlone Basi di dati - Modelli e linguaggi di interrogazione , McGraw-Hill, 2002.

Objective

- suppose we want to create a database containing the following undergraduate student data:

- personal and identification data
  - first and last name,
  - date, town and province of birth, matriculation number,
  - tax code
- curricular data
  - for each exam taken:
  - vote,
  - date,
  - course code, title and teacher

Hypothesis 1

the database consists of a single schema:

Curriculum (Matr, TC, SurN, Name, BirthD, City, Prov, C#, Tit, Doc, Date, Grade)

Hypothesis 1 (problems)

Curriculum

Matr	TC	SurN	Name	BirthD	City	Prov	C#	TitC	DocC	Date	Grade
01	**…**	**Rossi**	**Mario**	**…**	**Tolfa**	**Rome**	**10**	**Physics**	**Goofy**	…	…
02	…	Bianchi	Paolo	…	Tolfa	Rome	**10**	**Physics**	**Goofy**	…	…
01	**…**	**Rossi**	**Mario**	**…**	**Tolfa**	**Rome**	20	Chemistry	Pluto	…	…

Hypothesis 1 (problems)

Curriculum

Matr	TC	SurN	Name	BirthD	City	Prov	C#	TitC	DocC	Date	Grade
01	**…**	**Rossi**	**Mario**	**…**	**Tolfa**	**Rome**	**10**	**Physics**	**Goofy**	…	…
02	…	Bianchi	Paolo	…	Tolfa	Rome	**10**	**Physics**	**Goofy**	…	…
01	**…**	**Rossi**	**Mario**	**…**	**Tolfa**	**Rome**	20	Chemistry	Pluto	…	…

redundancy:

- student's biographical data is stored for each exam taken by the student
- course data is stored for each exam taken for that course

Hypothesis 1 (problems)

- redundancy gives rise to:
- waste of memory space
- update, insert and delete anomalies

Hypothesis 1 (problems)

Curriculum	Matr	TC	SurN	Name	BirthD	City	Prov	C#	TitC	DocC	Date	Grade
	01	…	Rossi	Mario	…	Tolfa	Rome	**10**	**Physics**	**Goofy**	…	…
	02	…	Bianchi	Paolo	…	Tolfa	Rome	**10**	**Physics**	**Goofy**	…	…
	01	…	Rossi	Mario	…	Tolfa	Rome	20	Chemistry	Pluto	…	…

update anomaly

- if the teacher of a course changes, that information has to be modified for **each exam** taken for that course

Curriculum	Matr	TC	SurN	Name	BirthD	City	Prov	C#	TitC	DocC	Date	Grade
	01	…	Rossi	Mario	…	Tolfa	Rome	**10**	**Physics**	**Minni**	…	…
	02	…	Bianchi	Paolo	…	Tolfa	Rome	**10**	**Physics**	**Minni**	…	…
	01	…	Rossi	Mario	…	Tolfa	Rome	20	Chemistr	Pluto	…	…

Hypothesis 1 (problems)

Curriculum

Matr	TC	SurN	Name	BirthD	City	Prov	C#	TitC	DocC	Date	Grade
01	…	Rossi	Mario	…	Tolfa	Rome	10	Physics	Goofy	…	…
02	…	Bianchi	Paolo	…	Tolfa	Rome	10	Physics	Goofy	…	…
01	…	Rossi	Mario	…	Tolfa	Rome	20	Chemistry	Pluto	…	…

insertion anomaly

- we cannot enter a student's personal data **until they have taken at least one exam**
- the same happens for courses

Curriculum

Matr	TC	SurN	Name	BirthD	City	Prov	C#	TitC	DocC	Date	Grade
01	…	Rossi	Mario	…	Tolfa	Rome	10	Physics	Minni	…	…
02	…	Bianchi	Paolo	…	Tolfa	Rome	10	Physics	Minni	…	…
01	…	Rossi	Mario	…	Tolfa	Rome	20	Chemistry	Pluto	…	…
03	**…**	**Neri**	**Giulio**	**…**	**Nepi**	**Rome**	**-**	**-**	**-**	**-**	**-**

Hypothesis 1 (problems)

Curriculum	Matr	TC	SurN	Name	Birth D	City	Prov	C#	TitC	DocC	Date	Grad e
	01	**…**	**Rossi**	**Mario**	**…**	**Tolfa**	**Rome**	10	Physics	Goofy	…	…
	02	…	Bianchi	Paolo	…	Tolfa	Rome	10	Physics	Goofy	…	…
	01	**…**	**Rossi**	**Mario**	**…**	**Tolfa**	**Rome**	**20**	**Chemistr y**	**Pluto**	…	…

deletion anomaly:

