Biomedicine and bioinformatics databases (module *Fundamentals of database systems*)

Introduction

organization and content of this module, relational model and relational algebra

Alberto Belussi

ver. 1.0, 30/09/2017

Summary

1. Structure of the course
2. The relational model
3. The relational algebra

Biomedicine and bioinformatics databases

Module *Fundamentals of database systems*: content

- Relational approach: (i) relational model, (ii) relational algebra and relational calculus.
- Object-relational approach: (i) SQL3 model, (ii) SQL3 query language, (iii) SQL3 in PostgreSQL.
- Data modeling: (i) conceptual data modeling (Entity Relationship model and UML class diagram), (ii) mapping rules from UML classes towards SQL3 model, (iii) functional dependencies and normal forms.
- Spatio-temporal data: (i) Valid time and transaction time (ii) temporal relations, (iii) TSQL2, (iv) Spatial data types (OGC and ISO standards), (v) spatial relations.
- PostgreSQL for spatial and temporal data: (i) PostGIS, (ii) Temporal relations, (iii) spatio-temporal queries in PostgreSQL.
- Big Data: (i) MapReduce paradigm, (ii) Hadoop, (iii) Pig query language.

Timetable

Lessons

- Thursday 11.30 - 13.30 (lecture hall G)
- Friday 8.30 - 10.30 (lecture hall I)

Practice

- Thursday 13.30 14.30 (lab Gamma) starting from November

Written partial exams

- First partial exam (end of November): relational calculus, functional dependencies and normal forms.
- Second partial exam (end of January): data modeling (UML), SQL3, spatio-temporal databases, big data.

Written complete exam

- during the examination sessions (February 2018, June-July 2018, September 2018) on the whole program.

I suppose that you already know

Notions

- The basic construct of the relational model: **relation** (or table)
- The operators of the relational algebra: syntax and semantics
- The query language SQL2.
- Some basic notions of logic (logic operator: and, or, not, first order logic?)
- Some basic notions of data modeling (Entity relationship model)

Ability

- Writing a non trivial query in relational algebra and SQL
- Specifying a conceptual schema using the ER model

Let's start. . .

1. Structure of the course
2. The relational model
   - Introduction
   - Formal definition
3. The relational algebra

Relational model

History

The **relational model** was defined in 1970 by E. F. Codd and it had its maximum development in the '80.
The **relational model** is still the most used data model in the current database systems and in many cases it is the starting point of the new approaches for dealing the new types and volumes of data.

Main characteristic

The **relational model** and the corresponding systems are based on a declarative paradigm. Declarations are used to define the data structures and to query the data base.

Relational model

Constructs of the relational model

The basic construct of the relational model for data representation is the **relation** (or table).
In the model we find also: a set of **basic domains**, some **integrity constraints** (keys, primary keys and referential integrity constraints).
Moreover some manipulation languages are defined for querying and data insert/update and delete: among them we find the **relational algebra** and the **relational calculus**.

Relation example

Relation

Example: Relation *Patient*

PATIENT

code	**surname**	**name**	**dateOfBirth**
CMB. . .	Combi	Carlo	. . .
BLS. . .	Belussi	Alberto	. . .
PSN. . .	Posenato	Roberto	. . .
RSS. . .	Rossi	Mario	. . .
. . .	. . .	. . .	. . .

where the head row of the table describes the relation schema, while all the other rows are not ordered instances (called tuples).

Formal definition of a relational schema

Basic sets of values and symbols

We introduce the following sets of values/symbols to be used for the instances of data (table content) and for identifiers (or names) in the schema of the data structures (table schema).

- DOM is the set of constants (domain values) to be used in the instances of data,
- ATTNAME is the set of names for attributes (column names),
- RELNAME is the set of names for relations (table names).

Formal definition of a relational schema

Basic sets of values and symbols

We introduce the following sets of values/symbols to be used for the instances of data (table content) and for identifiers (or names) in the schema of the data structures (table schema).

