Basi di Dati e
Sistemi Informativi I

Relational algebra

(2/2)

Join

- The most typical operator in relational algebra
- allows to establish connections among data in different relations, taking into advantage the "value-based" nature of the relational model
- Two main versions of the join:
  - "natural" join: takes attribute names into account
  - "theta" join
- They are all denoted by the symbol $\bowtie$

A natural join

The natural join is an operator that correlates data in different relations, on the basis of equal values of attributes with the same name

The result of the join is a relation on the union of the sets of attributes of the oprands: the tuples are obtained by combining the tuples of the operands with equal values on the common attributes, in the example the attribute Department

r_1

Employee	Department
Smith	sales
Black	production
White	production

r_2

Department	Head
production	Mori
sales	Brown

$r_1 \bowtie r_2$

Employee	Department	Head
Smith	sales	Brown
Black	production	Mori
White	production	Mori

Natural join: definition

- $r_1 (X_1)$, $r_2 (X_2)$
- $r_1 \bowtie r_2$ (natural join of r_1 and r_2) is a relation on X_1X_2 (the union of the two sets):

$$\{ t \text{ on } X_1X_2 \mid t[X_1] \in r_1 \text{ and } t[X_2] \in r_2 \}$$

or, equivalently

$$\{ t \text{ on } X_1X_2 \mid \text{exist } t_1 \in r_1 \text{ and } t_2 \in r_2 \text{ with } t[X_1] = t_1 \text{ and } t[X_2] = t_2 \}$$

Natural join: comments

- The tuples in the result are obtained by combining tuples in the operands with equal values on the common attributes
- If we indicate the common attributes as $X_{1,2}$ (i.e., $X_{1,2}= X_1 \cap X_2$), then the two conditions $t[X_1]=t_1$ and $t[X_2]=t_2$ imply (since $X_{1,2} \subseteq X_1$ and $X_{1,2} \subseteq X_2$) that $t[X_{1,2}]= t_1 [X_{1,2}]$ and $t[X_{1,2}]= t_2 [X_{1,2}]$ and thus $t_1[X_{1,2}]= t_1 [X_{1,2}]$
- The degree of the result of a join is less than or equal to the sum of the degrees of the two operands, because the common attributes of the operands appear only once in the result.
- The common attributes often form a key of one of the operands (remember: references are realized by means of keys, and we join in order to follow references)

Another natural join

Offences

Code	Date	Officer	Dept	Registartion
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

Cars

Registration	Dept	Owner	…
6544 XY	75	Cordon Edouard	…
7122 HT	75	Cordon Edouard	…
5694 FR	75	Latour Hortense	…
6544 XY	47	Mimault Bernard	…

Offences $\bowtie$ Cars

Code	Date	Officer	Dept	Registration	Owner	…
143256	25/10/1992	567	75	5694 FR	Latour Hortense	…
987554	26/10/1992	456	75	5694 FR	Latour Hortense	…
987557	26/10/1992	456	75	6544 XY	Cordon Edouard	…
630876	15/10/1992	456	47	6544 XY	Cordon Edouard	…
539856	12/10/1992	567	47	6544 XY	Cordon Edouard	…

Yet another join

- Compare with the union:
  - the same data can be combined in various ways

Paternity

Father	Child
Adam	Cain
Adam	Abel
Abraham	Isaac
Abraham	Ishmael

Maternity

Mother	Child
Eve	Cain
Eve	Seth
Sarah	Isaac
Hagar	Ishmael

Paternity $\bowtie$ Maternity

Father	Child	Mother
Adam	Cain	Eve
Abraham	Isaac	Sarah
Abraham	Ishmael	Hagar

Complete Joins

- In the example we can say that each tuple of each of the operands contributes to at least one tuple of the result. In this case, the join is said to be *complete*.
- For each tuple t_1 of r_1, there exists a tuple t in $r_1 \bowtie r_2$ such that $t[X_1] = t_1$ (and similarly for r_2)

r_1

Employee	Department
Smith	sales
Black	production
White	production

r_2

Department	Head
production	Mori
sales	Brown

$r_1 \bowtie r_2$

Employee	Department	Head
Smith	sales	Brown
Black	production	Mori
White	production	Mori

Joins can be "incomplete"

