The Relational Model

Fabrice Rossi

CEREMADE
Université Paris Dauphine

2020

Relational model

History

- invented by Edgar F. Codd in 1969-1970
- based on a mathematical model, the relational algebra
- associated relational calculus
- still the dominant model

Based on relations/tables

id	first_name	last_name	gender	film_count
567368	Olivia	Burnette	F	1
758314	Beata	Pozniak	F	1
636385	Joanne	Gordon	F	1
588101	Suzanne	Cox	F	1
683913	Melissa	Kurtz	F	1

Outline

Tables and relations

Schemas and instances

Keys

Integrity constraints

Using a relational database

Outline

Tables and relations

Schemas and instances

Keys

Integrity constraints

Using a relational database

Informal point of view

A relational database

- is a collection of tables
- each table has
  - a name
  - columns and rows
  - a header which gives names to the columns
  - values in a column are all of the same type (such as integer)
  - rows that record information/data

Example

Actors

id	first_name	last_name	gender	film_count
933	Lewis	Abernathy	M	1
2547	Andrew	Adamson	M	1
2700	William	Addy	M	1
2898	Seth (I)	Adkins	M	1
2925	Charles (I)	Adler	M	1

Movies

id	name	year	rank
192017	Little Mermaid, The	1989	7.30
300229	Shrek	2001	8.10
306032	Snatch.	2000	7.90
333856	Titanic	1997	6.90

Roles

actor_id	movie_id	role
933	333856	Lewis Bodine
2547	300229	Duloc Mascot
2700	306032	Tyrone
2898	333856	Slovakian three-year-old boy
2925	192017	Additional Voices

IMDB database

- 3 tables
  - Actors
  - Movies
  - Roles
- facts about entities
  - actor first name, last name, etc.
  - movie title, year of release, etc.
- relationship between entities
  - an actor played a role in a movie

Formal point of view

Definitions

- a domain is a set of values
- a relation is a subset of the Cartesian product of n domains $\prod_{i=1}^{n} D_i$

$$R \subset \prod_{i=1}^{n} D_i = \{(t_1, \ldots, t_n) \mid t_1 \in D_1, \ldots, t_n \in D_n\}$$

- each dimension of a relation is an attribute, identified by a unique name (for the relation)
- a tuple is an element of a relation
- a database is a collection of relations

Vocabulary and notations

Informal ↔ formal

informal	**formal**
table	relation
column	attribute
row	tuple
type	domain

Notations

- if t is a tuple and A an attribute name, $t.A$ (or $t[A]$) denotes the value of the attribute for in the tuple
- if R is a relation and A an attribute name, $R.A$ denotes the attribute with name A in R

Examples

Domains

- numerical values $\mathbb{R}$
- non negative integers $\mathbb{N}$
- short texts (strings)
- *ad hoc* domains such as dates
- arbitrary finite sets such $Gender = \{F, M\}$

Cartesian product

$$\underbrace{\mathbb{N}}_{\textit{age}} \times \underbrace{\mathbb{R}^{+}}_{\textit{weight}} \times \underbrace{\mathbb{R}^{+}}_{\textit{height}}$$

Relations?

age	weight	height
25	50.00	1.60
30	52.30	1.58
35	64.00	1.70

age	weight	height
30	65.00	1.70
35	95.00	1.88
35	95.00	1.88

age	height	weight
30	1.70	65.00
35	1.88	95.00
40	1.90	87.30

age	weight	height
-30	65.00	1.70
-35	95.00	1.88
-40	87.30	1.90

Outline

Tables and relations

Schemas and instances

Keys

Integrity constraints

Using a relational database

Schema of a relation

Logical schema

- specifying the possible values of a relation (i.e. the acceptable subsets)
- naming things

Definitions

A relation schema

- specifies the name of the relation, e.g. X
- specifies the (unique) names and the domains of its attributes, e.g. $A : \mathbb{N}$, $B : \mathbb{R}$
- is denoted $Name(Attribute1 : domain1, Attribute2 : domain2, \ldots)$, e.g. $X(A : \mathbb{N}, B : \mathbb{R})$

A relation is an instance of a relation schema.

Example

Cartesian product

$$\underbrace{\mathbb{N}}_{\textit{age}} \times \underbrace{\mathbb{R}^+}_{\textit{weight}} \times \underbrace{\mathbb{R}^+}_{\textit{height}}$$

Schema

$$\textit{Person}(\textit{age} : \mathbb{N}, \textit{weight} : \mathbb{R}, \textit{height} : \mathbb{R})$$

Remark

- frequently the domains are obvious or implicit
- we can drop them, e.g. $\textit{Person}(\textit{age}, \textit{weight}, \textit{height})$

Examples

IMDB database

Actors

id	first_name	last_name	gender	film_count
631024	Kim	Genell	F	1
620283	Carrie	Fisher	F	1
567368	Olivia	Burnette	F	1
623010	Gloria	Foster	F	1
737979	Pat	Nixon	F	1

Movies

id	name	year	rank
167324	JFK	1991	7.80
207992	Matrix, The	1999	8.50
257264	Planes, Trains & Automobiles	1987	7.20
313459	Star Wars	1977	8.80

Roles

actor_id	movie_id	role
567368	257264	Marti
620283	313459	Princess Leia Organa
623010	207992	Oracle
631024	257264	Receptionist
737979	167324	Herself (with Richard)