- DOM is the set of constants (domain values) to be used in the instances of data,
- ATTNAME is the set of names for attributes (column names),
- RELNAME is the set of names for relations (table names).

Formal definition of a relational schema

Basic sets of values and symbols

We introduce the following sets of values/symbols to be used for the instances of data (table content) and for identifiers (or names) in the schema of the data structures (table schema).

- DOM is the set of constants (domain values) to be used in the instances of data,
- ATTNAME is the set of names for attributes (column names),
- RELNAME is the set of names for relations (table names).

Formal definition of a relational schema

Basic sets of values and symbols

We introduce the following sets of values/symbols to be used for the instances of data (table content) and for identifiers (or names) in the schema of the data structures (table schema).

- DOM is the set of constants (domain values) to be used in the instances of data,
- ATTNAME is the set of names for attributes (column names),
- RELNAME is the set of names for relations (table names).

Formal definition of a relational schema

Definition 1 (Function SORT(R))

Any relation has a name $R \in \text{RELNAME}$ and a finite set of attributes given by the function SORT(R)

$$\text{SORT} : \text{RELNAME} \longrightarrow \underbrace{\mathcal{P}(\text{ATTNAME})}_{\textit{Power Set}}$$

Example:

$$R(U) \text{ or } R(A_1, A_2, \ldots, A_n) \rightarrow \text{SORT(R)} = U = \{A_1, A_2, \ldots, A_n\}$$

Formal definition of a relational schema

Definition 1 (Function SORT(R))

Any relation has a name $R \in \text{RELNAME}$ and a finite set of attributes given by the function SORT(R)

$$\text{SORT} : \text{RELNAME} \longrightarrow \underbrace{\mathcal{P}(\text{ATTNAME})}_{\textit{Power Set}}$$

Example:

$$R(U) \text{ or } R(A_1, A_2, \ldots, A_n) \rightarrow \text{SORT(R)} = U = \{A_1, A_2, \ldots, A_n\}$$

Formal definition of a relational schema

Definition 2 (Arity of a relation)

The arity of a relation R is the number of its attributes. ARITY(R) = |SORT(R)|

Example:

$$\text{SORT(R)} = \{A_1, A_2, A_3\} \rightarrow \text{ARITY(R)} = |\{A_1, A_2, A_3\}| = 3$$

Formal definition of a relational schema

Definition 2 (Arity of a relation)

The arity of a relation R is the number of its attributes. ARITY(R) = |SORT(R)|

Example:

$$\text{SORT(R)} = \{A_1, A_2, A_3\} \rightarrow \text{ARITY(R)} = |\{A_1, A_2, A_3\}| = 3$$

Formal definition of a tuple

Definition 3 (Tuple of a relation)

A tuple of $R(U)$ is a function:

$$t : U \longrightarrow \text{DOM}$$

where U is finite and $U \subset$ ATTNAME.
If $U = \emptyset \rightarrow t = <>$ (empty tuple)

Example:

U = {cf, surname, name, dateOfBirth}
t[cf] = "CMB...", t[surname] = "Combi"
t[name] = "Carlo", t[dateOfBirth] = "24/05/1962"

Formal definition of a tuple

Definition 3 (Tuple of a relation)

A tuple of $R(U)$ is a function:

$$t : U \longrightarrow \text{DOM}$$

where U is finite and U $\subset$ ATTNAME.
If U $= \emptyset \rightarrow$ t $= <>$ (empty tuple)

Example:

U = {cf, surname, name, dateOfBirth}
t[cf] = "CMB. . . ", t[surname] = "Combi"
t[name] = "Carlo", t[dateOfBirth] = "24/05/1962"

Formal definition of a tuple (unnamed perspective)

Tuple as a list of values

In this perspective a relation only has an arity, and the tuples are elements of the Cartesian product of DOM. Thus, a tuple t of arity n is defined as follows:

$$t \in \text{DOM}^n$$

where $\text{DOM}^n = (\underbrace{\text{DOM} \times \text{DOM} \times \cdots \times \text{DOM}}_{n\text{—}times})$

