Coding for Data Science and Data Management
Module of Data Management

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

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

Mathematical relation

- 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 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	1
a	3
b	2
b	3

- A relation

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

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 ⊆ 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 $X = \{A_1,\ldots, A_n\}$

- **Database schema R = {R$_1$ (X$_1$),..., 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
  $R = \{R_1 (X_1),..., R_n (X_n)\}$:
- A set of relations $r = \{r_1,..., r_n\}$ (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 $(d_1, d_2, \ldots, d_n)$ 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 domian value
- It is possibile 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 (crew)

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

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, 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
	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

The risks of concurrency

- A relational DMBS is a concurrent, multi-user system
- This means that data (e.g., any single tuple) can be accessed and updated by multple users at the same time
- If data access is not supervised, the database integrity can be violated

A naive example of concurrency (2)

- At t3, the User B updates the balance by depositing € 50. The User B writes the new balance according to the value read at t2. The new balance is € 150
- At t4, the User A updates the balance by depositing € 75. The User A writes the new balance according to the value read at t2. The new balance is € 175
- € 50 are lost!

Transaction concept

- A transaction is an executing program (e.g., a sequence of operations) that forms a logical unit of database processing
- A transaction can include one or more database operations, such as insert, delete, update of database tuples

Transaction management

- Transactions ensure the database integrity also when the following critical issues occur:
  - Failures of various kinds, such as hardware failures and system crashes
  - Concurrent execution of multiple transactions on a given set of data

Transaction outcome

- Consider a transaction *T* containing a list of data manipulation operations *O* over a database
- The execution of *T* implies the execution of all the operations *O*
- The transaction *T* ends with two possible results:
  - **Commit**: all the operations *O* are successfully executed, the database status is updated
  - **Rollback**: an error occurs in the execution of *O*, the database goes back to the status before the execution of *T*

ACID properties of relational DBs

- Transactions preserve data integrity by enforcing ACID properties:
  - **Atomicity**. A transaction should be either be performed in its entirety or not performed at all
  - **Consistency**. If a transaction is completely executed from beginning to end without interference from other transactions, the database moves from one consistent status to another. Consistency in relational databases is also known as **strict consistency**

ACID properties of relational DBs

- Transactions preserve data integrity by enforcing ACID properties:
  - **Isolation**. The execution of a transaction should not be interfered with by any other transactions executing concurrently
  - **Durability**. The changes applied to the database by a committed transaction must persist in the database
    - Changes must not be lost due to any failure