Week 2 – Part 1: The Relational Model

The Relational Model

Mathematical Relations

Attributes and Database Schema

Null Values and Database Constraints

Keys, Primary and Foreign Keys

The Relational Model

- Proposed by E. F. Codd in 1970 as a data model which strongly supports 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 represented as tables.

Mathematical Relations

- Given sets $D_1, D_2, \ldots, D_n$, not necessarily distinct.
- The ***Cartesian product*** $D_1 x D_2 x \ldots x D_n$ is the set of all (ordered) n-tuples $<d_1, d_2, \ldots, dn>$ such that $d_1 \in D_1, d_2 \in D_2, \ldots, d_n \in D_n$
- A ***mathematical relation*** on $D_1, D_2, \ldots, D_n$ is a subset of the Cartesian product $D_1 x D_2 x \ldots x D_n$.
- $D_1, D_2, \ldots, D_n$ are the ***domains*** of the relation.
- n is the ***degree*** of the relation.
- The number of n-tuples in a given relation is the ***cardinality*** of that relation; the cardinality of a relation is always finite.

An Example

`Games ⊆ String × String × Integer × Integer`

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

- Note that `String` and `Integer` each play two ***roles***, distinguished by means of position.
- The structure of a relation is ***positional***.

Attributes

- We would like to have a ***non-positional*** structure for relations. To do so, we associate a unique name (***attribute***) with each domain of a relation which 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

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, <Lazio,1>
- More generally, if X is a sequence of attribute names $A_1,...A_n$, t[X] is $<t[A_1],t[A_2],...t[A_n]>$

Value-Based References

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 Value-Based References

- Value-based references lead to independence from physical structures, such as pointers.
- Pointers are implemented differently on different hardware, inhibit portability of a database.

Notes:

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

Definitions

Relation schema:

A name (of the relation) R with a set of attributes $A_1,\ldots, A_n$: $R(A_1,\ldots, A_n)$

Database schema:

A set of relation schemas with different names

$D = \{R_1(X_1), \ldots, R_n(X_n)\}$

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

Set r of tuples on X

Database (instance) on a schema $D= \{R_1(X_1), \ldots, R_n(X_n)\}$:

Set of relations $r = \{r_1,\ldots, r_n\}$ (with r_i relation on R_i)

Examples

- Relations on a single attribute are admissible:

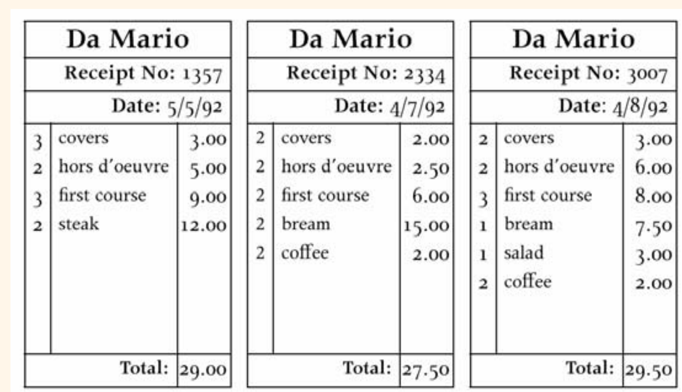
Students

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

Representating Nested Structures

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

Receipts

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

Questions

- Have we represented all details of receipts?
- Well, it depends on what we are really interested in:

What else could we require?

- If needed, an alternative organization is possible

More Detailed Representation

Details

How could we Preserve the Detail sequence Or allow duplicates?

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

Receipts

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

Incomplete Information

- The relational model imposes a rigid structure on data:
  - information is represented by means of tuples;
  - tuples **must** conform to relation schemas.
- In practice, available data need not conform to the required formats. In particular, values of attributes **may be missing** for a particular tuple we want to add to a relational database.

Incomplete information: Motivation

(County towns have government offices, other towns 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 use domain values (0, 99, empty string, etc.) to represent lack of information:

- Using **unused values** may lead to **ambiguity** and confusion.
- Unused values **could become meaningful**.
- Within applications, we should be able to distinguish between **actual values** and **placeholders**.
- For example, in order to calculate the average age of a set of people, use `50` as **default value** for unknown ages!

Incomplete Information in the Relational Model

- A simple but effective technique is adopted by the Relational Model: use ***null values***.
- A ***null value*** is a special value (i.e., not a value of the domain) which denotes the absence of a domain value.
- We could (and often should) put restrictions on the presence of null values in tuples (more on this later.)