- by deleting a student's master data **course data may be deleted** (if the student is the only one to have taken the examination of that course)
- the same happens when deleting a course

Curriculum	Matr	TC	SurN	Name	Birth D	City	Prov	C#	TitC	DocC	Date	Grad e
	02	…	Bianchi	Paolo	…	Tolfa	Rome	10	Physics	Goofy	…	…

Hypothesis 2

the database consists of **three schemas**:

Student (Matr, TC, SurN, Name, Date, City, Province)
Course (C#, Tit, Doc)
Exam (Matr, C#, Date, Grade)

Hypothesis 2

data of hypothesis 1:

Curriculum

Matr	TC	SurN	Name	Birth D	City	Prov	C#	Tit	Doc	Date	Grade
01	**…**	**Rossi**	**Mario**	**…**	**Tolfa**	**Rome**	**10**	**Physics**	**Goofy**	…	…
02	…	Bianchi	Paolo	…	Tolfa	Rome	**10**	**Physics**	**Goofy**	…	…
01	**…**	**Rossi**	**Mario**	**…**	**Tolfa**	**Rome**	20	Chemistry	Pluto	…	…

can be stored in three relations:

Student

Matr	TC	SurN	Name	Date	City	Prov
01	**…**	**Rossi**	**Mario**	**…**	**Tolfa**	**Rome**
02	…	Bianchi	Paolo	…	Tolfa	Rome

Course

C#	Tit	Doc
10	**Physics**	**Goofy**
20	Chemistry	Pluto

Examination

Matr	C#	Date	Grade
01	10	…	…
02	10	…	…
01	20		

Hypothesis 2

we no longer have the previously described redundancy issues and update, insertion and deletion anomalies, however...

Hypothesis 2

Student	Matr	TC	SurN	Name	Date	City	Prov
	01	…	Rossi	Mario	…	**Tolfa**	**Rome**
	02	…	Bianchi	Paolo	…	**Tolfa**	**Rome**

Redundancy issue

- The fact that a municipality is located in a certain province is repeated for every student born in that town

Update anomaly

- If a municipality changes province (as a result of the creation of a new Province) multiple tuples must be modified

Insertion anomaly

- It is not possible to memorize the fact that a certain municipality is located in a certain province if there is not at least one student born in that municipality

Deletion anomaly

- If the personal data of a student are deleted, the information that a certain municipality is located in a certain province may be lost (if he/she is the only student born in that municipality).

Hypothesis 3

the database consists of four schemas:

Student (Matr, TC, SurN, Name, Date, City)
Course (C#, Tit, Doc)
Exam (Matr, C#, Date, Grade)
Municipality (City, Prov)

Hypothesis 3

Student

Matr	TC	SurN	Name	Date	City
01	…	Rossi	Mario	…	Tolfa
02	…	Bianchi	Paolo	…	Tolfa

Municipality

City	Prov
Tolfa	Rome

Examination

Matr	C#	Date	Grade
01	10	…	…
02	10	…	…
01	20	…	…

Course

C#	Tit	Doc
10	Physics	Goofy
20	Chemistry	Pluto

no more redundancies or anomalies!

A "good" database...

a database schema is "good" if it has no

- redundancies
- update, insertion and deletion anomalies

so...

… the schema

- Student (Matr, TC, SurN, Name, Date, City)
- Course (C#, Tit, Doc)
- Exam (Matr, C#, Date, Grade)
- Municipality (City, Prov)

...is a "good" schema

Problem

- how to design a "good" schema?

Observation

the problems we found in the schema:

Curriculum (Matr, TC, SurN, Name, BirthD, City, Prov, C#, Tit, Doc, Date, Grade)

derive from the fact that three **distinct concepts** (student, course, exam) are represented in **a single relation** and…

Observation

…are overcome when the three concepts are represented separately in three distinct relations:

- Student (Matr, TC, SurN, Name, Date, City, Province)
- Course (C#, Tit, Doc)
- Exam (Matr, C#, Date, Grade)

Observation

similarly, the problems examined in the schema:

Student (Matr, TC, SurN, Name, Date, City, Province)

derive from the fact that **two distinct concepts** (student, municipality) are represented in **a single** relation and...

