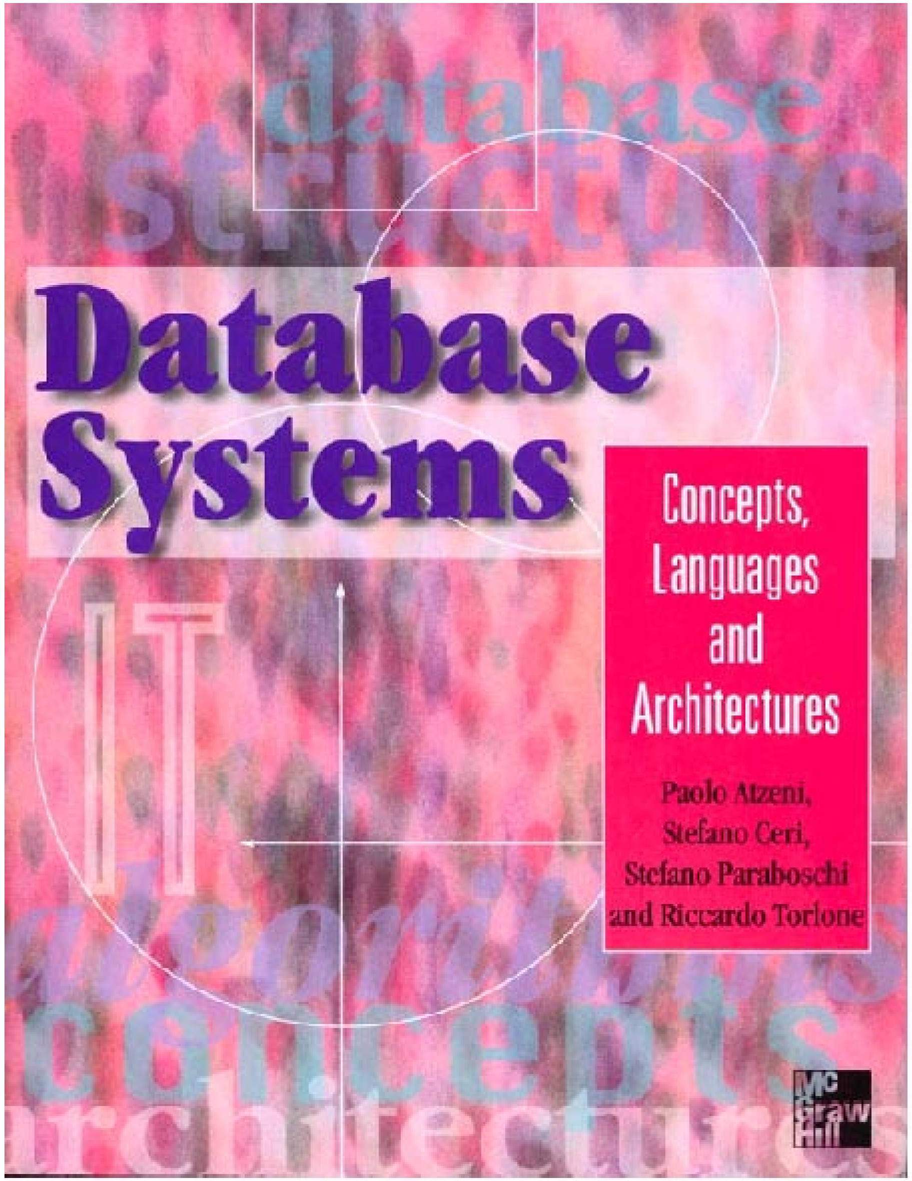

Chapter 2

The relational model

Logical data models

- The traditional ones:
  - **hierarchical**
  - **network**
  - **relational**
- Hierarchical and network closer to physical structures, relational higher level
  - in the relational model we have only values: even references between data in different sets (relations) are represented by means of values
  - in the hierarchical and network model there are explicit references (pointers)
- More recently, the **object** model has been introduced

The relational model

- Proposed by E. F. Codd in 1970 in order to support data independence
- Made available in commercial DBMSs in 1981 (it is not easy to implement data independence efficiently and reliably!)
- It is based on (a variant of) the mathematical notion of **relation**
- Relations are naturally represented by means of tables

