

Relational databases



Stefano Montanelli
Department of Computer Science
Università degli Studi di Milano
stefano.montanelli@unimi.it

The relational model

- Proposed by E. F. Codd in 1970
- Available in commercial DBMS in 1981
- **Relation** as mathematical foundation
- **Table** as a simple, intuitive, and natural structure to represent relations

Mathematical relation

- D_1, D_2, \dots, D_n (n -not necessarily distinct- sets of values)
- The **cartesian product** $D_1 \times D_2 \times \dots \times D_n$ is the set of all ordered n-tuples (d_1, d_2, \dots, d_n) such that $d_1 \in D_1, d_2 \in D_2, \dots, d_n \in D_n$
- A mathematical relation on D_1, D_2, \dots, D_n is a subset of the cartesian product $D_1 \times D_2 \times \dots \times D_n$

Mathematical relation

- D_1, D_2, \dots, D_n are the **domains** of the relation
- n is the **degree** of the relation
- The number of n -tuples is the **cardinality** of the relation (in the practice, it is always finite)

Example

- $D_1 = \{a, b\}$ $D_2 = \{1, 2, 3\}$

a	1
a	2
a	3
b	1
b	2
b	3

- Cartesian product:

$$D_1 \times D_2 =$$

$$\{(a, 1), (a, 2), (a, 3), (b, 1), (b, 2), (b, 3)\}$$

- A relation

$$r \subseteq D_1 \times D_2 = \{(a, 1), (a, 3), (b, 2), (b, 3)\}$$

a	1
a	3
b	2
b	3

Mathematical relation

- The structure of a relation is *positional*
- This means that the order used for specifying tuples is important for correctly interpret the meaning of the relation (especially when integer values are used)

Mathematical relation

movie \subseteq string x string x string x integer

1375666	Inception	2010	148
0816692	Interstellar	2014	169
3460252	The Hateful Eight	2015	167

Relations in the relational model

- In order to exploit relations as non-positional structures, we associate a unique name (**attribute**) with each domain to describe the role of that domain in the relation
- In the table representation, attributes are used as column headings

id	official_title	year	length
1375666	Inception	2010	148
0816692	Interstellar	2014	169
3460252	The Hateful Eight	2015	167

Formalization

- Call X a set of attributes
- In a relation, there is a correspondence between the attributes and the corresponding domains:
 $dom: X \rightarrow D$
- For each attribute $A \in X$, we have an associated domain $dom(A) \in D$
- A **tuple** t on X is a function which associates a value from the domain $dom(A)$ with each $A \in X$
- A **relation** on X is a set of tuples on X
- $t[A]$ denotes the value of tuple t on the attribute A

Definition of a relational database

- **Relation schema $R(X)$**

A name (of the relation) R with a set of attributes $X = \{A_1, \dots, A_n\}$

- **Database schema $R = \{R_1(X_1), \dots, R_n(X_n)\}$**

A set of relation schemas with different names (i.e., each relation has a unique name in the database)

Definition of a relational database

- **Relation instance** on a schema $R(X)$:
- A set of r tuples on X
- **Database instance** on a schema $R = \{R_1(X_1), \dots, R_n(X_n)\}$:
- A set of relations $r = \{r_1, \dots, r_n\}$ (with r_i relation on R_i)

Example – database schema

movie

id	official_title	year	length
----	----------------	------	--------

movie	person	role	character	movie_person (crew)
-------	--------	------	-----------	---------------------

person

id	first_name	last_name	birth_date
----	------------	-----------	------------

Example – database instance

movie

1375666	Inception	2010	148
0816692	Interstellar	2014	169
3460252	The Hateful Eight	2015	167

movie_person (crew)

1375666	0362766	actor	Eames
0816692	0634240	director	
0816692	0004266	actor	Brand

person

0634240	Christopher Johnathan James Nolan		30/07/1970
0362766	Edward Thomas Hardy		15/09/1977
0004266	Anne Jacqueline Hathaway		12/11/1982

Relational database

- A relational database is composed by a collection of relations with attributes represented as tables:
 - Each relation has a unique name in the database
 - Each column has associated a distinct **attribute name** A_k ; each attribute A_k has a domain D_k of possible values
 - Each row of the table is a tuple of values (d_1, d_2, \dots, d_n) each of them belonging to the domain D_k of the corresponding attribute A_k

Value-based structure

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

Example

movie

id	official_title	year	length
1375666	Inception	2010	148
0816692	Interstellar	2014	169
3460252	The Hateful Eight	2015	167

movie_person (crew)

movie	person	role	character
1375666	0362766	actor	Eames
0816692	0634240	director	
0816692	0004266	actor	Brand

person

id	first_name	last_name	birth_date
0634240	Christopher Johnathan James	Nolan	30/07/1970
0362766	Edward Thomas	Hardy	15/09/1977
0004266	Anne Jacqueline	Hathaway	12/11/1982

Incomplete information

- A relation represents the knowledge acquired on the UoD of interest
- Some aspects of the UoD could be unknown
- The relational model imposes a rigid structure to the data:
 - Information is represented by means of tuples
 - Tuples have to conform to relation schemas

Incomplete information: motivations

- A person has a birth date and a death date, but:
 - The death date of Anne Hathaway does not exist
 - The birth date of Alfred Hitchcock exists, but it is unknown to us
 - For Heath Ledger, we do not know if the death date exists or not

id	first_name	last_name	birth_date	death_date
0004266	Anne Jacqueline	Hathaway	12/11/1982	
0000033	Alfred Joseph	Hitchcock		29/04/1980
0005132	Heath Andrew	Ledger	04/04/1979	