Example:

t = <"CMB. . . ", "Combi", "Carlo", "24/05/1962">
t[1] = "CMB. . . ", t[2] = "Combi"
t[3] = "Carlo", t[4] = "24/05/1962"

Formal definition of a tuple (unnamed perspective)

Tuple as a list of values

In this perspective a relation only has an arity, and the tuples are elements of the Cartesian product of DOM. Thus, a tuple *t* of arity *n* is defined as follows:

$$t \in \text{DOM}^n$$

where $\text{DOM}^n = (\underbrace{\text{DOM} \times \text{DOM} \times \cdots \times \text{DOM}}_{n-times})$

Example:

t = <"CMB. . . ", "Combi", "Carlo", "24/05/1962">

t[1] = "CMB. . . ", t[2] = "Combi"

t[3] = "Carlo", t[4] = "24/05/1962"

Notation for specifying a relation schema

Definition 4 (Relation schema)

The schema of a relation is composed of its name followed by the list of the names of its attributes: $R(U)$ or $R(A_1, \ldots, A_k)$.

Example:

PATIENT(cf, surname, name, dateOfBirth)

Notation for specifying a relation schema

Definition 4 (Relation schema)

The schema of a relation is composed of its name followed by the list of the names of its attributes: $R(U)$ or $R(A_1, \ldots, A_k)$.

Example:

PATIENT(cf, surname, name, dateOfBirth)

Notation for specifying a database schema

Definition 5 (Database schema)

The schema of a database is a finite and non-empty set of relation schemas: $\mathcal{R} = \{R_1(U_1), \ldots, R_n(U_n)\}$.

Example:

$$\mathcal{R} = \{\text{PATIENT(cf, surname, name, dateOfBirth)}, \text{SYMPTOM(names, descr, severity)}, \text{EVENT(patient, symptom, date)}\}$$

Notation for specifying a database schema

Definition 5 (Database schema)

The schema of a database is a finite and non-empty set of relation schemas: $\mathcal{R} = \{R_1(U_1), \ldots, R_n(U_n)\}$.

Example:

$$\mathcal{R} = \{\text{PATIENT(cf, surname, name, dateOfBirth)}, \\ \text{SYMPTOM(names, descr, severity)}, \\ \text{EVENT(patient, symptom, date)}\}$$

Notation for specifying a relation instance

Definition 6 (Relation instance)

The instance of a relation on a schema R is a finite set of tuple on SORT(R):

$$I(R) = \{t_1, t_2, \ldots, t_n\}$$

where t_i : SORT(R) $\rightarrow$ DOM, $(1 \leqslant i \leqslant n)$

Notation for specifying a database instance

Definition 7 (Database instance)

The instance of a database on a schema $\mathcal{R} = \{R_1(U_1), \ldots, R_n(U_n)\}$ is the set of all the relation instances of the relation schemas belonging to $\mathcal{R}$:

$$I(\mathcal{R}) = \{I(R_1), I(R_2), \ldots, I(R_n)\}$$

Relational algebra characteristics

- it is a query language for a relational database
- it is a **procedural** language: it allows the user to specify expressions as sequence of operators that, when applied to the instances of the relations of the database, produce the query result
- it is an **algebra** for relations, thus its expressions always produce a relation as result
- it is not used by the front end of the systems; they prefer declarative query language (like SQL)
- it is used by the systems for the internal representation of a **query execution plan.**

Relational algebra characteristics

- it is a query language for a relational database
- it is a **procedural** language: it allows the user to specify expressions as sequence of operators that, when applied to the instances of the relations of the database, produce the query result
- it is an **algebra** for relations, thus its expressions always produce a relation as result
- it is not used by the front end of the systems; they prefer declarative query language (like SQL)
- it is used by the systems for the internal representation of a **query execution plan.**