Mathematical relations

- D_1, D_2, …, D_n (n sets, not necessarily distinct)
- **cartesian product** $D_1 \times D_2 \times \ldots \times D_n$:
  - the set of all (ordered) n-tuples (d_1, d_2, …, dn) such that $d_1 \in D_1$, $d_2 \in D_2$, …, $d_n \in D_n$
- a **mathematical relation** on D_1, D_2, …, D_n:
  - a subset of the cartesian product $D_1 \times D_2 \times \ldots \times D_n$.
- D_1, D_2, …, D_n are the **domains of the relation**
- n is the **degree** of the relation
- the number of n-tuples is the **cardinality** ofthe relation; in practice, it is always finite

Mathematical relations, properties

- A mathematical relation is a **set** of **ordered** n-tuples

  (d1, d2, …, dn) tali che d1 $\in$ D1, d2 $\in$ D2, …, dn $\in$ Dn
- a set, so:
  - there is no ordering between n-tuples
  - the n-tuples are distinct from one another
- the n-tuples are **ordered**: the i-th value come from the i-th domain: so there is an ordering among the domains

Mathematical relation, example

$$games \subseteq string \times string \times integer \times integer$$

Juve	Lazio	3	1
Lazio	Milan	2	0
Juve	Roma	1	2
Roma	Milan	0	1

- Each of the domains has two **roles**, distinguished by means of position
- The structure is **positional**

Relations in the relational data model

- We would like to have a **non-positional** structure
- We associate a unique name (**attribute**) with each domain; it “describes” the role of the domain
- In the tabular representation, attributes are used as column headings

HomeTeam	VisitingTeam	HomeGoals	VisitorGoals
Juve	Lazio	3	1
Lazio	Milan	2	0
Juve	Roma	1	2
Roma	Milan	0	1

Formalizing

- The correspondence between attributes and domains:

  DOM: $X \rightarrow \mathcal{D}$

  (where X is a set of attributes and $\mathcal{D}$ the set of all domains)
- A **tuple** on X is a function that associates with each A in X a value from the domain **DOM**(A)
- A **relation** on X is a set of tuples on X

Notation

- t[A] (or t. A) denotes the value on A of a tuple t
- In the example, if t is the first tuple in the table

  t[VisitingTeam] = Lazio
- The same notation is extended to sets of attributes, thus denoting tuples:

  t[VisitingTeam,VisitorGoals]

  is a tuple on two attributes

The relational model is “value-based”

- References between data in different relations are represented by means of values of the domains

Students

RegNum	Surname	FirstName	BirthDate
6554	Rossi	Mario	5/12/1978
8765	Neri	Paolo	3/11/1976
9283	Verdi	Luisa	12/11/1979
3456	Rossi	Maria	1/2/1978

Exams

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

Courses

Code	Title	Tutor
01	Analisi	Neri
02	Chimica	Bruni
04	Chimica	Verdi

Students

RegNum	Surname	FirstName	BirthDate
6554	Rossi	Mario	5/12/1978
8765	Neri	Paolo	3/11/1976
9283	Verdi	Luisa	12/11/1979
3456	Rossi	Maria	1/2/1978

Exams

Student	Grade	Course
	30	
	24	
	28	
	26	

Courses

Code	Title	Tutor
01	Analisi	Neri
02	Chimica	Bruni
04	Chimica	Verdi

Advantages of a value-based structure

- Independence of physical structures
- Only information that is relevant from the application point of view
- Easy transferrability of data between systems

Notes:

- pointers usually exist at the physical level, but they are not visible at the logical level
- object-identifiers in object databases show some features of pointers, at a higher level of abstraction

Definitions

Relation schema:

a name (of the relation) R with a set of attributes $A_1,..., A_n$

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

Database schema:

a set of relation schemas with different names

$$\mathbf{R} = \{R_1(X_1), ..., R_n(X_n)\}$$

Relation (instance) on a schema R(X):

set r of tuples on X

Database (instance) on a schema $\mathbf{R}= \{R_1(X_1), ..., R_n(X_n)\}$:

set of relations $\mathbf{r} = \{r_1,..., r_n\}$ (with r_i relation on R_i)