Types of Null Values

- At least three different types are useful:
  - ***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 these types: they implicitly adopt the no-information value.

A Meaningless Database

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

WHAT ARE SOME PROBLEMS

WITH THIS DATABASE?

Integrity Constraints

- An ***integrity constraint*** is a property that must be satisfied by all meaningful database instances.
- A constraint can be seen as a ***predicate***; a database is ***legal*** if it satisfies all integrity constraints.
- Types of constraints
  - Intra-relational constraints, with domain constraints and tuple constraints as special cases;
  - Inter-relational constraints.

Rationale for Integrity Constraints

- Useful for describing the application in greater detail.
- Contribute to ***data quality***.
- An element in the design process; we will discuss later ***normal forms***.
- Used by the system in choosing a strategy for query processing

Note: It is not the case that all desirable properties of the data in a database can be described by means of integrity constraints!

e.g., "data in the relation **Employee** must be valid"

Tuple and Domain Constraints

- A ***tuple constraint*** expresses conditions on the values of each tuple, independently of other tuples.
- For example,

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

- Another example (***derivation rule***)

Net = Amount-Deductions

- A ***domain constraint*** is a tuple constraint that involves a single attribute

e.g., **(Grade ≤ 'A') AND (Grade ≥ 'F')**

Unique Identification for 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

- Registration number identifies students, i.e., there is no pair of tuples with the same value for **`RegNum`**.
- Personal data could identify students as well, i.e., there is no pair of tuples with the same values for all of **`Surname`**, **`FirstName`**, **`BirthDate`**.

Keys

- A ***key*** is 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 can not contain two distinct tuples t_1 and t_2 such that $t_1[K]=t_2[K]$;
  - 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.)

An Example

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: i.e., **`RegNum`** is a superkey and it contains a sole attribute, so it is minimal.
- **`Surname`**, **`Firstname`**, **`BirthDate`** is another key: the three attributes form a superkey and there is no proper subset that is also a superkey.

Beware!

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`**;

  i.e., in each programme students have different surnames; can we conclude that **`Surname`** and **`DegreeProg`** form a key for this relation?
- No! There ***could be*** students with the same surname in the same programme

Existence of Keys

- Relations are sets; therefore **each relation is composed of distinct tuples**.
- It follows that the whole set of attributes for a relation defines a **superkey**.
- Therefore **each relation has a key**, which is the set of all its attributes, or a subset thereof.
- The existence of keys guarantees that each piece of data in the database can be accessed,
- Keys are a major feature of the Relational Model and allow us to say that it 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 help in establishing 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

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

Primary Keys

- The presence of nulls in keys has to be limited.
- Each relation must have a ***primary key*** on which nulls are not allowed,
- Notation: the attributes of 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

Do we Always Have Primary Keys?

- In most cases we do have reasonable primary keys.
- In other cases we don't, so we need to introduced new attributes by identifying ***codes***.
- Note that most of the "obvious" codes we have now (social security number, student number, area code, …) were introduced ***before*** the adoption of databases with the same goal in mind, i.e. to offer an unambiguous identification of things.

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 existing tuples in the referenced relation.
- For example, if the manager of the employee with employee# 76544 is an employee with employee# 87233, there better be an employee with such an employee number.
- Also called ***inclusion dependencies***.

Example of Referential Constraints

Offences

Code	Date	Officer	Dept	Registration
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*** requires that the values on a set X of attributes of a relation R_1 must appear as values for the primary key of another relation R_2.
- In such a situation, we say that X is a foreign key of relation R_1.
- In the previous example, we have referential constraints between the attribute `Officer` of the relation `Offences` and the relation ; also between the attributes `Registration` and `Department` of `Offences` and the relation

Violation of Referential Constraints

Offences

Code	Date	Officer	Dept	Registration
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 important role in making the relational model value-based.
- It is possible to have **features** that support the management of referential constraints ("actions" activated by violations).
- Care is needed in case of referential constraints that involve two or more attributes.

Complications with Constraints

Accidents

Code	Dept1	Registration1	Dept2	Registration2
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	…

- Here we have two referential constraints for **`Accidents`**: from **`Registration1`**, **`Dept1`** to **`Cars`**; also from **`Registration2`**, **`Dept2`** to **`Cars`**.