- If a tuple does not have a "counterpart" in the other relation, then it does not contribute to the join ("dangling" tuple)

r_1

Employee	Department
Smith	sales
Black	production
White	production

r_2

Department	Head
production	Mori
purchasing	Brown

$r_1 \bowtie r_2$

Employee	Department	Head
Black	production	Mori
White	production	Mori

Joins can be empty

- As an extreme, we might have that no tuple has a counterpart, and all tuples are dangling

r_1

Employee	Department
Smith	sales
Black	production
White	production

r_2

Department	Head
marketing	Mori
purchasing	Brown

$r_1 \bowtie r_2$

Employee	Department	Head

The other extreme

- If each tuple of each operand can be combined with all the tuples of the other, then the join has a cardinality that is the product of the cardinalities of the operands

r_1

Employee	Project
Smith	A
Black	A
White	A

r_2

Project	Head
A	Mori
A	Brown

$r_1 \bowtie r_2$

Employee	Project	Head
Smith	A	Mori
Black	A	Brown
White	A	Mori
Smith	A	Brown
Black	A	Mori
White	A	Brown

How many tuples in a join?

Given $r_1(X_1)$, $r_2(X_2)$

- the join has a cardinality between zero and the products of the cardinalities of the operands:

  $$0 \leq | r_1 \bowtie r_2 | \leq | r_1 | \times | r_2 |$$

  ($| r |$ is the cardinality of relation r)

- moreover:
  - if the join is complete, then its cardinality is at least the maximum of $| r_1 |$ and $| r_2 |$
  - if $X_1 \cap X_2$ contains a key for r_2, then $| r_1 \bowtie r_2 | \leq | r_1 |$
  - if $X_1 \cap X_2$ is the primary key for r_2, and there is a referential constraint between $X_1 \cap X_2$ in r_1 and such a key, then $| r_1 \bowtie r_2 | = | r_1 |$

Outer joins

The fact that the join operator “leaves out” the tuples of a relation that have no counterpart in the other operand is useful in some cases but inconvenient in others, given the possibility of omitting important information.

- The *outer join* is a variant of the join, to keep all pieces of information from the operands
- It "pads with nulls" the tuples that have no counterpart
- Three variants:
  – “LEFT": only tuples of the first operand are padded
  – “RIGHT": only tuples of the second operand are padded
  – “FULL": tuples of both operands are padded

Outer joins

r_1

Employee	Department
Smith	sales
Black	production
White	production

r_2

Department	Head
production	Mori
purchasing	Brown

$r_1 \bowtie_{LEFT} r_2$

Employee	Department	Head
Smith	Sales	NULL
Black	production	Mori
White	production	Mori

$r_1 \bowtie_{RIGHT} r_2$

Employee	Department	Head
Black	production	Mori
White	production	Mori
NULL	purchasing	Brown

$r_1 \bowtie_{FULL} r_2$

Employee	Department	Head
Smith	Sales	NULL
Black	production	Mori
White	production	Mori
NULL	purchasing	Brown

N-ary join

- The natural join is
  - commutative: $r_1 \bowtie r_2 = r_2 \bowtie r_1$
  - associative: $(r_1 \bowtie r_2) \bowtie r_3 = r_1 \bowtie (r_2 \bowtie r_3)$
- Therefore, we can write n-ary joins without ambiguity:

$$r_1 \bowtie r_2 \bowtie \ldots \bowtie r_n$$

N-ary join

r_1

Employee	Department
Smith	sales
Black	production
Brown	marketing
White	production

r_2

Department	Division
production	A
marketing	B
purchasing	B

r_3

Division	Head
A	Mori
B	Brown

$r_1 \bowtie r_2 \bowtie r_3$

Employee	Department	Division	Head
Black	production	A	Mori
Brown	marketing	B	Brown
White	production	A	Mori

Intersection and cartesian product

- We have stated no hypothesis about the sets of attributes X_1 and X_2 on which the operands are defined. Therefore, the two sets could even be equal or be disjoint. Let us examine these extreme cases.
  - If $X_1=X_2$ then $r_1 \bowtie r_2 = r_1 \cap r_2$ since, by definition, the result is a relation on the union of the two sets of attributes, and must contain the tuples t such that $t[X_1] \in r_1$ and $t[X_2] \in r_2$. If $X_1=X_2$ then, the union of X_1 and X_2 is also equal to X_1, and thus t is defined on X_1: the definition thus requires that $t \in r_1$ and $t \in r_2$, and therefore coincides with the definition of intersection.
  - The natural join is defined also when the operands have no attributes in common. In this case no condition is imposed on tuples, and therefore the result contains tuples obtained by combining the tuples of the operands in all possible ways (we say that the join becomes a *cartesian product*)

Cartesian product: example

Employees

Employee	Project
Smith	A
Black	A
Black	B

Projects

Code	Name
A	Venus
B	Mars

Employees $\bowtie$ Projects

Employee	Project	Code	Name
Smith	A	A	Venus
Black	A	A	Venus
Black	B	A	Venus
Smith	A	B	Mars
Black	A	B	Mars
Black	B	B	Mars

Theta-join

- In most cases, a cartesian product is meaningful only if followed by a selection:
  - **theta-join:** a derived operator

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

  - if F is a conjunction of equalities, then we have an **equi-join**

- From the practical point of view, the theta-join and the equi-join are very important. This is because most current database systems do not take advantage of attribute names in order to combine relations, and thus use the equi-join and theta-join rather than the natural join.