Examples

- Relations on a single attribute are admissible

Students

RegNum	Surname	FirstName	BirthDate
6554	Rossi	Mario	5/12/1978
8765	Neri	Paolo	3/11/1976
9283	Verdi	Luisa	12/11/1979
3456	Rossi	Maria	1/2/1978

Workers

RegNum
6554
8765

Nested structures

Da Mario		
Receipt No: 1357		
Date: 5/5/92		
3	covers	3.00
2	hors d'oeuvre	5.00
3	first course	9.00
2	steak	12.00
	Total:	29.00

Da Mario		
Receipt No: 2334		
Date: 4/7/92		
2	covers	2.00
2	hors d'oeuvre	2.50
2	first course	6.00
2	bream	15.00
2	coffee	2.00
	Total:	27.50

Da Mario		
Receipt No: 3007		
Date: 4/8/92		
2	covers	3.00
2	hors d'oeuvre	6.00
3	first course	8.00
1	bream	7.50
1	salad	3.00
2	coffee	2.00
	Total:	29.50

Nested structures by means of relations

Receipts

Number	Date	Total
1357	5/5/92	29.00
2334	4/7/92	27.50
3007	4/8/92	29.50

Details

Number	Quantity	Description	Cost
1357	3	Covers	3.00
1357	2	Hors d'oeuvre	5.00
1357	3	First course	9.00
1357	2	Steak	12.00
2334	2	Covers	2.00
2334	2	Hors d'oeuvre	2.50
2334	2	First course	6.00
2334	2	Bream	15.00
2334	2	Coffee	2.00
3007	2	Covers	3.00
3007	2	Hors d'oeuvre	6.00
3007	3	First course	8.00
3007	1	Bream	7.50
3007	1	Salad	3.00
3007	2	Coffee	2.00

- Have we represented all details of receipts?
- Well, it depends on what we are really interested in:
  - does the order of lines matter?
  - could we have duplicate lines in a receipt?

- If needed, an alternative organization is possible

More detailed representation

Receipts

Number	Date	Total
1357	5/5/92	29.00
2334	4/7/92	27.50
3007	4/8/92	29.50

Details

Number	Line	Quantity	Description	Cost
1357	1	3	Covers	3.00
1357	2	2	Hors d'oeuvre	5.00
1357	3	3	First course	9.00
1357	4	2	Steak	12.00
2334	1	2	Covers	2.00
2334	2	2	Hors d'oeuvre	2.50
2334	3	2	First course	6.00
2334	4	2	Bream	15.00
2334	5	2	Coffee	2.00
3007	1	2	Covers	3.00
3007	2	2	Hors d'oeuvre	6.00
3007	3	3	First course	8.00
3007	4	1	Bream	7.50
3007	5	1	Salad	3.00
3007	6	2	Coffee	2.00

Incomplete information

- The relational model impose a rigid structure to data:
  - information is represented by means of tuples
  - tuples have to conform to relation schemas
- In practice, the available data need not conform to the required formats

Incomplete information: motivation

(County towns have government offices, other cities do not)

- Florence is a county town; so it has a government office, but we do not know its address
- Tivoli is not a county town; so it has no government office
- Prato has recently become a county town; has the government office been established? We don‘t know

City	GovtAddress
Roma	Via IV novembre
Florence	
Tivoli	
Prato	

Incomplete information: solutions?

- We should not (despite what often happens) use domain values (zero, 99, empty string, etc.) to represent lack of information:
  - there need not be “unused” values
  - “unused” values could become meaningful
  - in programs, we should be able to distinguish between actual values and placeholders (for example: calculate the average age of a set of people, where 0 is used for unknown ages!)