Observation

…are overcome when the two concepts are represented in the two distinct relations

- Student (Matr, TC, SurN, Name, Date, City)
- Municipality (City, Prov)

obviously, we could (we should!) jump immediately to the correct solution… in the initial schema the different concepts are four, but one is less obvious

Observation

- often, a bad design, that is, the error of representing several concepts in the same relation, stems from the need to retrieve information related **not only to objects but also to the associations in which they are involved**

- e.g., in the case we have just seen, frequently performed operations may require to retrieve data relating to **all the exams** taken by a student or the teachers of the courses they have attended, so it may seem **convenient** to store all this information together

- we have seen that the relational algebra operators (and those of relational database languages) allow, by creating **references by value** between associated objects and through appropriate **joins** between relations, to **obtain the same information** <u>**without running into the problems we have discussed**</u>

Solution

in order to design a "good" schema, it is necessary to represent **each concept separately** in a distinct relation

here, we have all the information available at once… but we have all the anomalies we talked about!

Orders

Name	C#	City	A#	N-pieces	Descr	Price
Rossi	C1	Rome	A1	100	Dish	3
Rossi	C2	Milan	A2	200	Glass	2
Bianchi	C3	Rome	A2	150	Glass	2
Greens	C4	Rome	A3	200	Mug	4
Rossi	C1	Rome	A2	200	Glass	2
Rossi	C1	Rome	A3	100	Mug	4

Orders	Name	C#	City	A#	N-pieces	Descr	Price
	Rossi	C1	Rome	A1	100	Dish	3
	Rossi	C2	Milan	A2	200	Glass	2
	Bianchi	C3	Rome	A2	150	Glass	2
	Greens	C4	Rome	A3	200	Mug	4
	Rossi	C1	Rome	A2	200	Glass	2
	Rossi	C1	Rome	A3	100	Mug	4

- if we have to update a customer's city, we have to update ALL the tuples of that customer, and in case of inconsistencies we are no longer able to know which one is right!
- if we delete an item that goes out of production, we risk losing the data of a customer who has ordered only that item
- to insert an article there must be a customer who orders it

A worse solution

Orders

Name	C#	City	A#	N-pieces
Rossi	C1	Rome	A1	100
Rossi	C2	Milan	A2	200
Bianchi	C3	Rome	A2	150
Greens	C4	Rome	A3	200
Rossi	C1	Rome	A2	200
Rossi	C1	Rome	A3	100

Article

A#	Descr	Price
A1	Dish	3
A2	Glass	2
A3	Mug	4

- **we only increase the probability of error!**

- so, here is a "good" database in which we can still derive the information through join operations:

Customer

Name	C#	City
Rossi	C1	Rome
Rossi	C2	Milan
Bianchi	C3	Rome
Greens	C4	Rome

Order

C#	A#	N-pieces
C1	A1	100
C2	A2	200
C3	A2	150
C4	A3	200
C1	A2	200
C1	A3	100

Article

A#	Descr	Price
A1	Dish	3
A2	Glass	2
A3	Mug	4

Problem

how can the concepts represented in a (well-designed) schema be identified?

Solution

we can identify the concepts represented in a relation by means of a (the?)

key

which is an attribute or a group of attributes

that determine a particular

functional dependence

which is a particular kind of **constraint**

Constraints

Conditions in the reality of interest

in the context that we are representing in a database, certain **conditions** are met, for example:

1. grade is an integer between 18 and 30
2. matriculation number uniquely identifies a student
3. matriculation number in an examination report must be the matriculation number of a student
4. an employee's salary cannot go down
5. extra salary is given by the number of overtime hours worked the hourly wage

Constraints on the data base

- when representing a reality of interest in a database there must be a way to represent these conditions
- a **constraint** is the representation in the database schema of a condition that is valid in the reality of interest
- an instance of the database is **legal** if it satisfies all constraints (i.e., if it is an "honest" representation of reality)

Definition and verification of constraints in a DBMS

- a DBMS allows us:
- to **define** the constraints together with the database schema
- to **verify** that an instance of the database is legal
- based on special predefined constraints, to **prevent the** insertion of tuples that would violate those constraints

Definition and verification of constraints in a DBMS

- a DBMS has procedures for checking the most frequently occurring constraints:
  - **domain constraints** (a grade is an integer between 18 and 30)
  - **keys** (the matriculation number uniquely identifies a student)
  - **containment of domains** (matriculation number in an exam report must be a student's matriculation number)
- for other types of constraints (e.g., dynamic constraints, constraints involving values of multiple attributes in a mathematical expression) it may be necessary to define appropriate procedures

Functional dependencies

we will see that the **functional dependencies** defined on a relation schema express particular dependency constraints **between subsets of attributes of the schema** itself, which must be satisfied **by each legal instance of** the schema