Databases

The Relational Data model

Logical Models

- There are three traditional logical models
  - Hierarchical
  - Network
  - Relational
- More recent
  - Object Oriented (quite uncommon)
  - over XML ("complementary" to the relational one)

Data models, features

- Hierarchical and Network
  - use explicit references (pointers) between records
- Relational is “value based"
  - references between data stored within relations are represented through values

Students

Number	Surname	Name	Date of Birth
6554	Rossi	Mario	1978/12/05
8765	Neri	Paolo	1976/11/03
9283	Verdi	Luisa	1979/11/12
3456	Rossi	Maria	1978/02/01

Exams

Student	Grade	Lecture
3456	30	04
3456	24	02
9283	28	01
6554	26	01

Lectures

Code	Name	Lecturer
01	Maths	Mario
02	Chemistry	Bruni
04	Chemistry	Verdi

The relational data model

- Defined by E. F. Codd in 1970 in order to allow data independence
- Implemented in real DBMS in 1981 (it's not easy to implement data independence both efficiently and in a reliable way!)
- It is based on the logical definition of "relation" (with some differences)
- Relations are represented through tables

Relations and Relationships

- logic relation as in Set Theory
- relation as in the relational data model
- relationship expresses a specific class of facts in the Entity-Relationship model

Logical Relation

$D_1 = \{a, b\}$

$D_2 = \{x, y, z\}$

cartesian product $D_1 \times D_2$

a	x
a	y
a	z
b	x
b	y
b	z

a relation $r \subseteq D_1 \times D_2$

a	x
a	z
b	y

Logical Relation

- $D_1, \ldots, D_n$ (n not necessairly different sets)
- cartesian product $D_1 \times \ldots \times D_n$:
  - the set of all the tuples $(d_1, \ldots, d_n)$ such that $d_1 \in D_1, \ldots, d_n \in D_n$
- logical relation over $D_1, \ldots, D_n$:
  - a subset of $D_1 \times \ldots \times D_n$.
- $D_1, \ldots, D_n$ are the relation's domains

Logical Relation, properties

- a logical relation is a set of ordered tuples:
  - $(d_1, \ldots, d_n)$ such that $d_1 \in D_1, \ldots, d_n \in D_n$
- a relation is a set:
  - there is no order between the tuples
  - the tuples are all distinct
  - each n-uple is ordered: the i-th values “comes from” the i-th domain

Logical relation, an example

Matches $\subseteq$ ***string*** $\times$ ***string*** $\times$ ***int*** $\times$ ***int***

Barca	Bayern	3	1
Bayern	Real	2	0
Barca	Juve	0	2
Juve	Real	0	1

- Each domain appears with two distinct roles, that could be distinguished by its position:
  - This structure is positional

Non positional data structure

- **Each unique name in the table (attribute) is associated to a domain. The attribute provides the "role" of the domain.**

Home	Away	GoalsH	GoalsA
Barca	Bayern	3	1
Bayern	Real	2	0
Barca	Juve	0	2
Juve	Real	0	1

- The specific position of each attribute in the table schema is irrelevant: the data structure is **non positional**

Non positional data structure (2)

Home	Away	GoalsH	GoalsA
Barca	Bayern	3	1
Bayern	Real	2	0
Barca	Juve	0	2
Juve	Real	0	1

Away	Home	GoalsA	GoalsH
Bayern	Barca	1	3
Real	Real	0	2
Roma	Juve	2	0
Milan	Real	1	0

Tables and Relations

- A table representing a relation:
  - each row could assume any position
  - each column could assume any position
- A table represents a relation if
  - all rows are different
  - all columns headers are different
  - values within the columns' are homogeneous

The data model is value based

- References between data stored in different relations are represented through values in the tuples.

Example

Students

Number	Surname	Name	Date of Birth
6554	Rossi	Mario	1978/12/05
8765	Neri	Paolo	1976/11/03
9283	Verdi	Luisa	1979/11/12
3456	Rossi	Maria	1978/02/01

Exams

Student	Grade	Lecture
3456	30	04
3456	24	02
9283	28	01
6554	26	01

Lectures

Code	Name	Lecturer
01	Maths	Mario
02	Chemistry	Bruni
04	Chemistry	Verdi

Another option

- Some data models (such as network, hierarchical, object oriented) use explicit references, handled by the system.