Relational algebra characteristics

- it is a query language for a relational database
- it is a **procedural** language: it allows the user to specify expressions as sequence of operators that, when applied to the instances of the relations of the database, produce the query result
- it is an **algebra** for relations, thus its expressions always produce a relation as result
- it is not used by the front end of the systems; they prefer declarative query language (like SQL)
- it is used by the systems for the internal representation of a **query execution plan.**

Relational algebra characteristics

- it is a query language for a relational database
- it is a **procedural** language: it allows the user to specify expressions as sequence of operators that, when applied to the instances of the relations of the database, produce the query result
- it is an **algebra** for relations, thus its expressions always produce a relation as result
- it is not used by the front end of the systems; they prefer declarative query language (like SQL)
- it is used by the systems for the internal representation of a **query execution plan.**

Relational algebra characteristics

- it is a query language for a relational database
- it is a **procedural** language: it allows the user to specify expressions as sequence of operators that, when applied to the instances of the relations of the database, produce the query result
- it is an **algebra** for relations, thus its expressions always produce a relation as result
- it is not used by the front end of the systems; they prefer declarative query language (like SQL)
- it is used by the systems for the internal representation of a **query execution plan**.

Operators

The set of basic operators (SPJRU- algebra)

This set contains the following operators of the algebra:

- selection (σ)
- projection (Π)
- join ($\bowtie$)
- renaming (ρ)
- union ($\cup$)
- difference ($-$)

Note

- All operators represent closed operations on the set of relations of homogeneous tuples; thus, they always produce as result a relation of homogeneous tuples.

Operators

The set of basic operators (SPJRU- algebra)

This set contains the following operators of the algebra:

- selection (σ)
- projection (Π)
- join ($\bowtie$)
- renaming (ρ)
- union ($\cup$)
- difference ($-$)

Note

- All operators represent closed operations on the set of relations of homogeneous tuples; thus, they always produce as result a relation of homogeneous tuples.

Database used for query examples (1/2)

STUDENT

mat	sn	na	db
VR000001	Rossi	Luca	'1/1/1990'
VR000002	Bianchi	Andrea	'1/3/1993'
VR000003	Verdi	Sonia	'10/10/1995'
VR000004	Rossini	Stefania	'3/4/1993'
VR000005	Giusti	Sara	'2/2/1996'
...	...	...	...

EXAM

mat	de	gr	cc
VR000001	01/03/2006	30	DB
VR000002	01/03/2006	18	ALG
VR000003	06/07/2010	20	GEN
VR000004	01/03/2006	25	MBI
VR000005	20/07/2010	23	PRO
...	...	...	...

EMPLOYEE

mat	su	na	db
VR000001	Rossi	Luca	'1/1/1990'
VR000009	Neri	Massimo	'10/1/1990'
...	...	...	...

COURSE

cc	tea	cr	yr
BD	Belussi	6	2016-17
ALG	Cicalese	12	2016-17
GEN	Delledonne	6	2015-16
MBI	Perduca	6	2015-16
PRO	Giugno	12	2016-17
...	...	...	...

Database used for query examples (2/2)

EXAM_ENROLLMENT

mat	**dn**	**cc**
VR000001	01/03/2016	BD
VR000002	01/03/2016	ALG
VR000003	06/07/2015	GEN
VR000004	01/03/2016	MBI
VR000005	20/07/2016	SED
VR000005	22/07/2016	FIS1
VR000003	09/11/2016	GEN
...	...	...

