

Databases

The Relational Data Model

Danilo Montesi

danilo.montesi@unibo.it

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



Value-based: an Example

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

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

LECTURE	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 (1)

$$D_1 = \{a, b\}$$

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

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 (2)

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

Logical Relation: Properties

- A **logical relation** is a set of ordered tuples:
 - (d_1, \dots, d_n) such that $d_1 \in D_1, \dots, d_n \in D_n$
- 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	Psg	0	2
Psg	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 (1)

- 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	Psg	0	2
Psg	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	Psg	0	2
Psg	Real	0	1

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

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

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

LECTURE	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



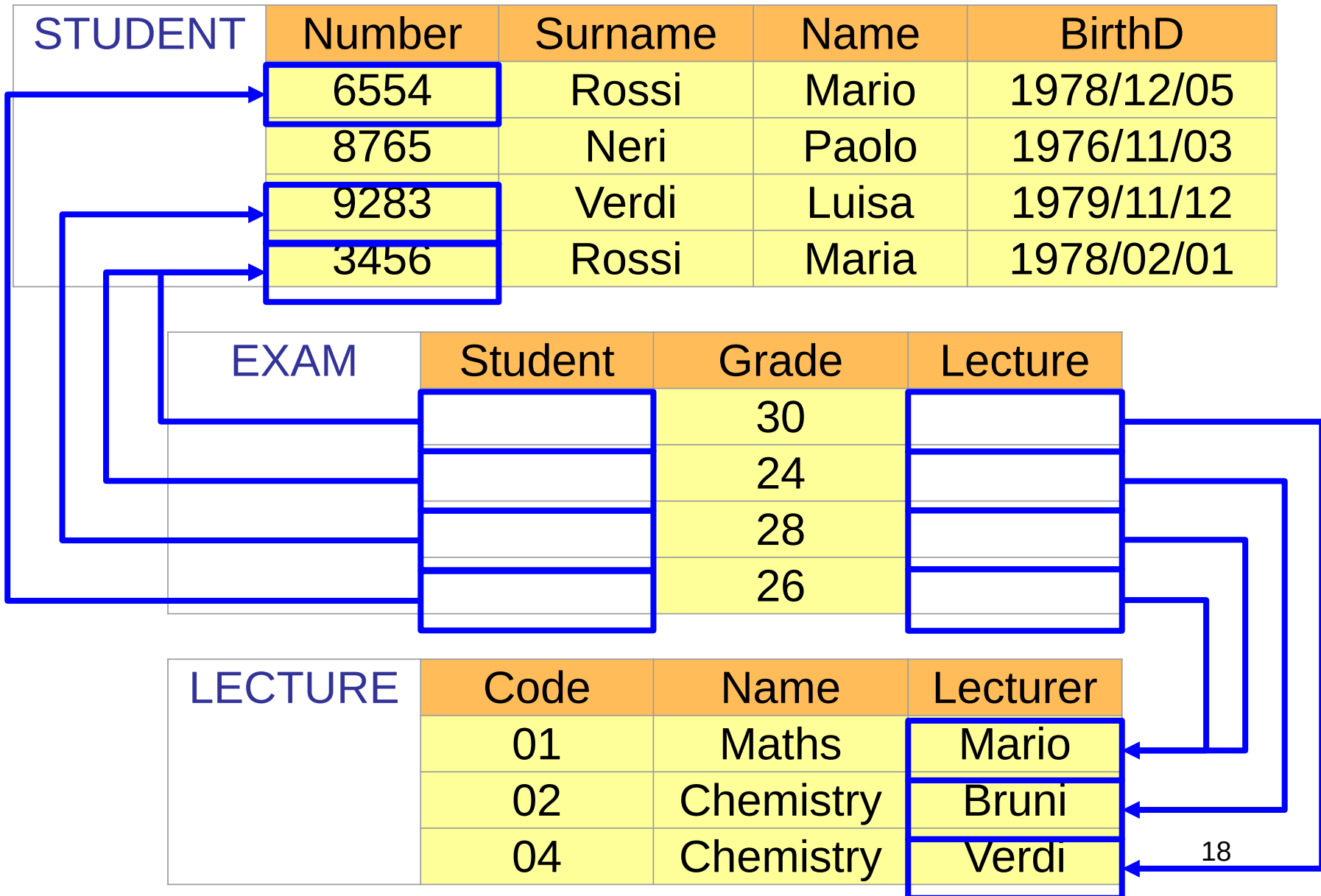
Example: Value-based

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

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

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

Example: Pointers





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 from 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 (1)

■ Schema of a relation:

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

$$R(A_1, \dots, A_n)$$

■ Schema of a database:

a set of schemas (relation):