Equi-join: example

Employees

Employee	Project
Smith	A
Black	A
Black	B

Projects

Code	Name
A	Venus
B	Mars

Employes $\bowtie_{\text{Project=Code}}$ Projects

Employee	Project	Code	Name
Smith	A	A	Venus
Black	A	A	Venus
Black	B	B	Mars

Queries

- A query is a function from database instances to relations: *given a schema* ***R*** *of a database, a query is a function that, for every instance* ***r*** *of* ***R****, produces a relation on a given set of attributes X*
- Queries are formulated in relational algebra by means of expressions over relations: we indicate by means of *E*(**r**) the result of the application of the expression *E* to the database **r**

A database for the examples

Employees

Number	Name	Age	Salary
101	Mary Smith	34	40
103	Mary Bianchi	23	35
104	Luigi Neri	38	61
105	Nico Bini	44	38
210	Marco Celli	49	60
231	Siro Bisi	50	60
252	Nico Bini	44	70
301	Steve Smith	34	70
375	Mary Smith	50	65

Supervision

Head	Employee
210	101
210	103
210	104
231	105
301	210
301	231
375	252

Example 1

- Find the numbers, names and ages of employees earning more than 40 thousand.

Employees

Number	Name	Age	Salary
101	Mary Smith	34	40
103	Mary Bianchi	23	35
104	Luigi Neri	38	61
105	Nico Bini	44	38
210	Marco Celli	49	60
231	Siro Bisi	50	60
252	Nico Bini	44	70
301	Steve Smith	34	70
375	Mary Smith	50	65

Supervision

Head	Employee
210	101
210	103
210	104
231	105
301	210
301	231
375	252

Example 1

- Find the numbers, names and ages of employees earning more than 40 thousand.

$$\pi_{\text{Number,Name,Age}}(\sigma_{\text{Salary}>40}(\textsc{Employees})) \tag{3.1}$$

Number	Name	Age
104	Luigi Neri	38
210	Marco Celli	49
231	Siro Bisi	50
252	Nico Bini	44
301	Steve Smith	34
375	Mary Smith	50

Example 2

- Find the registration numbers of the supervisors of the employees earning more than 40 thousand

Employees

Number	Name	Age	Salary
101	Mary Smith	34	40
103	Mary Bianchi	23	35
104	Luigi Neri	38	61
105	Nico Bini	44	38
210	Marco Celli	49	60
231	Siro Bisi	50	60
252	Nico Bini	44	70
301	Steve Smith	34	70
375	Mary Smith	50	65

Supervision

Head	Employee
210	101
210	103
210	104
231	105
301	210
301	231
375	252

Example 2

- Find the registration numbers of the supervisors of the employees earning more than 40 thousand

$$\pi_{\text{Head}}(\text{SUPERVISION} \bowtie_{\text{Employee}=\text{Number}}(\sigma_{\text{Salary}>40}(\text{EMPLOYEES}))) \qquad (3.2)$$

Head
210
301
375

Example 3

- Find the names and salaries of the supervisors of the employees earning more than 40 thousand

Employees

Number	Name	Age	Salary
101	Mary Smith	34	40
103	Mary Bianchi	23	35
104	Luigi Neri	38	61
105	Nico Bini	44	38
210	Marco Celli	49	60
231	Siro Bisi	50	60
252	Nico Bini	44	70
301	Steve Smith	34	70
375	Mary Smith	50	65

Supervision

Head	Employee
210	101
210	103
210	104
231	105
301	210
301	231
375	252

Example 3

- Find the names and salaries of the supervisors of the employees earning more than 40 thousand

$$\pi_{\mathrm{NameH,SalaryH}}\,(\rho_{\mathrm{NumberH,NameH,SalaryH,AgeH} \leftarrow \mathrm{Number,Name,Salary,Age}}\,(\mathrm{EMPLOYEES})$$
$$\bowtie_{\mathrm{NumberH=Head}}$$
$$(\mathrm{SUPERVISION} \bowtie_{\mathrm{Employee=Number}} (\sigma_{\mathrm{Salary}>40}(\mathrm{EMPLOYEES})))) \qquad (3.3)$$