Example: value-based

Students

Number	Surname	Name	Date of Birth
6554	Rossi	Mario	1978/12/05
8765	Neri	Paolo	1976/11/03
9283	Verdi	Luisa	1979/11/12
3456	Rossi	Maria	1978/02/01

Exams

Student	Grade	Lecture
3456	30	04
3456	24	02
9283	28	01
6554	26	01

Lectures

Code	Name	Lecturer
01	Maths	Mario
02	Chemistry	Bruni
04	Chemistry	Verdi

Example: pointers

Students

Number	Surname	Name	Date of Birth
6554	Rossi	Mario	1978/12/05
8765	Neri	Paolo	1976/11/03
9283	Verdi	Luisa	1979/11/12
3456	Rossi	Maria	1978/02/01

Exams

Student	Grade	Lecture
	30	
	24	
	28	
	26	

Lectures

Code	Name	Lecturer
01	Maths	Mario
02	Chemistry	Bruni
04	Chemistry	Verdi

Value-Based data structure: pros

- independent from the physical data structure that could dynamically change (the same thing could be provided with “high-level” pointers).
- the only data that is stored is the one that is relevant form the software application point of view
- the user and the programmer see the same data
- data can be easily shared between different environments
- pointers are directional

Definitions

- Schema of a relation:

  a relation name R with a set of attributes A_1, ..., A_n:

  $$R(A_1, ..., A_n)$$

- Schema of a database:

  set of schemas (relation):

  $$R = \{R_1(X_1), ..., R_k(X_k)\}$$

Example

Attributes

- Schema of a relation:
  STUDENTS (Number, Surname, Name, Year of Birth)

Relation

- Relational database schema:
  R= {**STUDENTS** (Number, Surname, Name, Year of Birth), **EXAMS** (Student, Grade, Lecture), **LECTURES**(Code, Name, Lecturer)}

Definitions (2)

- A tuple over a set of attributes X is a **mapping** from each attribute A in X to a value in domain of A
- t[A] expresses the value of a tuple t over the attribute A

E.g., if *t* is the first tuple within **STUDENTS**, then

t [Name] = Mario

Definitions (3)

- a relation instance r is a finite set of tuples having a schema R(X)
- a relational database instance on a schema R= $\{R_1(X_1), ..., R_n(X_n)\}$ is a set of relation instances r = $\{r_1,..., r_n\}$ (with r_i a relation over R_i)

Example of a relation instance

Students

Number	Surname	Name	Year of Birth
6554	Rossi	Mario	1978/12/05
8765	Neri	Paolo	1976/11/03
9283	Verdi	Luisa	1979/11/12
3456	Rossi	Maria	1978/02/01

Students

Number	Surname	Name	Year Of Birth
6554	Rossi	Mario	1978/12/05
8765	Neri	Paolo	1976/11/03
9283	Verdi	Luisa	1979/11/12
3456	Rossi	Maria	1978/02/01

Exams

Student	Voto	Lecture
3456	30	04
3456	24	02
9283	28	01
6554	26	01

Lectures

Code	Name	Lecturer
01	Maths	Mario
02	Chemistry	Bruni
04	Chemistry	Verdi

Relations with just one attribute

Students

Number	Surname	Name	Year of Birth
6554	Rossi	Mario	05/12/1978
8765	Neri	Paolo	03/11/1976
9283	Verdi	Luisa	12/11/1979
3456	Rossi	Maria	01/02/1978

Worker Students

Number
6554
3456

Nested Data Structures

Restaurant Da Filippo *Via Roma 2, Roma*		
Receipt 1235 *at* 2002/10/12		
3	Cover Charge	3,00
2	Appetizer	6,20
3	Soup	12,00
2	Meat Loaf	18,00
	Total Sum	39,20

Restaurant Da Filippo *Via Roma 2, Roma*		
Receipt 1240 *at* 2002/10/13		
2	Cover Charge	2,00
2	Appetizer	7,00
2	Soup	8,00
2	Fish	20,00
2	Coffee	2,00
	Total Sum	39,00

Nested Data Structures

Restaurant Da Filippo *Via Roma 2, Roma*		
Receipt 1235 *at* 2002/10/12		
3	Cover Charge	3,00
2	Appetizer	6,20
3	Soup	12,00
2	Meat Loaf	18,00
	Total Sum	39,20

Restaurant Da Filippo *Via Roma 2, Roma*		
Receipt 1240 *at* 2002/10/13		
2	Cover Charge	2,00
2	Appetizer	7,00
2	Soup	8,00
2	Fish	20,00
2	Coffee	2,00
	Total Sum	39,00

Nested Data Structures

Da Filippo *Via Roma 2, Roma*		
Receipt 1235 *at* 2002/10/12		
3	Cover Charge	3,00
2	Appetizer	6,20
3	Soup	12,00
2	Meat Loaf	18,00
	Total Sum	39,20

Da Filippo *Via Roma 2, Roma*		
Receipt 1240 *at* 2002/10/13		
2	Cover Charge	2,00
2	Appetizer	7,00
2	Soup	8,00
2	Eel	20,00
2	Coffee	2,00
	Total Sum	39,00

Nested Data Structures

Receipts

No	Date	Per	Description	Price	Total
1235	2002/10/12	3	Cover Charge	3,00	39,20
		2	Appetizer	6,20	
		3	Soup	12,00	
		2	Meat Loaf	18,00	
1240	2002/10/13	2	Cover Charge	2,00	39,00
		...	...	...	