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

Example

Attributes



■ Schema of a relation:

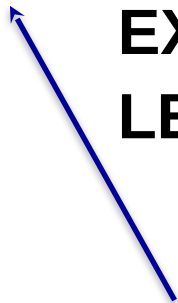
STUDENTS (Number, Surname, Name, Year of Birth)



Relation

■ Schema of a database:

$R = \{\textbf{STUDENTS (Number, Surname, Name, Year of Birth)},$
 $\textbf{EXAMS (Student, Grade, Lecture)},$
 $\textbf{LECTURES (Code, Name, Lecturer)}\}$



Relational database schema

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*

$$t[\text{Name}] = \text{Mario}$$

Definitions (3)

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



Example of a Relation Instance

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



Example of a Relational Database Instance

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

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

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



Relations with Just One Attribute

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

WORKERSTU	Number
	6554
	3456



Nested Data Structures (1)

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

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

Nested Data Structures (1)

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

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



Nested Data Structures (1)

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

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



Nested Data Structures (2)

RECEIPTS					
Num	Date	Qty	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

COURSE	Receipt	Qty	Description	Price
	1235	3	Cover Charge	3,00
	1235	2	Appetizer	6,20
	1235	3	Soup	12,00
	1235	2	Meat Loaf	18,00
	1240	2	Cover Charge	2,00

Nested Relations: Some Thoughts

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

- 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

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

COURSE	Receipt	Row	Qty	Description	Price
	1235	1	3	Cover Charge	3,00
	1235	2	2	Appetizer	6,20
	1235	3	3	Soup	12,00
	1235	4	2	Meat Loaf	18,00
	1240	1	2	Cover Charge	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: an Example

Name	MidName	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

- 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 $\text{dom}(A)$ 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, that is:
 - *unknown* value
 - *inexistant* value
 - *uninformative* value
- There are no specific distinctions between such values in modern DBMSs



Too Many NULLs

STUDENT	Number	Surname	Name	BirthD
	6554	Rossi	Mario	1978/12/05
	8765	Neri	Paolo	NULL
	9283	Verdi	Luisa	1979/11/12
	NULL	Rossi	Maria	1978/02/01

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

LECTURE	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	Lecture
	276545	32		01
	276545	30	yes	02
	787643	27	yes	03
	739430	24		04

STUDENT	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 *code* such constraints outside the DBMS

Types of Constraints

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

Example

**Intra-relational
over values**

EXAM	Student	Grade	Laude	Lecture
	276545	32		01
	276545	30	yes	02
	787643	27	yes	03
	739430	24		04

Inter-relational

**Intra-relational
over tuples**

STUDENT	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
 $(Grade \geq 18) \text{ AND } (Grade \leq 30)$
 - arithmetic expressions on those values
 $GrossPay = (Deductions + Net)$



Tuple Constraints: an 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

$$\text{GrossPay} = (\text{Deductions} + \text{Net})$$

Tuple Constraints: Violation

SALARY	Employee	GrossPay	Deductions	Net
	Rossi	55'000	12'500	42'500
	Neri	45'000	10'000	35'000
	Bruni	50'000	11'000	36'000

$$\text{GrossPay} = (\text{Deductions} + \text{Net})$$

Example

Intra-relational
over values

EXAM	Student	Grade	Laude	Lecture
	276545	32		01
	276545	30	yes	02
	787643	27	yes	03
	739430	24		04

Inter-relational

Intra-relational
over tuples

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

Identifying Tuples

Number	Surname	Name	Curriculum	Birth
27655	Rossi	Mario	CSE	78/12/05
78763	Rossi	Mario	BioTech	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

Superkey & 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	78/12/05
78763	Rossi	Mario	BioTech	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

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	78/12/05
78763	Rossi	Mario	BioTech	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

The set *Surname, Name, Birth* is another key:

- it is a superkey ...
- ... and it is minimal

Another Key??

Number	Surname	Name	Curriculum	Birth
27655	Rossi	Mario	CSE	78/12/05
78763	Rossi	Mario	BioTech	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

There are no identical tuples over *Surname* and *Curriculum*:

- *Surname* and *Curriculum* is a key

But, is it always true?

Constraint, Schema and Instances

- Constraints correspond to real-world properties, modelled within the database
- They 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”)



Constraint and Schema: an Example

STUDENTS

Number	Surname	Name	Curriculum	Birth
--------	---------	------	------------	-------

Keys:

Number

Surname, Name, Birth

Constraints Satisfaction: an Example

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

The usage of NULL values in keys must be limited!