Examples

IMDB database

Actors

id	first_name	last_name	gender	film_count
631024	Kim	Genell	F	1
620283	Carrie	Fisher	F	1
567368	Olivia	Burnette	F	1
623010	Gloria	Foster	F	1
737979	Pat	Nixon	F	1

Movies

id	name	year	rank
167324	JFK	1991	7.80
207992	Matrix, The	1999	8.50
257264	Planes, Trains & Automobiles	1987	7.20
313459	Star Wars	1977	8.80

Roles

actor_id	movie_id	role
567368	257264	Marti
620283	313459	Princess Leia Organa
623010	207992	Oracle
631024	257264	Receptionist
737979	167324	Herself (with Richard)

$$\begin{aligned}&\mathit{Actors}(\mathit{id} : \mathbb{N}^+, \mathit{first\_name} : \mathit{string},\\ &\quad \mathit{last\_name} : \mathit{string},\\ &\quad \mathit{gender} : \{F, M\}, \mathit{film\_count} : \mathbb{N}^+)\end{aligned}$$

Examples

IMDB database

Actors

id	first_name	last_name	gender	film_count
631024	Kim	Genell	F	1
620283	Carrie	Fisher	F	1
567368	Olivia	Burnette	F	1
623010	Gloria	Foster	F	1
737979	Pat	Nixon	F	1

$$\mathit{Actors}(\mathit{id} : \mathbb{N}^+, \mathit{first\_name} : \mathit{string}, \mathit{last\_name} : \mathit{string}, \mathit{gender} : \{F, M\}, \mathit{film\_count} : \mathbb{N}^+)$$

Movies

id	name	year	rank
167324	JFK	1991	7.80
207992	Matrix, The	1999	8.50
257264	Planes, Trains & Automobiles	1987	7.20
313459	Star Wars	1977	8.80

$$\mathit{Movies}(\mathit{id} : \mathbb{N}^+, \mathit{name} : \mathit{string}, \mathit{year} : \mathbb{N}^+, \mathit{rank} : \mathbb{R}^+)$$

Roles

actor_id	movie_id	role
567368	257264	Marti
620283	313459	Princess Leia Organa
623010	207992	Oracle
631024	257264	Receptionist
737979	167324	Herself (with Richard)

Examples

IMDB database

Actors

id	first_name	last_name	gender	film_count
631024	Kim	Genell	F	1
620283	Carrie	Fisher	F	1
567368	Olivia	Burnette	F	1
623010	Gloria	Foster	F	1
737979	Pat	Nixon	F	1

$$\mathit{Actors}(\mathit{id} : \mathbb{N}^+, \mathit{first\_name} : \mathit{string}, \mathit{last\_name} : \mathit{string}, \mathit{gender} : \{F, M\}, \mathit{film\_count} : \mathbb{N}^+)$$

Movies

id	name	year	rank
167324	JFK	1991	7.80
207992	Matrix, The	1999	8.50
257264	Planes, Trains & Automobiles	1987	7.20
313459	Star Wars	1977	8.80

$$\mathit{Movies}(\mathit{id} : \mathbb{N}^+, \mathit{name} : \mathit{string}, \mathit{year} : \mathbb{N}^+, \mathit{rank} : \mathbb{R}^+)$$

Roles

actor_id	movie_id	role
567368	257264	Marti
620283	313459	Princess Leia Organa
623010	207992	Oracle
631024	257264	Receptionist
737979	167324	Herself (with Richard)

$$\mathit{Roles}(\mathit{actor\_id} : \mathbb{N}^+, \mathit{movie\_id} : \mathbb{N}^+, \mathit{role} : \mathit{string})$$

Database schema

Definitions

- a database schema is a collection of relation schemas
- relation names must be unique
- a instance of a database schema is database whose relations are instances of the relation schemas

Conventions

- relation schemas in a database schema can share attribute names
- to avoid confusion, use common names only for identical domains

Outline

Tables and relations

Schemas and instances

Keys

Integrity constraints

Using a relational database

Unicity

Relations are sets

- no order
- each tuple/row is unique

Superkeys and keys

- a *superkey*
  - any subset of the attributes such that no tuples of the relation share exactly the same values for these attributes
  - default superkey: all the attributes
- a *key*: a minimal superkey (removing attributes removes it superkey status)

Example

France administrative structure

Region

region_id	region
27	Bourgogne-Franche-Comté
4	La Réunion
32	Hauts-de-France
11	Île-de-France
93	Provence-Alpes-Côte d'Azur

Region(region_id, region)

Department

region_id	department_id	department
27	25	Doubs
84	3	Allier
75	40	Landes
44	88	Vosges
84	26	Drôme

Department(region_id, department_id, department)

Superkeys

- (region_id, region)
- (region_id)
- (region)

Superkeys

- (region_id, department_id, department)
- (region_id, department_id)
- (region_id, department)
- (department_id, department)
- (department_id)
- (department)

Primary keys

Primary key

- several keys: *candidate* keys
- one of the is *the primary key*
- other are *alternative* keys

Region

region_id	region
4	La Réunion
11	Île-de-France
84	Auvergne-Rhône-Alpes
75	Nouvelle-Aquitaine
93	Provence-Alpes-Côte d’Azur

Department

region_id	department_id	department
93	83	Var
75	33	Gironde
11	91	Essonne
84	3	Allier
44	57	Moselle