We introduce the following short names for attributes:

mat = student code
sn = surname
na = name
db = date of birth
tea = teacher
gr = grade
cc = course code
cr = credits
de = date of exam
yr = academic year
dn = date of exam enrollment

Specific operators: Projection

Projection: introduction

- it is a unary operator: it applies to one input relation R
- it requires a set of attributes as parameter: $\{A_1, \ldots, A_k\}$
- for each tuple t of R it returns a tuple that is obtained from t by dropping all the attributes that are not contained in the set $\{A_1, \ldots, A_k\}$ *(vertical cut)*
- it eliminate duplicate tuples that can be generated after attribute dropping *(operation closure)*

Specific operators: Projection Π

Definition 8 (Projection)

Given a relation of schema $R(U)$ and instance $I(R)$, $\Pi_{A_1,\ldots,A_k}(R)$ produces a new relation *RIS* with schema:

$$\text{SORT(RIS)} = \{A_1, \ldots, A_k\}$$

and content:

$$I(RIS) = \{t \mid \exists t' \in I(R) : \; t'[A_1, \ldots, A_k] = t\}$$

where $\{A_1, \ldots, A_k\} \subseteq \text{SORT(R)}$ and
$t'[A_1, \ldots, A_k]$ is a tuple t_0 on $\{A_1, \ldots, A_k\}$ such that
$t_0[A_1] = t'[A_1], \ldots, t_0[A_k] = t'[A_k]$.

Specific operators: Projection Π

Projection: examples

Q1: *Find the student code, the surname and name of every student*

In line expression:

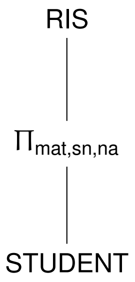
$$RIS \leftarrow \Pi_{mat,sn,na}(STUDENT)$$

Result:

RIS

mat	**sn**	**na**
VR000001	Rossi	Luca
VR000002	Bianchi	Andrea
VR000003	Verdi	Sonia
VR000004	Rossini	Stefania
VR000005	Giusti	Sara

Graph expression:



Specific operators: Selection

Selection: introduction

- it is a unary operator: it applies to one input relation R
- it requires a boolean expression as parameter: $cond(x)$
- for each tuple t of R it returns the same tuple if it satisfies the boolean expression $cond(x)$, i.e. the expression $cond(t)$ is evaluated true *(horizontal cut)*
- it can produce as result an empty relation.

Specific operators: Selection σ

Definition 9 (Selection)

Given a relation of schema $R(U)$ and instance $I(R)$, $\sigma_{cond}(R)$ produces a new relation RIS with schema:

$$\text{SORT(RIS)} = \text{SORT(R)}$$

and with content:

$$I(RIS) = \{t | \exists t' \in I(R) : cond(t') \wedge t' = t\}$$

where *cond* is a formula obtained by combining through the logical operators $\wedge, \vee, \neg$ atomic formulas having the form $A\ \theta\ B$ or $A\ \theta\ c$, $\theta \in \{=, \neq, <, \leqslant, >, \geqslant\}$, $A \in \text{SORT(R)}$, $B \in \text{SORT(R)}$ and $c \in \text{DOM}$.

Specific operators: Selection σ

Selection: examples

Q1: *Find the surname, the name and the date of birth of the student with code 'VR000001'*

In line expression:

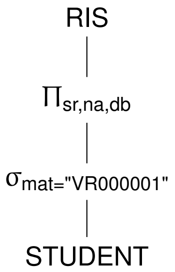
$$RIS \longleftarrow \Pi_{sn,na,db}(\sigma_{mat='VR000001'}(STUDENT))$$

Result:

RIS

sn	na	db
Rossi	Luca	'1/1/1990'

Graph expression:



Specific operators: Renaming

Renaming: introduction

- it is a unary operator: it applies to one input relation R
- it requires a mapping between old and new attribute name as parameter: $A_1, \ldots, A_k \rightarrow B_1, \ldots, B_k$
- for each tuple t of R it returns the same tuple with the new attribute names according to the mapping that is specified in the parameter.

Specific operators: Renaming ρ

Definition 10 (Renaming)

Given a relation of schema $R(U)$ and instance $I(R)$, $\rho_{A_1,\ldots,A_k \rightarrow B_1,\ldots,B_k}(R)$ produces a new relation RIS with schema:

$$\mathrm{SORT(RIS)} = \mathrm{SORT(R)} - \{A_1, \ldots, A_k\} \cup \{B_1, \ldots, B_k\}$$

and with content:

$$I(RIS) = \{t | \exists t' \in I(R) :$$

$$t'[A_1, \ldots, A_k, A_{k+1}, \ldots, A_n] = t[B_1, \ldots, B_k, A_{k+1}, \ldots, A_n]\}$$

where: $\{A_1, \ldots, A_k\} \subseteq \mathrm{SORT(R)}$ and $\{B_1, \ldots, B_k\} \not\subseteq \mathrm{SORT(R)}$.