Relations representing nested data structures

Receipt

Number	Data	Total
1235	12/10/2002	39,20
1240	13/10/2002	39,00

Courses

Receipt	Per	Description	Price
1235	3	Coperti	3,00
1235	2	Antipasti	6,20
1235	3	Primi	12,00
1235	2	Bistecche	18,00
1240	2	Coperti	2,00
…	…	…	…

Nested relations: some thoughts

- Did we really map all the possible aspects of a receipt?
- It depends on our target!
  - are repeated lines allowed in a receipt?
  - is the line order relevant?
- There are many different possible representations

Unnesting nested data structures

Receipts

Number	Date	Total
1235	12/10/2002	39,20
1240	13/10/2002	39,00

Courses

Receipt	Row	No	Description	Price
1235	1	3	Coperti	3,00
1235	2	2	Antipasti	6,20
1235	3	3	Primi	12,00
1235	4	2	Bistecche	18,00
1240	1	2	Coperti	2,00
…	…	…	…	…

Partial Information

- The relational data model has a rigid structure:
  - information is expressed in tuples
  - not all the tuples are accepted: the valid tuples respect a schema
- Data could not match the expected format

Partial Information: examples

Name	Middle Name	Surname
Franklin	Delano	Roosevelt
Winston		Churchill
Charles		De Gaulle
Josip		Stalin

Partial Information: viable solutions?

- We should not use specific values within the domain (0, empty string, “99”, ...):
  - it can be the case that there are no “unused” values available
  - such values could at some point become useful
  - we should track such “non representable” values in our software, in order to provide the expected meaning

Partial information in the relational model

- Rough but effective technique:
  - NULL value: remarks the absence of an expected value (however this value does not belong to the domain)
- t[A], for each attribute A, either maps into a value in dom(A) or is NULL.
- We have to define specific constraints to enforce when NULL values are not allowed

Different NULL types

- There are at least three different cases of missing values. E.g.:
  - *unknown* value
  - *inexistant* value
  - *uninformative* value
- There are no specific distinctions between such values in modern DBMSs

Too many NULLs

Students

Number	Surname	Name	Date of Birth
6554	Rossi	Mario	1978/12/05
8765	Neri	Paolo	1976/11/03
9283	Verdi	Luisa	1979/11/12
NULL	Rossi	Maria	1978/02/01

Exams

Student	Grade	Lecture
3456	30	04
NULL	24	02
NULL	28	01
6554	26	01

Lectures

Code	Name	Lecturer
01	Maths	Mario
02	**NULL**	**NULL**
04	Chemistry	Verdi

Integrity constraints

- Some database instances, while syntactically correct, may not represent any correct information for the target application.

Wrong representation

Exam

Student	Grade	Laude	Lectur e
276545	32		01
276545	30	laude	02
787643	27	laude	03
739430	24		04

Students

Number	Surname	Name
276545	Rossi	Mario
787643	Neri	Piero
787643	Bianchi	Luca

Integrity constraint

- Property that must be held in the instances in order to represent correct information for the target application
- A constraint could be interpreted as a boolean function (predicate).
  - it associates to each instance a boolean value of true or false.

Integrity constraints, why?

- A more accurate description of our real-world scenario
- Support the “data quality”
- Useful in Database Design
- Used in Databases’ query evaluation

Integrity constraints, observation

- DBMSs do not support all the types of constraints:

  we could state the provided types of constraints in our database and the DBMSs will prevent their violation

- When DBMSs do not support specific constraints, either the user or the programmer must *program* such constraints outside the DBMS.