Primary key

- region_id
- Region(region_id, region)

Primary key

- department_id
- Department(region_id, department_id, department)

Missing values

Null value

- relaxing the relation definition
- an attribute value can be either a value of its domain or the special value **`NULL`**
- useful for
  - missing values (not recorded for instance)
  - non applicable attribute
- should not be accepted for all attributes, e.g. primary keys

Foreign keys

Foreign keys

- a set of attributes FK in a relation R_1 is a *foreign key* of R_1 if
  1. a candidate key in a relation R_2 has exactly the same domains as the ones of the attributes in FK
  2. values on FK in a tuple in r_1 are either *NULL* or occur in a tuple in r_2
- in simple terms: a foreign key links R_1 to R_2

Example: region_id

Department

region_id	department_id	department
1	971	Guadeloupe
24	41	Loir-et-Cher
75	23	Creuse
53	29	Finistère
84	42	Loire

Region

region_id	region
84	Auvergne-Rhône-Alpes
24	Centre-Val de Loire
44	Grand-Est
28	Normandie
4	La Réunion

Example

IMDB database

- Actors(id, first_name, last_name, gender, film_count)
- Movies(id, name, year, rank)
- Roles(#actor_id,#movie_id,role)
  - Roles.actor_id is a foreign key to Actors.id
  - Roles.movie_id is a foreign key to Movies.id
  - (actor_id, movie_id) is the primary key of Roles

Outline

Tables and relations

Schemas and instances

Keys

Integrity constraints

Using a relational database

Principle

Legal instances

- many subsets of a Cartesian product are not acceptable as relations
- integrity constraints *specify* legal/correct instances of a database

Implicity constraints

- some constraints are already implied by the definition of relations
- domain integrity: attribute values must be **`NULL`** or element of the associated domain
- unicity: tuples are unique within a relation

Key integrity

Primary key

- values of (attributes of) the primary key cannot by **`NULL`**
- values of (attributes of) the primary key are unique in a relation

Referential integrity

- foreign keys are valid
- in other words: foreign keys can be either *NULL* or must refer to an existing tuple

Other constraints

Tuple constraints

- additional constraints can be specified at the tuple level
- this insures consistency in the tuple, such as
  - a start date must be earlier than an end date
  - one cannot be married below a certain age
  - etc.

Global constraints

- additional constraints at the relation or database level
- this insures global consistency, such as
  - a book cannot be borrowed by two different persons at the same time
  - the total working hours of a employee cannot exceed the legal limit

Design

Numerous possibilities

- schemas are not created equal
  - avoid redundancy
  - leverage constraints
  - numerous other considerations
- in addition
  - how to select the primary key among the candidate keys?
  - what are the natural constraints?
  - etc.

To be discussed in later courses

Outline

Tables and relations

Schemas and instances

Keys

Integrity constraints

Using a relational database

Using a database

The relational model so far

- database specification via schemas and constraints
- database instances (i.e. content)

How to use a database?

- questions about the data expressed as new data/information
- a query language
    - input: a database and a query
    - output: a relation

Examples

How many distinct roles are there in each movie?

name	year	rank	nb_role
JFK	1991	7.80	230
Titanic	1997	6.90	130
Star Wars	1977	8.80	104
Apollo 13	1995	7.50	97
Ocean’s Eleven	2001	7.50	84
Vanilla Sky	2001	6.90	84
Mystic River	2003	8.10	70
Snatch.	2000	7.90	64
UHF	1989	6.60	64
Fight Club	1999	8.50	63

Average rank of movies per year?

year	avg_rank
1972	9.00
1977	8.80
1978	7.50
1984	5.80
1986	8.20
1987	7.20
1989	6.95
1991	7.80
1992	7.90
1994	8.85

Query languages

Two families

- procedural languages
  - classical approaches in general purpose languages (C, C++, Python, Java, R, Javascript, etc.)
  - describe a computation via a series of steps
  - steps build the result progressively
  - relational algebra
- declarative languages
  - far less common (functional languages, XQuery/XSLT, Prolog, etc.)
  - describe properties of the result but not the way to get it
  - relational calculus (outside the scope of this course)

SQL

- dominant query language for DBMS
- mostly declarative, with some procedural aspects

Relational algebra

Principle

- a set of operations on relations
- extension of standard set oriented operations (union, intersection, etc.):
  - selection (subsetting) and projection (dropping attributes)
  - renaming (of attributes)
  - numerous derived and extended operations
- used to transform and combine relations
- query as the evaluation of a formula

Availability

- many data oriented framework offer the relational operations (in disguise)
- e.g. R with dplyr and Python with Pandas

Fundamental operations

Selection

- row oriented subsetting operation
- find tuples that fulfill some conditions
- $\sigma_{condition}(relation)$
    - *relation* is a relation (!)
    - *condition* is a predicate defined on the attribute of the relation
        - a function that maps a tuple to a truth value
        - a logical formula
    - the result is the subset of the tuples for which the predicate is true

$$\sigma_{condition}(relation) = \{t \in relation \mid condition(t) = True\}$$

Examples

Movies ranked above 8.5

$$\sigma_{rank \geq 8.5}(Movies)$$

id	name	year	rank
112290	Fight Club	1999	8.50
130128	Godfather, The	1972	9.00
207992	Matrix, The	1999	8.50
210511	Memento	2000	8.70
267038	Pulp Fiction	1994	8.70
297838	Shawshank Redemption, The	1994	9.00
313459	Star Wars	1977	8.80