For sake of simplicity when it is necessary to rename all the attributes of a relation in the same way (for example by adding a top bar) we will use the following notation $\rho_{X \rightarrow \overline{X}}(R)$ where $X = \{A_1, \ldots, A_n\}$ and $\overline{X} = \{\overline{A_1}, \ldots, \overline{A_n}\}$.

Specific operators: Renaming ρ

Renaming: examples

Q1: *Change the attribute name of the relation STUDENT from (sn, na) to (surname, name)*

In line expression:

$$RIS \longleftarrow \rho_{sn,na \rightarrow surname,\, name}(STUDENT)$$

Result:

Student

mat	**surname**	**name**	**dn'**
VR000001	Rossi	Luca	...
VR000002	Bianchi	Andrea	...
VR000003	Verdi	Sonia	...
VR000004	Rossini	Stefania	...
VR000005	Giusti	Sara	...

Graph expression:

RIS

|

$\rho_{sn,na \rightarrow surname,name}$

|

STUDENT

Join operators: Natural join

Natural join: introduction

- it is a binary operator: it applies to two input relations R and S
- it requires no parameters
- for each pair of tuples t_r, t_s, where $t_r \in I(R)$ and $t_s \in I(S)$, it returns a tuple in the result merging t_r and t_s only if t_r and t_s has the same value in the shared attributes (i.e., the attributes that appear both in R and S)

Join operators: Natural join $\bowtie$

Definition 11 (Natural join)

Given two relations of schema $R(U)$, $S(V)$ and instance $I(R)$ and $I(S)$, $R \bowtie S$ produces a new relation RIS with schema:

$$\text{SORT(RIS)} = \text{SORT(R)} \cup \text{SORT(S)}$$

and with content:

$$I(RIS) = \{t | (\exists t_r \in I(R) : t_r = t[\text{SORT(R)}]) \wedge$$

$$(\exists t_s \in I(S) : t_s = t[\text{SORT(S)}])\}$$

Join operators: Natural join $\bowtie$

Natural join: examples

Q1: *Find the exams of each student returning the date, grade and course of the exam together with all attributes of the student*

In line expression:

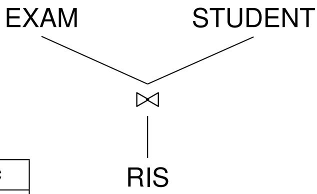
$$\text{RIS} \leftarrow \text{STUDENT} \bowtie \text{EXAM}$$

Result:

Student $\bowtie$ Exam

mat	**sn**	**na**	**db**	**de**	**gr**	**cc**
VR000001	Rossi	Luca	...	01/03/2006	30	BD
VR000002	Bianchi	Andrea	...	01/03/2006	18	SO
VR000003	Verdi	Sonia	...	06/07/2010	20	SDR
VR000004	Rossini	Stefania	...	01/03/2006	25	ASP
VR000005	Giusti	Sara	...	20/07/2010	23	SED

Graph expression:



Set operators: Union

Union: introduction

- it is a binary operator: it applies to two input relations *R* and *S*
- it requires no parameters
- it requires that *R* and *S* **have the same schema**
- it returns in the result both the tuples of *R* and the tuples of *S*

Set operators: Union $\cup$

Definition 12 (Union)

Given two relations of schema $R(U)$, $S(U)$, where SORT(R) $=$ SORT(S), and instance $I(R)$ and $I(S)$, $R \cup S$ produces a new relation RIS with schema:

$$\text{SORT(RIS)} = \text{SORT(R)} = \text{SORT(S)}$$

and with content:

$$I(RIS) = \{t | (\exists t_r \in I(R) : t_r = t) \vee (\exists t_s \in I(S) : t_s = t)\}$$