Incomplete information in the relational model

- A simple but effective technique is used:
  - **null value:** a special value (not a value of the domain) denotes the absence of a domain value
- We could (and often should) put restrictions on the presence of null values in tuples (we will see later)

Types of null value

- (at least) three
  - **unknown value:** there is a domain value, but it is not known (Florence)
  - **non-existent value:** the attribute is not applicable for the tuple (Tivoli)
  - **no-information value:** we don‘t know whether a value exists or not (Prato); this is the disjunction (logical or) of the other two
- DBMSs do not distinguish between the types: they implicitly adopt the no-information value

A meaningless database instance

Exams

RegNum	Name	Course	Grade	Honours
6554	Rossi	B01	K	
8765	Neri	B03	C	
3456	Bruni	B04	B	honours
3456	Verdi	B03	A	honours

Courses

Code	Title
B01	Physics
B02	Calculus
B03	Chemistry

- grades are between A and F
- honours can be awarded only if grade is A
- different students must have different registration numbers
- exames must refer to existing courses

Integrity constraints

- **integrity constraint**: a property that must be satisfied by all meaningful database instances;
- it can be seen as a **predicate**: a database instance is **legal** if it satisfies all integrity constraints
- types of constraints
  - intrarelational constraints; special cases:
    - domain constraints
    - tuple constraints
  - interrelational constraints

Integrity constraints, motivations

- Useful to describe the application in greater detail
- A contribution to “data quality”
- An element in the design process (we will discuss “normal forms”)
- Used by the system in choosing the strategy for query processing

Note:

- it is not the case that all properties can be described by means of integrity constraints

Tuple constraints

- express conditions on the values of each tuple, independently of other tuples
- a possible syntax: boolean expressions with atoms that compare attributes, constants or expressions over them
- **domain constraint**: a tuple constraint that involve a single attribute
- a domain constraint

  (Grade ≥ "A") AND (Grade ≤ "F")

- a tuple constraint

  (NOT (Honours = "honours"))OR (Grade = "A")

- a tuple constraint (on another schema) with expressions:

  Net = Amount - Deductions

Unique identification of tuples

RegNum	Surname	FirstName	BirthDate	DegreeProg
284328	Smith	Luigi	29/04/59	Computing
296328	Smith	John	29/04/59	Computing
587614	Smith	Lucy	01/05/61	Engineering
934856	Black	Lucy	01/05/61	Fine Art
965536	Black	Lucy	05/03/58	Fine Art

- the registration number identifies students:
  - there is no pair of tuples with the same value for RegNum
- personal data identifies students:
  - there is no pair of tuples with the same values on each of Surname, FirstName, BirthDate

Keys

- **Key** :
  - a set of attributes that uniquely identifies tuples in a relation
- more precisely:
  - a set of attributes K is a **superkey** for a relation r if r does not contain two distinct tuples t_1 and t_2 with $t_1[K]=t_2[K]$;
  - K is a **key** for r if K is a minimal superkey (that is, there exists no other superkey K’ of r that is contained in K as proper subset)

RegNum	Surname	FirstName	BirthDate	DegreeProg
284328	Smith	Luigi	29/04/59	Computing
296328	Smith	John	29/04/59	Computing
587614	Smith	Lucy	01/05/61	Engineering
934856	Black	Lucy	01/05/61	Fine Art
965536	Black	Lucy	05/03/58	Fine Art

- RegNum is a key:
  - RegNum is a superkey
  - it contains a sole attribute, so it is minimal
- Surname, Firstname, BirthDate is another key:
  - Surname, Firstname, BirthDate form a superkey
  - no proper subset is also a superkey

RegNum	Surname	FirstName	BirthDate	DegreeProg
296328	Smith	John	29/04/59	Computing
587614	Smith	Lucy	01/05/61	Engineering
934856	Black	Lucy	01/05/61	Fine Art
965536	Black	Lucy	05/03/58	Engineering

- there is no pair of tuples with the same values on both Surname and DegreeProg:
  - in each programme students have different surnames;
  - Surname and DegreeProg form a key for this relation
- is this a general property?
  - No! There could be students with the same surname in the same programme

Keys, schemas, and instances

- Constraints correspond to properties in the real world to be modelled by our database
- therefore, they are relevant at the schema level (wrt the whole set of instances)
  - we associate with a schema a set of constraints, and we consider as correct (legal, valid, …) the instances that satisfy all the constraints
  - individual instances could satisfy ("by chance") other constraints

Existence of keys

- Relations are sets; therefore each relation is composed of distinct tuples: the whole set of attributes is a superkey;
- so each relation has a superkey; since the set of attributes is finite, each relation schema has at least a key:
  - the whole set is either a key
  - or it contains a (smaller superkey), and for it we can repeat the argument, over a smaller set