NameH	SalaryH
Marco Celli	60
Steve Smith	70
Mary Smith	65

Example 4

- Find the employees earning more than their respective supervisors, showing registration numbers, names and salaries of the employees and supervisors

Employees

Number	Name	Age	Salary
101	Mary Smith	34	40
103	Mary Bianchi	23	35
104	Luigi Neri	38	61
105	Nico Bini	44	38
210	Marco Celli	49	60
231	Siro Bisi	50	60
252	Nico Bini	44	70
301	Steve Smith	34	70
375	Mary Smith	50	65

Supervision

Head	Employee
210	101
210	103
210	104
231	105
301	210
301	231
375	252

Example 4

- Find the employees earning more than their respective supervisors, showing registration numbers, names and salaries of the employees and supervisors

$$\pi_{Number,Name,Salary,NumberH,NameH,SalaryH}$$
$$(\sigma_{Salary>SalaryH}(\rho_{NumberH,NameH,SalaryH,AgeH \leftarrow Number,Name,Salary,Age}(\text{EMPLOYEES})$$
$$\bowtie_{NumberH=Head}(\text{SUPERVISION} \bowtie_{Employee=Number}(\text{EMPLOYEES})))) \qquad (3.4)$$

Number	Name	Salary	NumberH	NameH	SalaryH
104	Luigi Neri	61	210	Marco Celli	60
252	Nico Bini	70	375	Mary Smith	65

Example 5

- Find the registration numbers and names of the supervisors whose employees all earn more than 40 thousand

Employees

Number	Name	Age	Salary
101	Mary Smith	34	40
103	Mary Bianchi	23	35
104	Luigi Neri	38	61
105	Nico Bini	44	38
210	Marco Celli	49	60
231	Siro Bisi	50	60
252	Nico Bini	44	70
301	Steve Smith	34	70
375	Mary Smith	50	65

Supervision

Head	Employee
210	101
210	103
210	104
231	105
301	210
301	231
375	252

Example 5

- Find the registration numbers and names of the supervisors whose employees all earn more than 40 thousand

$$\pi_{\text{Number,Name}}(\text{EMPLOYEES} \bowtie_{\text{Number=Head}} (\pi_{\text{Head}}(\text{SUPERVISION}) - \pi_{\text{Head}}(\text{SUPERVISION} \bowtie_{\text{Employee=Number}} (\sigma_{\text{Salary} \leq 40}(\text{EMPLOYEES}))))) \qquad (3.5)$$

Number	Name
301	Steve Smith
375	Mary Smith

Equivalence of algebraic expressions

Equivalence: $E1 \equiv_{\mathbf{R}} E2$ if $E1(\mathbf{r})=E2(\mathbf{r})$, for every instance of **r** in **R**

$$\pi_{AB}(R1) \bowtie \pi_{AC}(R2) \equiv_R \pi_{ABC}(R1 \bowtie R2)$$

Absolute Equivalence: $E1 \equiv E2$ if $E1(\mathbf{r})=E2(\mathbf{r})$, for every schema **R**

$$\pi_{AB}(\sigma_{A>0}(R)) \equiv \sigma_{A>0}(\pi_{AB}(R))$$

Equivalence Transformations (1/2)

1. Atomization of selections: $\sigma_{F1 \wedge F2}(E) \equiv \sigma_{F1}(\sigma_{F2}(E))$
2. Cascading projections: $\pi_X(E) \equiv \pi_X(\pi_{XY}(E))$
3. Anticipation of selection wrt the join: $\sigma_F(E1 \bowtie E2) \equiv E1 \bowtie \sigma_F(E2)$
4. Anticipation of projection wrt the join: $\pi_{X1Y2}(E1 \bowtie E2) \equiv E1 \bowtie \pi_{Y2}(E2)$, with E1 and E2 defined on X1 and X2, respectively; and $Y2 \subset X2$ and $Y2 \supseteq X1 \cap X2$
5. Combination of a selection and acartesian product to form a theta-join: $\sigma_F(E1 \bowtie E2) \equiv E1 \bowtie_F E2$
6. Distribution of the selection wrt the union: $\sigma_F(E1 \cup E2) \equiv \sigma_F(E1) \cup \sigma_F(E2)$
7. Distribution of the selection wrt the difference: $\sigma_F(E1 - E2) \equiv \sigma_F(E1) - \sigma_F(E2)$

Equivalence Transformations (2/2)