Set operators: Union ∪

Union: examples

Q1: *Find the set containing both students and employees*

In line expression:

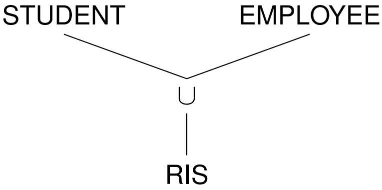
RIS ← STUDENT ∪ EMPLOYEE

Result:

STUDENT ∪ EMPLOYEE

mat	sr	na	db
VR000001	Rossi	Luca	...
VR000002	Bianchi	Andrea	...
VR000003	Verdi	Sonia	...
VR000004	Rossini	Stefania	...
VR000005	Giusti	Sara	...
VR000009	Neri	Massimo	...

Graph expression:



Set operators: Difference

Difference: introduction

- it is a binary operator: it applies to two input relations R and S
- it requires no parameters
- it requires that R and S have the same schema
- it returns in the result the tuples of R that are not tuples of S

Set operators: Difference $-$

Definition 13 (Difference)

Given two relations of schema $R(U)$, $S(U)$, where SORT(R) $=$ SORT(S), and instance $I(R)$ and $I(S)$, $R - S$ produces a new relation R/S with schema:

$$\text{SORT(RIS)} = \text{SORT(R)} = \text{SORT(S)}$$

and with content:

$$I(R/S) = \{t | (\exists t_r \in I(R) : t_r = t) \wedge (\nexists t_s \in I(S) : t_s = t)\}$$

Set operators: Difference −

Difference: examples

Q1: *Find the students that are not employees*

In line expression:

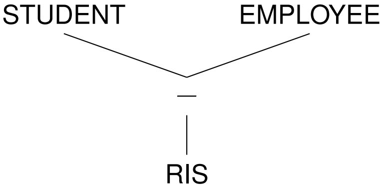
RIS ← STUDENT − EMPLOYEE

Result:

STUDENT ∪ EMPLOYEE

mat	**sr**	**na**	**db**
VR000002	Bianchi	Andrea	...
VR000003	Verdi	Sonia	...
VR000004	Rossini	Stefania	...
VR000005	Giusti	Sara	...

Graph expression:



Example 1 (use of the difference operator)

Query 1

Find all the students that have not passed the exam of “Databases" yet (cc = “DB”).

Query 1: discussion

The idea is to subtract from the tuples of STUDENT the tuples that representing students that have passed the “Databases" exam.
Notice that:

- a common mistake in queries of this type is to select from the table EXAM only the tuples that satisfy the condition: cc $\neq$ “DB”.
- This is not nearly enough! Indeed, there could be students that have not passed any exams, or students that have passed the “Databases" exam, but also other exams. In the first case the students do not appear in the result, while they should, and in the second case the students will be part of the result, while they should not.

Example 1 (use of the difference operator)

Query 1

Find all the students that have not passed the exam of “Databases" yet (cc = “DB”).

Query 1: discussion

The idea is to subtract from the tuples of STUDENT the tuples that representing students that have passed the “Databases" exam.

Notice that:

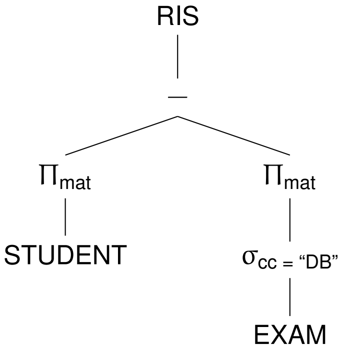
- a common mistake in queries of this type is to select from the table EXAM only the tuples that satisfy the condition: cc $\neq$ “DB”.
- This is not nearly enough! Indeed, there could be students that have not passed any exams, or students that have passed the “Databases" exam, but also other exams. In the first case the students do not appear in the result, while they should, and in the second case the students will be part of the result, while they should not.