Actresses with a least 2 roles

$$\sigma_{(gender=F)\wedge(film\_count\geq 2)}(Actors)$$

id	first_name	last_name	gender	film_count
602370	Cameron	Diaz	F	2
623535	Vivica A.	Fox	F	2
635153	Jenette	Goldstein	F	2
646020	Daryl	Hannah	F	2
673070	Linda	Kaye	F	2
697645	Lucy	Liu	F	2
715702	Edie	McClurg	F	2
729933	Carrie-Anne	Moss	F	2
812916	Uma	Thurman	F	3
820312	Venessia	Valentino	F	2

Notations

- logical and: $\wedge$
- logical or: $\vee$
- negation: $\neg$

Fundamental operations

Projection

- column oriented subsetting operation
- keep only a subset of the attributes
- $\Pi_{attributes}(relation)$
  - *relation* is a relation
  - *attributes* is a subset of the attributes of *relation*
  - the result is a projection of *relation* whose tuples have kept only the attributes in specified subset

Example

Titles of the movies from 2000

$$\Pi_{name}(\sigma_{year=2000}(Movies))$$

name
Hollow Man
Memento
O Brother, Where Art Thou?
Snatch.

Fundamental operations

Renaming

- simple attribute renaming
- useful
  - to define derived operations
  - to fulfill some compatibility requirement for set oriented operations
- $\rho_{N_1 \leftarrow O_1, \ldots, N_k \leftarrow O_k}(\mathit{relation})$
  - *relation* is a relation
  - $\{O_1, \ldots, O_k\}$ are attributes (names) from *relation*
  - $\{N_1, \ldots, N_k\}$ are new attribute names
  - the result is a new relation in which each attribute whose name is in $\{O_1, \ldots, O_k\}$ has been renamed to the corresponding name in $\{N_1, \ldots, N_k\}$
  - other attributes are kept intact

Example

Titles of the movies from 2000

$$\rho_{title \leftarrow name}(\Pi_{name}(\sigma_{year=2000}(Movies)))$$

title
Hollow Man
Memento
O Brother, Where Art Thou?
Snatch.

Fundamental operations

Operations from set theory

- union $\bigcup$: restricted to relations that share exactly the same attributes (hence the need for renaming)
- set difference $r_1 - r_2$
  - same restrictions as $\bigcup$
  - $r_1 - r_2$ is the set of tuples from r_1 that are not in r_2
- cartesian product $r_1 \times r_2$
  - $r_1 \times r_2$ is the set of concatenation of tuples from r_1 and r_2
  - all the pairs are used to generate the concatenated tuples
  - attributes with common names are handled via prefixing them with the name of the relations
  - generally useless with selection

Example

Cartesian product

Movies × *Roles*

id	name	year	rank	actor_id	movie_id	role
10920	Aliens	1986	8.20	933	333856	Lewis Bodine
10920	Aliens	1986	8.20	2547	300229	Duloc Mascot
10920	Aliens	1986	8.20	2700	306032	Tyrone
10920	Aliens	1986	8.20	2898	333856	Slovakian three-year-old boy
10920	Aliens	1986	8.20	2925	192017	Additional Voices
10920	Aliens	1986	8.20	3226	238072	Virgil Malloy
10920	Aliens	1986	8.20	4306	194874	Mr. Valentine
10920	Aliens	1986	8.20	4856	194874	Hans
10920	Aliens	1986	8.20	6005	167324	Maitre D'
10920	Aliens	1986	8.20	6990	30959	Shadow Warrior
10920	Aliens	1986	8.20	7124	238072	Police Officer
10920	Aliens	1986	8.20	7817	267038	Dead Floyd Wilson
10920	Aliens	1986	8.20	7979	18979	Anchor
10920	Aliens	1986	8.20	8161	194874	American Businessman #2
10920	Aliens	1986	8.20	8409	238072	French High Roller

Example

Cartesian product with selection

$$\sigma_{id=movie\_id}(Movies \times Roles)$$

id	name	year	rank	actor_id	movie_id	role
10920	Aliens	1986	8.20	16844	10920	Lydecker
10920	Aliens	1986	8.20	36641	10920	Russ Jorden
10920	Aliens	1986	8.20	42278	10920	Cpl. Dwayne Hicks
10920	Aliens	1986	8.20	144260	10920	Doctor
10920	Aliens	1986	8.20	204493	10920	Timmy Jorden
10920	Aliens	1986	8.20	204719	10920	Bishop
10920	Aliens	1986	8.20	213646	10920	Lt. Gorman
10920	Aliens	1986	8.20	240277	10920	Pvt. Spunkmeyer
10920	Aliens	1986	8.20	272557	10920	Power Loader Operator
10920	Aliens	1986	8.20	305705	10920	Sgt. Apone
10920	Aliens	1986	8.20	306790	10920	Van Leuwen
10920	Aliens	1986	8.20	310030	10920	Al Simpson, Colony Officer
10920	Aliens	1986	8.20	366173	10920	Private Hudson
10920	Aliens	1986	8.20	378476	10920	Insurance Man
10920	Aliens	1986	8.20	394516	10920	Carter Burke