Types of constraints

- Intra-relational constraints
  - over values (or domain constraint)
  - over tuples
- Inter-relational constraints

Example

Exam

Studente	Grade	Laude	Lectur e
276545	32		01
276545	30	laude	02
787643	**27**	laude	03
739430	24		04

Students

Number	Surname	Name
276545	Rossi	Mario
787643	Neri	Piero
787643	Bianchi	Luca

Tuple constraints

- They express rules on the values of each tuple, independently from the other tuples.
- Specific case:
  - **Domain constraints**: they involve a single attribute

Syntax and Examples

- A possible syntax:
  - A boolean expression made of atoms. Each atom can compare:
    - values of the attribute domain

      e.g. (Grade ≥ 18) AND (Grade ≤ 30)

    - arithmetic expressions on those values

      e.g. GrossPay = (Deductions + Net)

Tuple constraints, another example

Salary

Employee	GrossPay	Deductions	Net
Rossi	55.000	12.500	42.500
Neri	45.000	10.000	35.000
Bruni	47.000	11.000	36.000

GrossPay = (Deductions + Net)

Tuple constraints, violation

Salary

Employee	Gross Pay	Deductions	Net
Rossi	55.000	12.500	42.500
Neri	45.000	10.000	35.000
Bruni	50.000	11.000	36.000

GrossPay = (Deductions + Net)

Identifying tuples

Number	Surname	Name	Curriculum	Birth
27655	Rossi	Mario	CSE	78/12/05
78763	Rossi	Mario	CSE	76/11/03
65432	Neri	Piero	Mech En	79/07/10
87654	Neri	Mario	CSE	76/11/03
67653	Rossi	Piero	Mech En	78/12/05

- Each tuple has a unique Number ID
- There are no tuples that have the same Surname, Name and Date of Birth

Key

- A set of attributes univocally identifying tuples within a relation

More formally:

- A superkey is a set K of attributes on r, if there are not two distinct tuples t_1 and t_2 in r such that $t_1[K] = t_2[K]$
- A key is a minimal superkey of a relation (that is, it does not contain another superkey)

A key

Number	Surname	Name	Curriculum	Birth
27655	Rossi	Mario	CSE	78/12/5
78763	Rossi	Mario	CSE	76/11/3
65432	Neri	Piero	Mech En	79/07/10
87654	Neri	Mario	CSE	76/11/3
67653	Rossi	Piero	Mech En	78/12/5

The “Number” is a key:

- is a superkey
- contains only one attribute, hence it is minimal

Yet another key

Number	Surname	Name	Curriculum	Birth
27655	Rossi	Mario	CSE	78/12/5
78763	Rossi	Mario	CSE	76/11/3
65432	Neri	Piero	Mech En	79/7/10
87654	Neri	Mario	CSE	76/11/3
67653	Rossi	Piero	Mech En	78/12/5

The set Surname, Name, Birth is another key:

- it is a superkey…
- …and it is minimal

Another key??

Number	Surname	Name	Curricul	Birth
27655	Rossi	Mario	CSE	78/12/5
78763	Rossi	Mario	Construc.	76/11/3
65432	Neri	Piero	Mech En	79/7/10
87654	Neri	Mario	CSE	76/11/3
67653	Rossi	Piero	Mech En	78/12/5

There are no identical tuples over Surname and Curriculum:

- Surname and Curriculum is a key

But, is it always true?

Constraint, schema and states

- constraints correspond to real-world properties, modelled within the database
- are modelled over a schema (they apply over all the tuples)
- each schema could have a set of constraints; each stored tuple is to be considered correct (valid) because they satisfy all the constraints
- an instance could satisfy other constraints (“by chance”)

Students

Number	Surname	Name	Curricul	Birth

keys:

Number

Surname, Name, Birth

Number	Surname	Name	Curriculum	Birth
27655	Rossi	Mario	CSE	78/12/05
78763	Rossi	Mario	Construc.	76/11/03
65432	Neri	Piero	Mech En	79/07/10
87654	Neri	Mario	CSE	76/11/03
67653	Rossi	Piero	Mech En	78/12/05

- It is correct: all the constraints are satisfied
- Other constraints are satisfied ("by chance"):

  Surname, Curriculum is a key

Keys existence

- Each relation could not contain tuples with same values
- Each relation has a superkey that is the set of all the attributes
- Hence, it has (at least) one key

The importance of keys

- The existence of at least one key ensures the accessibility of each tuple in the database.
- Keys allow to correlate tuples across different relations: the relational data model is value based

Keys and null values

When the tuple contains null values, the values of the key do not allow to:

- identify the tuples
- easily create external references from other relations

Keys and null values: example

Number	Surname	Name	Curriculum	Birth
NULL	NULL	Mario	CSE	78/12/05
78763	Rossi	Mario	Construc	76/11/03
65432	Neri	Piero	Mech En	79/07/10
87654	Neri	Mario	CSE	NULL
NULL	Neri	Mario	NULL	78/12/05