Example 1 (use of the difference operator) - solution

In Line:

$$RIS \leftarrow (\Pi_{mat}(STUDENT)) - (\Pi_{mat}(\sigma_{cc="DB"}(EXAM)))$$

Grafo ad albero:



Example 2 (the query for finding "the next")

Query 2

Find for all the students the pairs of consecutive exams (two exams e_1, e_2 are consecutive if the date of e_1 is before the date of e_2 and there is no other exams of the same student with a date that falls between e_1 and e_2); in the result it is required to show the code of the student and the code of the courses.

Query 2: discussion

The idea is to find all the pairs of exams e_1, e_2 such that the date of e_1 comes before the date of e_2; from this result then we subtract the pairs of exams that have another exam with a date that falls between the date of e_1 and the date of e_2. Notice that:

- a common mistake in queries of this type is to forget to subtract the second part, i.e. the pairs e_1, e_2 that have another exam with a date that falls between the date of e_1 and the date of e_2.

Example 2 (the query for finding "the next")

Query 2

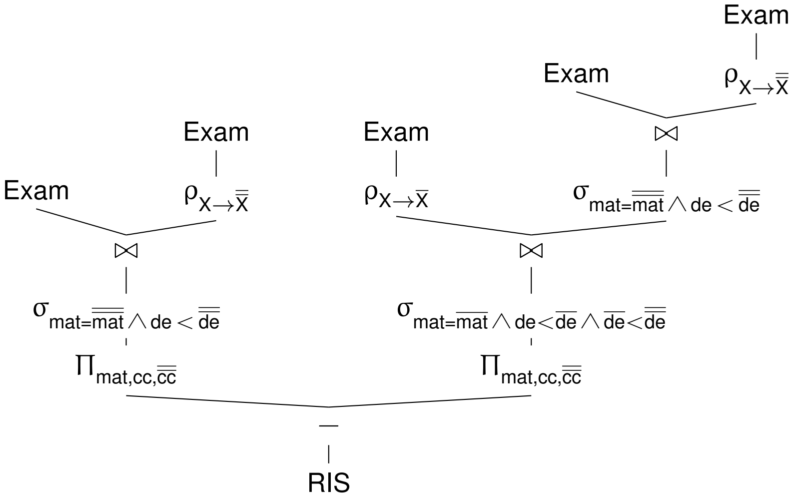
Find for all the students the pairs of consecutive exams (two exams e_1, e_2 are consecutive if the date of e_1 is before the date of e_2 and there is no other exams of the same student with a date that falls between e_1 and e_2); in the result it is required to show the code of the student and the code of the courses.

Query 2: discussion

The idea is to find all the pairs of exams e_1, e_2 such that the date of e_1 comes before the date of e_2; from this result then we subtract the pairs of exams that have another exam with a date that falls between the date of e_1 and the date of e_2. Notice that:

- a common mistake in queries of this type is to forget to subtract the second part, i.e. the pairs e_1, e_2 that have another exam with a date that falls between the date of e_1 and the date of e_2.

Example 2 (the query for the next) - solution



Exercises

- **EX1** Find all the student that passed all the registered exams with grade = 30.
- **EX2** Find all exams that the student with code VR000004 passed; in the result it is required to show the name of the course, the date of the exam, the grade and the credits.
- **EX3** Let A and B be two exams. If A is a requirement for B, then it means that a student cannot enroll for an exam of B without having a grade of A. Find all students that have violated the following rule: Physic 1 (cc = phy1) is a requirement for Physic 2 (cc = phy2); in the result it is required to show the the code, surname and name of the student.

Exercises

The system for the enrollment of students to the exams does not check whether a student has already passed an exam when he/she tries to enroll for it.

- Find all the students that enrolled for an exam that they have already passed; in the result it is required to show the code of the student, the name of the course, the date of exam registration and the date of the second enrollment.