Joins

Derived operations

- numerous operations can be defined based on the six fundamental operations
- the most important ones are join operations
  - combination of the cartesian product with specific selections
    - can be seen as a filter on pairs of tuples before the concatenation
    - for instance only tuples that share some common values
  - implement the connections between tuples of the relational model

Joins

Natural join

- assumptions
  - r_i has the attributes R_i
  - $R_1 \cap R_2 = \{A_1, \ldots, A_k\}$ (maybe be empty)
- the natural join of r_1 and r_2 is denoted $r_1 \bowtie r_2$ and is given by

$$r_1 \bowtie r_2 = \Pi_{R_1 \cup R_2}(\sigma_{r_1.A_1=r_2.A_1 \wedge \ldots \wedge r_1.A_k=r_2.A_k}(r_1 \times r_2))$$

- intuitively $r_1 \bowtie r_2$ is obtained by concatenation of tuples that agree on the common attributes
- particularly adapted for matching foreign keys and primary keys!

Example

Roles in Movies

$$\rho_{movie\_id \leftarrow id}(Movies) \bowtie Roles$$

movie_id	name	year	rank	actor_id	role
10920	Aliens	1986	8.20	16844	Lydecker
10920	Aliens	1986	8.20	36641	Russ Jorden
10920	Aliens	1986	8.20	42278	Cpl. Dwayne Hicks
10920	Aliens	1986	8.20	144260	Doctor
10920	Aliens	1986	8.20	204493	Timmy Jorden
10920	Aliens	1986	8.20	204719	Bishop
10920	Aliens	1986	8.20	213646	Lt. Gorman
10920	Aliens	1986	8.20	240277	Pvt. Spunkmeyer
10920	Aliens	1986	8.20	272557	Power Loader Operator
10920	Aliens	1986	8.20	305705	Sgt. Apone
10920	Aliens	1986	8.20	306790	Van Leuwen
10920	Aliens	1986	8.20	310030	Al Simpson, Colony Officer
10920	Aliens	1986	8.20	366173	Private Hudson
10920	Aliens	1986	8.20	378476	Insurance Man
10920	Aliens	1986	8.20	394516	Carter Burke

Example

Roles with Actors in Movies

$$(\rho_{movie\_id \leftarrow id}(Movies) \bowtie Roles) \bowtie \rho_{actor\_id \leftarrow id}(Actors)$$

movie_id	name	year	rank	actor_id	role	first_name	last_name	gender	film_count
10920	Aliens	1986	8.20	16844	Lydecker	William (I)	Armstrong	M	1
10920	Aliens	1986	8.20	36641	Russ Jorden	Jay (I)	Benedict	M	2
10920	Aliens	1986	8.20	42278	Cpl. Dwayne Hicks	Michael	Biehn	M	1
10920	Aliens	1986	8.20	144260	Doctor	Blain	Fairman	M	1
10920	Aliens	1986	8.20	204493	Timmy Jorden	Christopher	Henn	M	1
10920	Aliens	1986	8.20	204719	Bishop	Lance	Henriksen	M	1
10920	Aliens	1986	8.20	213646	Lt. Gorman	William	Hope	M	1
10920	Aliens	1986	8.20	240277	Pvt. Spunkmeyer	Daniel	Kash	M	1
10920	Aliens	1986	8.20	272557	Power Loader Operator	John	Lees	M	1
10920	Aliens	1986	8.20	305705	Sgt. Apone	Al	Matthews	M	1
10920	Aliens	1986	8.20	306790	Van Leuwen	Paul (I)	Maxwell	M	1
10920	Aliens	1986	8.20	310030	Al Simpson, Colony Officer	Mac (I)	McDonald	M	1
10920	Aliens	1986	8.20	366173	Private Hudson	Bill	Paxton	M	3
10920	Aliens	1986	8.20	378476	Insurance Man	Alan	Polonsky	M	1
10920	Aliens	1986	8.20	394516	Carter Burke	Paul	Reiser	M	1
10920	Aliens	1986	8.20	405572	Pvt. Drake	Mark	Rolston	M	2
10920	Aliens	1986	8.20	408215	Pvt. Frost	Ricco	Ross	M	1
10920	Aliens	1986	8.20	452549	Pvt. Wierzbowski	Trevor	Steedman	M	2
10920	Aliens	1986	8.20	474375	Pvt. Crowe	Tip	Tipping	M	1
10920	Aliens	1986	8.20	476429	Alien Warrior	Carl	Toop	M	1

Variations of natural join

Equi-join

- the equi-join extends the natural join by allowing comparison between attributes with different names
- one replaces $\sigma_{r_1.A_1=r_2.A_1\wedge\ldots\wedge r_1.A_k=r_2.A_k}(.)$ by a more general $\sigma_{r_1.A_1=r_2.B_1\wedge\ldots\wedge r_1.A_k=r_2.B_k}(.)$
- useful to avoid renaming

Theta-join

- the theta-join goes one step beyond by allowing any predicate in the selection
- definition

$$r_1 \bowtie_F r_2 = \sigma_F(r_1 \times r_2)$$

where F is any predicate

Missing data and joins

Excluded tuples

- some tuples from r_1 and from r_2 might be excluded from $r_1 \bowtie_F r_2$
- in some situations, it is useful to include this fact in the resulting relation