Primary Key

- **Primary Keys (PK)** do not allow NULL values
- Pointed out with an underline

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

Referential Integrity

- Information across different relations are correlated through common values (remember the value-based model?)
- In particular, values of the keys (i.e., **primary keys**)
- Such correlations must be “coherent”



Referential Integrity: an Example (1)

INFRINGEMENT	<u>Code</u>	Date	Policeman	Country	Plate
	34321	95/02/01	3987	Italy	MI395BK
	53524	95/03/04	3295	Italy	AJ046EZ
	64521	96/04/05	3295	France	ET234RY
	73321	98/02/05	9345	Finland	MMG-418

POLICEMAN	<u>Id</u>	Surname	Name
	3987	Rossi	Luca
	3295	Neri	Piero
	9345	Neri	Mario
	7543	Mori	Gino



Referential Integrity: an Example (2)

INFRINGEMENT	<u>Code</u>	Date	Policeman	Country	Plate
	34321	95/02/01	3987	Italy	MI395BK
	53524	95/03/04	3295	Italy	AJ046EZ
	64521	96/04/05	3295	France	ET234RY
	73321	98/02/05	9345	Finland	MMG-418

CAR	<u>Country</u>	<u>Plate</u>	Surname	Name
	Italy	MI395BK	Rossi	Mario
	Italy	AJ046EZ	Verdi	Giuseppe
	France	ET234RY	Debussy	Claude
	Finland	MMG-418	Sibelius	Jean

Referential Integrity Constraint

- A **referential integrity** constraint, i.e., **foreign key (FK)**, 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
- An attribute A in R_1 is a foreign key that references the relation R_2 if it satisfies the following rules:
 1. The attribute A has the same domain as the primary key attribute of R_2
 2. A value of A in a tuple t_1 in R_1 either is NULL or occurs as a value for some tuple t_2 in R_2 , such that $t_1[A] = t_2[PK]$ (a.k.a. $t_1[FK] = t_2[PK]$)

Example

- Referential integrity constraint between:
 - the attribute *Policeman* from **INFRINGEMENT** and the relation **POLICEMAN**
 - the attributes *Country* and *Plate* from **INFRINGEMENT** and the relation **CAR**






Referential Integrity Constraint: Violation

INFRINGEMENT	<u>Code</u>	Date	Policeman	Country	Plate
	34321	95/02/01	3987	Italy	MI395BK
	53524	95/03/04	3295	Italy	AJ046EZ
	64521	96/04/05	3295	France	ET234RY
	73321	98/02/05	9345	Finland	MMG-418

CAR	<u>Country</u>	<u>Plate</u>	Surname	Name
	Italy	MI395BK	Rossi	Mario
	France	AJ046EZ	Verdi	Giuseppe
	France	ET234RY	Debussy	Claude
	Finland	MMG-418	Sibelius	Jean

Referential Integrity Constraint: 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



EMPLOYEE	<u>Number</u>	Surname	Project
	34321	Rossi	IDEA
	53524	Neri	XYZ
	64521	Verdi	NULL
	73032	Bianchi	IDEA

PROJECT	<u>Code</u>	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:

- Cascading removal
- Introduction of:
 - NULL values
 - default values



Delete Operation: Cascade

EMPLOYEE	<u>Number</u>	Surname	Project
	34321	Rossi	IDEA
	64521	Verdi	NULL
	73032	Bianchi	IDEA

PROJECT	<u>Code</u>	Begin	Span	Cost
	IDEA	01/2000	36	200
	BOH	09/2001	24	150



Delete Operation: Set NULL

EMPLOYEE	<u>Number</u>	Surname	Project
	34321	Rossi	IDEA
	53524	Neri	NULL
	64521	Verdi	NULL
	73032	Bianchi	IDEA

PROJECT	<u>Code</u>	Begin	Span	Cost
	IDEA	01/2000	36	200
	BOH	09/2001	24	150



Constraints over Multiple Attributes (1)

CRASH	<u>Code</u>	Date	StateA	PlateA	StateB	PlateB
	34321	01/02/1995	Italy	AJ046EZ	Italy	MI395BK
	53524	05/04/1996	Finland	MMG-418	France	ET234RY

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



Constraints over Multiple Attributes (2)

CRASH	<u>Code</u>	Date	StateA	PlateA	StateB	PlateB
	34321	01/02/1995	Italy	AJ046EZ	Italy	MI395BK
	53524	05/04/1996	Spain	4189HEJ	France	ET234RY

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

Constraints over Multiple Attributes (3)

- Constraints between
 - Attributes *StateA* and *PlateA* from **CRASH** and the relation **CAR**
 - Attributes *StateB* and *PlateB* from **CRASH** and the relation **CAR**
- The order between the attributes is relevant