Importance of keys

- The existence of keys guarantees that each piece of data in the database can be accessed
- Keys are the major feature that allows us to say that the relational model is “value-based”

Keys and null values

- If there are nulls, keys do not work that well
  - they do not guarantee unique identification
  - they do not allow to establish correspondences between data in different relations

RegNum	Surname	FirstName	BirthDate	DegreeProg
NULL	Smith	John	NULL	Computing
587614	Smith	Lucy	01/05/61	Engineering
934856	Black	Lucy	NULL	NULL
NULL	Black	Lucy	05/03/58	Engineering

  - How do we access the first tuple?
  - Are the third and fourth tuple the same?

Primary keys

- The presence of nulls in keys has to be limited
- Practical solution: for each relation we select a **primary key** on which nulls are not allowed
  - notation: the attributes in the primary key are underlined
- References between relations are realized through primary keys

RegNum	Surname	FirstName	BirthDate	DegreeProg
643976	Smith	John	NULL	Computing
587614	Smith	Lucy	01/05/61	Engineering
934856	Black	Lucy	NULL	NULL
735591	Black	Lucy	05/03/58	Engineering

Primary keys: do we always have them?

- In most cases we do have reasonable primary keys
- In other cases we don't:
  - we need to introduced new attributes (identifying "codes")
- Note that most of the "obvious" codes we have now (social security number, student number, area code, …) were introduced (possibly before the adoption of databases) with the same goal: unambiguous identification of objects

Referential constraints ("foreign keys")

- Pieces of data in different relations are correlated by means of values of (primary) keys
- Referential integrity constraints are imposed in order to guarantee that the values refer to actual values in the referenced relation

A database with referential constraints

Offences

Code	Date	Officer	Dept	Registartion
143256	25/10/1992	567	75	5694 FR
987554	26/10/1992	456	75	5694 FR
987557	26/10/1992	456	75	6544 XY
630876	15/10/1992	456	47	6544 XY
539856	12/10/1992	567	47	6544 XY

Officers

RegNum	Surname	FirstName
567	Brun	Jean
456	Larue	Henri
638	Larue	Jacques

Cars

Registration	Dept	Owner	…
6544 XY	75	Cordon Edouard	…
7122 HT	75	Cordon Edouard	…
5694 FR	75	Latour Hortense	…
6544 XY	47	Mimault Bernard	…

Referential constraints

- A **referential constraint** imposes to the values on a set X of attributes of a relation R_1 to appear as values for the primary key of another relation R_2
- In the example, we have referential constraints between
  - the attribute Officer of Offences and relation Officers
  - the attributes Registration and Department of Offences and relation Cars

Database that violates referential constraints

- Offences

Code	Date	Officer	Dept	Registartion
987554	26/10/1992	456	75	5694 FR
630876	15/10/1992	456	47	6544 XY

Officers

RegNum	Surname	FirstName
567	Brun	Jean
638	Larue	Jacques

Cars

Registration	Dept	Owner	…
7122 HT	75	Cordon Edouard	…
5694 FR	93	Latour Hortense	…
6544 XY	47	Mimault Bernard	…

Referential constraints: comments

- Referential constraints play an essential role in the issue "the relational model is value-based."
- It is possible to have features that support the management of referential constraints ("actions" activated by violations)
- In presence of null values definitions have to be adapted
- Care is needed in case of constraints that involve two or more attributes

Integrity constraints can get intricated

Accidents

Code	Dept1	Registration1	Dept2	Registration1
6207	75	6544 XY	93	9775 GF
6974	93	5694 FR	93	9775 GF

Cars

Registration	Dept	Owner	…
7122 HT	75	Cordon Edouard	…
5694 FR	93	Latour Hortense	…
9775 GF	93	LeBlanc Pierre	
6544 XY	75	Mimault Bernard	…

- we have two referential constraints
  - from Registration1, Dept1 to Cars
  - from Registration2, Dept2 to Cars

  Note that ordering in the set of attributes is essential!

  The key of cars is Registration, Dept and not Dept, Registration