Examples

- roles in movies after 2000
  - $\mathit{Roles} \bowtie_{\mathit{movie\_id}=\mathit{id}} \sigma_{\mathit{year}\geq 2000}(\mathit{Movies})$
  - exclude roles before 2000
- actors with roles in movies after 2000
  - $\mathit{Actors} \bowtie_{\mathit{id}=\mathit{actors\_id}} (\mathit{Roles} \bowtie_{\mathit{movie\_id}=\mathit{id}} \sigma_{\mathit{year}\geq 2000}(\mathit{Movies}))$
  - excludes actors with no roles in movies after 2000

Extended joins

Outer joins

- principle
  - if $t_1 \in r_1$ is such that there is no $t_2 \in r_2$ with $t_1 t_2 \in r_1 \bowtie_F r_2$, then include $t_1(NULL, \ldots, NULL)$ in the result relation
  - do the same for $t_2 \in r_2$
- outer join $⟗$: allow $t_1(NULL, \ldots, NULL)$ and $(NULL, \ldots, NULL)t_2$
- left outer join $⟕$: allow $t_1(NULL, \ldots, NULL)$
- right outer join $⟖$: allow $(NULL, \ldots, NULL)t_2$

All cases

A

x	y
-3	1
4	2
5	NULL

Missing foreign key

B

y	z
1	a
2	b
3	c

Unreferenced key

All cases

A

x	y
-3	1
4	2
5	NULL

Missing foreign key

B

y	z
1	a
2	b
3	c

Unreferenced key

$A \bowtie B$

x	y	z
-3	1	a
4	2	b

Only full rows

All cases

A

x	y
-3	1
4	2
5	NULL

Missing foreign key

B

y	z
1	a
2	b
3	c

Unreferenced key

$A \bowtie B$

x	y	z
-3	1	a
4	2	b

Only full rows

A ⟗ B

x	y	z
-3	1	a
4	2	b
5	NULL	NULL
NULL	3	c

All combinations

All cases

A

x	y
-3	1
4	2
5	NULL

Missing foreign key

B

y	z
1	a
2	b
3	c

Unreferenced key

A ⋈ B

x	y	z
-3	1	a
4	2	b

Only full rows

A ⟗ B

x	y	z
-3	1	a
4	2	b
5	NULL	NULL
NULL	3	c

All combinations

A ⟕ B

x	y	z
-3	1	a
4	2	b
5	NULL	NULL

All rows from the left relation

A ⟖ B

x	y	z
-3	1	a
4	2	b
NULL	3	c

All rows from the right relation

Example

Movies after 2000

$$Roles \mathbin{⟗}_{movie\_id=id} \sigma_{year \geq 2000}(Movies)$$

actor_id	movie_id	role	name	year	rank
933	333856	Lewis Bodine	NULL	NULL	NULL
2547	300229	Duloc Mascot	Shrek	2001	8.10
2700	306032	Tyrone	Snatch.	2000	7.90
2898	333856	Slovakian three-year-old boy	NULL	NULL	NULL
2925	192017	Additional Voices	NULL	NULL	NULL
3226	238072	Virgil Malloy	Ocean’s Eleven	2001	7.50
4306	194874	Mr. Valentine	Lost in Translation	2003	8.00
4856	194874	Hans	Lost in Translation	2003	8.00
6005	167324	Maitre D’	NULL	NULL	NULL
6990	30959	Shadow Warrior	Batman Begins	2005	0.00

- no movie without role: this would break referential integrity!
- roles without movie: in practice, this is equivalent to a left join

Example

Movies after 2000

$$\mathit{Roles} ⟕_{\mathit{movie\_id}=\mathit{id}} \sigma_{\mathit{year}\geq 2000}(\mathit{Movies})$$

actor_id	movie_id	role	name	year	rank
933	333856	Lewis Bodine	NULL	NULL	NULL
2547	300229	Duloc Mascot	Shrek	2001	8.10
2700	306032	Tyrone	Snatch.	2000	7.90
2898	333856	Slovakian three-year-old boy	NULL	NULL	NULL
2925	192017	Additional Voices	NULL	NULL	NULL
3226	238072	Virgil Malloy	Ocean’s Eleven	2001	7.50
4306	194874	Mr. Valentine	Lost in Translation	2003	8.00
4856	194874	Hans	Lost in Translation	2003	8.00
6005	167324	Maitre D’	NULL	NULL	NULL
6990	30959	Shadow Warrior	Batman Begins	2005	0.00

- no movie without role: this would break referential integrity!
- roles without movie: in practice, this is equivalent to a left join

Example

Movies after 2000

$$Roles ⟖_{movie\_id=id} \sigma_{year \geq 2000}(Movies)$$

actor_id	movie_id	role	name	year	rank
2547	300229	Duloc Mascot	Shrek	2001	8.10
2700	306032	Tyrone	Snatch.	2000	7.90
3226	238072	Virgil Malloy	Ocean’s Eleven	2001	7.50
4306	194874	Mr. Valentine	Lost in Translation	2003	8.00
4856	194874	Hans	Lost in Translation	2003	8.00
6990	30959	Shadow Warrior	Batman Begins	2005	0.00
7124	238072	Police Officer	Ocean’s Eleven	2001	7.50
8161	194874	American Businessman #2	Lost in Translation	2003	8.00
8409	238072	French High Roller	Ocean’s Eleven	2001	7.50
8412	238072	Guard	Ocean’s Eleven	2001	7.50

- no movie without role: this would break referential integrity!
- roles without movie: in practice, this is equivalent to a left join