8. Distribution of the projection wrt the union:
   $\pi_X(E1 \cup E2) \equiv \pi_X(E1) \cup \pi_X(E2)$
9. $\sigma_{F1 \vee F2}(R) \equiv \sigma_{F1}(R) \cup \sigma_{F2}(R)$
10. $\sigma_{F1 \wedge F2}(R) \equiv \sigma_{F1}(R) \cap \sigma_{F2}(R) \equiv \sigma_{F1}(R) \bowtie \sigma_{F2}(R)$
11. $\sigma_{F1 \wedge \neg(F2)}(R) \equiv \sigma_{F1}(R) - \sigma_{F2}(R)$

Algebra with null values

People

Name	Age	Salary
Aldo	35	15
Andrea	27	21
Maria	NULL	42

- $\sigma_{Age>30}$ (People): which tuples belong to the result? The first yes, the second no, but the third?
- We use two new forms of atomic conditions of selection: A IS NULL and A IS NOT NULL
- $\sigma_{Age>30 \wedge Age\ IS\ NULL}$(People)

Views

- In a relation database, there can exist **base** relations and **derived** relations (whose content is derived from the content of other relations)
- There are two types of derived relations: *materialized views* (derived relations that are stored in the DB) and *virtual relations* (defined by means of functions, not stored in the DB, but useable in the queries as if they were)
- Materialized views can be immediately available for queries, but it is difficult to maintain their content consistent with those of the relations from which they are derived
- Virtual relations must be recalculated for each query but produce no consistency problem

Exercise

Given the databse schema containing the relations:

FILMS(<u>FilmNumber</u>,Title,Director,Year,ProductionCost)

ARTISTS(<u>ActorNumber</u>,Surname,FirsName,Sex,Birthday,Nationality)

ROLES(<u>FilmNumber,ActorNumber,Character</u>)

express the following queries in relation algebra:

1. The titles of the films starring Henry Fonda;
2. The titles of the films in which the director is also an actor
3. The actors who have played two characters in the same film; show the title of the films, first name and surname of the actor and the two characters
4. The titles of the films in which the actors are all of the same sex

Exercise: solution 1

FILMS(<u>FilmNumber</u>,Title,Director,Year,ProductionCost)
ARTISTS(<u>ActorNumber</u>,Surname,FirsName,Sex,Birthday,Nationality)
ROLES(<u>FilmNumber,ActorNumber,Character</u>)

1. The titles of the films starring Henry Fonda;

$\pi_{Title}(FILMS \bowtie (\sigma_{(FirstName=\text{“Henry”}) \wedge (Surname=\text{“Fonda”})}(ARTISTS) \bowtie ROLES))$

Exercise: solution 2

FILMS(<u>FilmNumber</u>,Title,Director,Year,ProductionCost)
ARTISTS(<u>ActorNumber</u>,Surname,FirsName,Sex,Birthday,Nationality)
ROLES(<u>FilmNumber,ActorNumber,Character</u>)

2. The titles of the films in which the director is also an actor

$$\pi_{Title}(\sigma_{(Director=ActorNumber)}(FILMS \bowtie ROLES))$$

Exercise: solution 3

FILMS(<u>FilmNumber</u>,Title,Director,Year,ProductionCost)

ARTISTS(<u>ActorNumber</u>,Surname,FirsName,Sex,Birthday,Nationality)

ROLES(<u>FilmNumber,ActorNumber,Character</u>)

3. The actors who have played two characters in the same film; show the title of the films, first name and surname of the actor and the two characters

$$\pi_{Title,FirstName,Surname,Character1,Character}$$
$$(\rho_{FilmNumber1,ActorNumber1,Charcter1 \leftarrow FilmNumber,ActorNumber,Character}(ROLES)$$
$$\bowtie_{(FilmNumber1=FilmNumber) \wedge (ActorNumber1=ActorNumber) \wedge \ (Character1=Character)}$$
$$ROLES) \ \bowtie \ ARTISTS \bowtie FILMS)$$

Exercise: solution 4

FILMS(<u>FilmNumber</u>,Title,Director,Year,ProductionCost)

ARTISTS(<u>ActorNumber</u>,Surname,FirsName,Sex,Birthday,Nationality)

ROLES(<u>FilmNumber,ActorNumber,Character</u>)

4. The titles of the films in which the actors are all of the same sex

π_{Title}(FILMS) –

π_{Title}(FILMS)$\bowtie\sigma_{sex\neq sex1}$((ARTISTS$\bowtie$ROLES)$\bowtie$

$\rho_{FN1,AN1,Ch1,S1,FN1,Sex1,BD1,N1\leftarrow FN,AN,Ch,S,FN,Sex,BD,N}$ (ARTISTS $\bowtie$ROLES))