The usage of null values in keys must be limited

Primary key

- Primary Keys do not allow NULL values
- Pointed out with an underline

Number	Surname	Name	Curric.	Birth
86765	NULL	Mario	Ing Inf	78/12/05
78763	Rossi	Mario	Ing Civile	76/11/03
65432	Neri	Piero	Ing Mecc	79/07/10
87654	Neri	Mario	Ing Inf	NULL
43289	Neri	Mario	NULL	78/12/05

Referential integrity

- information across different relations are correlated through common values.
- in particular, values of the keys (primary keys)
- such correlations must be "coherent"

Infringement

Code	Date	Policeman	Country	Plate
34321	95/02/01	3987	Italy	MI39548K
53524	95/03/04	3295	Italy	AJ046EZ
64521	96/04/05	3295	France	ET234RY
73321	98/02/05	9345	Finland	MMG-418

Policemen

ID	Surname	Name
3987	Rossi	Luca
3295	Neri	Piero
9345	Neri	Mario
7543	Mori	Gino

Infringment

Code	Date	Policeman	Country	Plate
34321	95/02/01	3987	Italy	MI39548K
53524	95/03/04	3295	Italy	AJ046EZ
64521	96/04/05	3295	France	ET234RY
73321	98/02/05	9345	Finland	MMG-418

Car

Country	Plate	Surname	Name
Italy	MI39548K	Rossi	Mario
Italy	AJ046EZ	Verdi	Giuseppe
France	ET234RY	Debussy	Claude
Finland	MMG-418	Sibelius	Jean

Referential integrity constraint

- A referential integrity constraint ("foreign key") between the attributes X of a relation R_1 and another relation R_2 , enforce the values on X in R_1 to appear as values in the primary key of R_2

Example

- Referential integrity constraint between:
  - the attribute Policeman from Infringement and the relation Policemen
  - the attributes Country and Plate from Infringement and the relation Car

Referential Integrity Constraint, violation

Infringment

Code	Date	Policeman	Country	Plate
34321	95/02/01	3987	Italy	MI39548K
53524	95/03/04	3295	Italy	AJ046EZ
64521	96/04/05	3295	France	ET234RY
73321	98/02/05	9345	Finland	MMG-418

Car

Country	Plate	Surname	Name
Italy	MI39548K	Rossi	Mario
France	AJ046EZ	Verdi	Giuseppe
France	ET234RY	Debussy	Claude
Finland	MMG-418	Sibelius	Jean

Referential Integrity Constraints, comments

- They have a crucial role in the “value based data model”
- When NULL values appears, constraints could be relaxed
- We could handle constraints violations ("side effects") with compensatory actions
- Be careful when constraints over more attributes are defined

Referential integrity and null values

Employees

Number	Surname	Project
34321	Rossi	IDEA
53524	Neri	XYZ
64521	Verdi	*NULL*
73032	Bianchi	IDEA

Projects

Code	Begin	Span	Cost
IDEA	01/2000	36	200
XYZ	07/2001	24	120
BOH	09/2001	24	150

Example: compensatory actions

- Example:
  - A tuple is deleted, causing the violation of a constraint

- "Standard" behaviour:
  - Restrict (reject the deletion)

- Possible compensatory actions:
  - Cascade removal
  - Introduction of null values: Set NULL
  - Introduction of default values

Delete Operation: cascade

Employees

Number	Surname	Project
34321	Rossi	IDEA
64521	Verdi	*NULL*
73032	Bianchi	IDEA

Projects

Code	Begin	Span	Cost
IDEA	01/2000	36	200
BOH	09/2001	24	150

Delete Operation, Set NULL

Employees

Number	Surname	Project
34321	Rossi	IDEA
53524	Neri	*NULL*
64521	Verdi	*NULL*
73032	Bianchi	IDEA

Projects

Code	Begin	Span	Cost
IDEA	01/2000	36	200
XYZ	07/2001	24	120
BOH	09/2001	24	150

Multiple Constraints

Crashes

Code	CountryA	PlateA	CountryB	PlateB
34321	Italy	AJ046EZ	Italy	MI39548K
53524	Finland	MMG-418	France	ET234RY

Car

Country	Plate	Surname	Name
Italy	MI39548K	Rossi	Mario
Italy	AJ046EZ	Verdi	Giuseppe
France	ET234RY	Debussy	Claude
Finland	MMG-418	Sibelius	Jean

Constraints over multiple attributes (2)

- Constraints betwen
  - Attributes StateA and PlateA from Crashes and the relation Car
  - Attributes StateB and PlateB from Crashes and the relation Car

- The order between the attributes is relevant