Example

Actors with roles in movies after 2000

$$Actors \mathbin{⟕}_{id=actors\_id} \left(Roles \bowtie_{movie\_id=id} \sigma_{year \geq 2000}(Movies)\right)$$

id	first_name	last_name	gender	film_count	movie_id	role	name	year	rank
933	Lewis	Abernathy	M	1	NULL	NULL	NULL	NULL	NULL
2547	Andrew	Adamson	M	1	300229	Duloc Mascot	Shrek	2001	8.10
2700	William	Addy	M	1	306032	Tyrone	Snatch.	2000	7.90
2898	Seth (I)	Adkins	M	1	NULL	NULL	NULL	NULL	NULL
2925	Charles (I)	Adler	M	1	NULL	NULL	NULL	NULL	NULL
3226	Casey	Affleck	M	1	238072	Virgil Malloy	Ocean's Eleven	2001	7.50
4306	Shigekazu	Aida	M	1	194874	Mr. Valentine	Lost in Translation	2003	8.00
4856	Julliet	Akinyi	M	1	194874	Hans	Lost in Translation	2003	8.00
6005	Henri	Alciatore	M	1	NULL	NULL	NULL	NULL	NULL
6990	Dean	Alexandrou	M	1	30959	Shadow Warrior	Batman Begins	2005	0.00

- no movie without role: this would break referential integrity!
- actors without role: in practice, this is equivalent to a left join

Example

Actors with roles in movies after 2000

$$\mathit{Actors} ⟕_{\mathit{id}=\mathit{actors\_id}} (\mathit{Roles} \bowtie_{\mathit{movie\_id}=\mathit{id}} \sigma_{\mathit{year}\geq 2000}(\mathit{Movies}))$$

id	first_name	last_name	gender	film_count	movie_id	role	name	year	rank
933	Lewis	Abernathy	M	1	NULL	NULL	NULL	NULL	NULL
2547	Andrew	Adamson	M	1	300229	Duloc Mascot	Shrek	2001	8.10
2700	William	Addy	M	1	306032	Tyrone	Snatch.	2000	7.90
2898	Seth (I)	Adkins	M	1	NULL	NULL	NULL	NULL	NULL
2925	Charles (I)	Adler	M	1	NULL	NULL	NULL	NULL	NULL
3226	Casey	Affleck	M	1	238072	Virgil Malloy	Ocean's Eleven	2001	7.50
4306	Shigekazu	Aida	M	1	194874	Mr. Valentine	Lost in Translation	2003	8.00
4856	Julliet	Akinyi	M	1	194874	Hans	Lost in Translation	2003	8.00
6005	Henri	Alciatore	M	1	NULL	NULL	NULL	NULL	NULL
6990	Dean	Alexandrou	M	1	30959	Shadow Warrior	Batman Begins	2005	0.00

- no movie without role: this would break referential integrity!
- actors without role: in practice, this is equivalent to a left join

Example

Actors with roles in movies after 2000

$$\mathit{Actors} \mathbin{⟕}_{\mathit{id}=\mathit{actors\_id}} (\mathit{Roles} \bowtie_{\mathit{movie\_id}=\mathit{id}} \sigma_{\mathit{year}\geq 2000}(\mathit{Movies}))$$

id	first_name	last_name	gender	film_count	movie_id	role	name	year	rank
2547	Andrew	Adamson	M	1	300229	Duloc Mascot	Shrek	2001	8.10
2700	William	Addy	M	1	306032	Tyrone	Snatch.	2000	7.90
3226	Casey	Affleck	M	1	238072	Virgil Malloy	Ocean's Eleven	2001	7.50
4306	Shigekazu	Aida	M	1	194874	Mr. Valentine	Lost in Translation	2003	8.00
4856	Julliet	Akinyi	M	1	194874	Hans	Lost in Translation	2003	8.00
6990	Dean	Alexandrou	M	1	30959	Shadow Warrior	Batman Begins	2005	0.00
7124	Jim	Alfonso	M	1	238072	Police Officer	Ocean's Eleven	2001	7.50
8161	Richard (XV)	Allen	M	1	194874	American Businessman #2	Lost in Translation	2003	8.00
8409	Anthony	Allison	M	1	238072	French High Roller	Ocean's Eleven	2001	7.50
8412	Bill (I)	Allison	M	1	238072	Guard	Ocean's Eleven	2001	7.50

- no movie without role: this would break referential integrity!
- actors without role: in practice, this is equivalent to a left join

Extensions

Generalized projections

- expression in projections
- calculated columns/attributes
- $\Pi_{F_1,\ldots,F_k}(\mathit{relation})$
  - the F_i are expressions formed over attribute names
  - they serve as attribute names in the resulting relation which is the set

$$\Pi_{F_1,\ldots,F_k}(\mathit{relation}) = \{(F_1(t),\ldots,F_k(t)) \mid t \in \mathit{relation}\}$$

Example

C

x	y
3	1
4	2
5	3

$\Pi_{x+y,x-y}(C)$

x + y	x - y
4	2
6	2
8	2

Example

Loan relation

loan_id	account_id	date	amount	duration	status
4959	2	1994-01-05	80952	24	A
4961	19	1996-04-29	30276	12	B
4962	25	1997-12-08	30276	12	A
4967	37	1998-10-14	318480	60	D
4968	38	1998-04-19	110736	48	C
4973	67	1996-05-02	165960	24	A
4986	97	1997-08-10	102876	12	A
4988	103	1997-12-06	265320	36	D
4989	105	1998-12-05	352704	48	C
4990	110	1997-09-08	162576	36	C