The NULL value

- In the relational model, the **NULL value** is defined to denote incomplete information
- NULL is a special value (not a value of the domain) which denotes the absence of a domain value
- It is possible to put a restriction (i.e., a constraint) on the opportunity to have null values in the tuples of a relation

The NULL value semantics

- A NULL value in an attribute can have (at least) three different meanings:
 - *Non-existent value* (e.g., death date of Hathaway)
 - *Unknown value* (e.g., birth date of Hitchcock)
 - *No-information value* (e.g., death date of Ledger)
- The DBMS adopts the **no-information value** semantics

The NULL value semantics

- A NULL value in an attribute can have (at least) three different meanings:
 - *Non-existent value* (e.g., death date of Hathaway)
 - *Unknown value* (e.g., birth date of Hitchcock)
 - *No-information value* (e.g., death date of Ledger)

id	first_name	last_name	birth_date	death_date
0004266	Anne Jacqueline	Hathaway	12/11/1982	NULL
0000033	Alfred Joseph	Hitchcock	NULL	29/04/1980
0005132	Heath Andrew	Ledger	04/04/1979	NULL

A meaningless database instance

movie

id	official_title	year	length
1375666	Inception	2010	148
1375666	Inception: The Cobol Job	2010	-15
0816692	Interstellar	2014	169

movie	person	role	character
1375666	0362766	actor	Eames
0816692	0000190	actor	Cooper
0816692	0004266	actor	Brand

movie_person (crew)

person

id	first_name	last_name	birth_date
0362766	Edward Thomas	Hardy	15/09/1977
0004266	Anne Jacqueline		12/11/1982

Problems

- Movies must have different identifier values
- The movie crew must be associated with an existing person
- The movie length must be a positive number
- The person names (first and last) must be non-null values

Integrity constraints

- An integrity constraint is a property that must be satisfied by all the meaningful instances of a database
- It can be seen as a **predicate** which is evaluated **TRUE** or **FALSE** for each instance of the database

Example

- First and last name of a person cannot be NULL
- In a movie, $\text{length} > 0$

Integrity constraints

- They correspond to properties in the UoD to be described in the database
- They are defined at the schema level and they apply to all the instances of the schema
 - We consider correct (i.e., valid) the instances that satisfy the constraints
- They are important to ensure data quality
- They are defined during the database definition

Unique identification of tuples

id	official_title	year	length
0331570	Moby Dick	2000	22
0049513	Moby Dick	1956	116
0816692	Interstellar	2014	169
3460252	The Hateful Eight	2015	167

- The *movie id* uniquely identifies a movie
 - there is no pair of tuples with same value of id
- The pair (*official_title*, *year*) also provides a unique identifier of a movie
(as well as the pair *official_title*, *length*)

Keys (integrity constraints)

- A set of attributes that uniquely identifies tuples in a relation
- A set K of attributes is a **superkey** for a relation R if R does not contain two distinct tuples t_1 e t_2 with $t_1[K] = t_2[K]$
(unique identification constraint)
- K is a **key** for R if it is a minimal superkey for R (in other words, no other superkey exists that is contained in K as proper subset)
(minimality constraint)

Example

id	official_title	year	length
0331570	Moby Dick	2000	22
0049513	Moby Dick	1956	116
0816692	Interstellar	2014	169
3460252	The Hateful Eight	2015	167

- *Id* is a key:
 - It is a superkey
 - It contains a single attribute, so it is minimal
- The pair (*official_title*, *year*) is another key

Existence of keys

- Relations are sets of tuples, therefore each relation is composed by distinct tuples
 - This means that the whole set of attributes of a tuple is a superkey
- The whole set of attributes:
 - Is either a key
 - Or it contains a (smaller) superkey
 - This line of reasoning can be repeated until no smaller superkeys are identified in the set of considered attributes

Keys and null values

- With nulls, keys do not work well
 - They do not guarantee unique identification
 - They do not allow to establish correspondences between tuples in different relations

id	official_title	year	length
0331570	Moby Dick	2000	NULL
0049513	Moby Dick	NULL	116
0816692	Interstellar	2014	169
NULL	The Hateful Eight	2015	167

- How can we access the 4th tuple?
- Are the 1st and the 2nd tuples the same?

Primary key

- The presence of null values within keys must be limited
- Practical solution: for each relation we select a **primary key** on which null values are not allowed (**entity integrity constraint**)
- Notation: attributes are underlined
- References between relations are implemented through primary keys

Primary keys

- In most cases, we have reasonable primary keys (e.g., unique descriptors)
- In other case, we do not
 - Then, we introduce new attributes with the role of “identifier codes”
- Note that the notion of «natural code» has been introduced with this goal (usually before the use of databases): unique identification of objects
 - This is the case of the id attribute of movies

Referential integrity constraint

- Tuples in different relations are correlated by means of values on primary keys
- Referential integrity constraints are defined in order to guarantee that the values refer to actual values in the referenced relation

Referential integrity

- A **referential integrity** constraint (“**foreign key**”) imposes to the values of attributes X of a relation R_1 to appear as values for the primary key of another relation R_2
- A referential integrity constraint exists between the attribute *id* of the relation *movie* and the attribute *movie* of the relation *crew*

Violation of referential integrity

movie

id	official_title	year	length
1375666	Inception	2010	148
0816692	Interstellar	2014	169
3460252	The Hateful Eight	2015	167

?

crew

movie	person	role	character
1375666	0362766	actor	Eames
0816692	0634240	director	
0816692	0004266	actor	Brand
0110912	0000233	actor	Jimmie