Example

$$\Pi_{loan\_id,payments=amount/duration}(Loan)$$

loan_id	payments
4959	3373.00
4961	2523.00
4962	2523.00
4967	5308.00
4968	2307.00
4973	6915.00
4986	8573.00
4988	7370.00
4989	7348.00
4990	4516.00

Example

$$\Pi_{loan\_id,payments=amount/duration}(Loan)$$

loan_id	payments
4959	3373.00
4961	2523.00
4962	2523.00
4967	5308.00
4968	2307.00
4973	6915.00
4986	8573.00
4988	7370.00
4989	7348.00
4990	4516.00

Why do we keep loan_id in the projection?

Extensions

Aggregation

- computes an aggregate over a (multi)set of values
- examples
  - count
  - sum
  - average
  - minimum and maximum
  - etc.
- for instance $\mathcal{G}_{avg(X)}$ (*relation*) computes a relation with only one attribute $avg(X)$ which contains the average value of all the values of X in *relation* (including multiple time the same value)

Example

D

x	y
A	1
B	2
C	4
B	4

$\mathcal{G}_{sum(y)}\,(D)$

sum(y)
11

Examples

$$\mathcal{G}_{min(rank),avg(rank),max(rank)}\,(Movies)$$

min(rank)	avg(rank)	max(rank)
0.00	7.36	9.00

$$\mathcal{G}_{avg(payments)}\left(\Pi_{loan\_id,payments=amount/duration}(Loan)\right)$$

avg(payments)
4190.66

Extensions

Grouped Aggregation

- group tuples based on the values of some of their attributes
- compute an aggregate over the other attributes, group by group
- ${}_{A_1,\ldots,A_k}\mathcal{G}_{agg(B)}\ (\mathit{relation})$
  - computes all subsets (groups) of *relation* that share the same values of attributes $A_1, \ldots, A_k$
  - computes $agg(B)$ for each subset
  - the resulting relation has attributes $A_1, \ldots, A_k, agg(B)$
  - it consists in one tuple per group with values given by the grouping attributes $A_1, \ldots, A_k$ and the aggregate computed on the group

Example

D

x	y
A	1
B	2
C	4
B	4

$\mathcal{G}_{sum(y)}\,(D)$

sum(y)
11

Example

D

x	y
A	1
B	2
C	4
B	4

$\mathcal{G}_{sum(y)}(D)$

sum(y)
11

${}_{x}\mathcal{G}_{sum(y)}(D)$

x	sum(y)
A	1
B	6
C	4

Examples

How many distinct roles are there in each movie?

name	year	rank	nb_role
JFK	1991	7.80	230
Titanic	1997	6.90	130
Star Wars	1977	8.80	104
Apollo 13	1995	7.50	97
Ocean’s Eleven	2001	7.50	84
Vanilla Sky	2001	6.90	84
Mystic River	2003	8.10	70
Snatch.	2000	7.90	64
UHF	1989	6.60	64
Fight Club	1999	8.50	63

Average rank of movies per year?

year	avg_rank
1972	9.00
1977	8.80
1978	7.50
1984	5.80
1986	8.20
1987	7.20
1989	6.95
1991	7.80
1992	7.90
1994	8.85

Examples

Average rank of movies per year?

Movies

id	name	year	rank
10920	Aliens	1986	8.20
17173	Animal House	1978	7.50
18979	Apollo 13	1995	7.50
30959	Batman Begins	2005	0.00
46169	Braveheart	1995	8.30
109093	Fargo	1996	8.20
111813	Few Good Men, A	1992	7.50
112290	Fight Club	1999	8.50
116907	Footloose	1984	5.80
124110	Garden State	2004	8.30

$$_{year}\mathcal{G}_{avg\_rank=avg(rank)}\ (Movies)$$

year	avg_rank
1972	9.00
1977	8.80
1978	7.50
1984	5.80
1986	8.20
1987	7.20
1989	6.95
1991	7.80
1992	7.90
1994	8.85

Example

How many distinct roles are there in each movie?

- the needed data are in two different relations
  - movie information in Movies
  - role information in Roles
- we need to join and then aggregate
  - $r_1 = \mathit{Movies} \bowtie_{\mathit{Movies.id}=\mathit{Roles.movie\_id}} \mathit{Roles}$

id	name	year	rank	actor_id	role
10920	Aliens	1986	8.20	16844	Lydecker
10920	Aliens	1986	8.20	36641	Russ Jorden
10920	Aliens	1986	8.20	42278	Cpl. Dwayne Hicks
10920	Aliens	1986	8.20	144260	Doctor
10920	Aliens	1986	8.20	204493	Timmy Jorden

  - ${}_{\mathit{name}}\mathcal{G}_{\mathit{nb\_role}=\mathit{count}(\mathit{role})}(r_1)$

name	nb_role
Aliens	30
Animal House	43
Apollo 13	97
Batman Begins	62
Braveheart	60

Changelog

- November 2020: initial version

Licence

Version

Last git commit: 2020-11-23
By: Fabrice Rossi (Fabrice.Rossi@apiacoa.org)
Git hash: 312a0636ceb585db2da88a95e73b59651b34a3fb