Politecnico di Torino

# Relational algebra

Relational model and relational algebra

0

## Relational Algebra

- ➢Introduction
- ➢Selection and projection
- ➢Cartesian product and join
- ➢Natural join, theta-join and semi-join
- ➢Outer join
- ➢Union and intersection
- ➢Difference and anti join
- ➢Division and other operators



1

# Introduction

Relation Algebra



2

## Relational Algebra

- Extends the algebra of sets for the relational model
- Defines a set of operators that operate on relations and whose output is another relation
- It satisfies the closure property
  - The result of any algebraic operation on relations is also a relation



3

## Relational algebra operators

- Unary operator
  - selection (σ)
  - projection (π)
- Binary operator
  - cartesian product (×)
  - join (⋈)
  - union (∪)
  - intersection (∩)
  - difference (-)
  - division (/)

- Set operators
  - union (∪)
  - intersection (∩)
  - difference (-)
  - cartesian product (×)
- Relational operators
  - selection (σ)
  - projection (π)
  - join (   )
  - division (/)



4

## Example of relations

Courses

| CCode | CName | Semester | TeacherID |
|---|---|---|---|
| M2170 | Computer science | 1 | D102 |
| M4880 | Digital systems | 2 | D104 |
| F1401 | Electronics | 1 | D104 |
| F0410 | Databases | 2 | D102 |

Teachers

| TeacherID | PName | Department |
|---|---|---|
| D102 | Green | Computer engineering |
| D105 | Black | Computer engineering |
| D104 | White | Department of electronics |



5

# Selection and projection

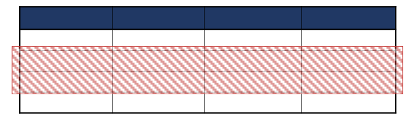
Relation Algebra



6

## Selection

- The selection extracts a "horizontal" subset from the relation
  - It operates a horizontal partition of the relation





7

## Selection: example

- *Find the courses held in the second semester*

Courses

| CCode | CName | Semester | TeacherID |
|---|---|---|---|
| M2170 | Computer science | 1 | D102 |
| M4880 | Digital systems | 2 | D104 |
| F1401 | Electronics | 1 | D104 |
| F0410 | Databases | 2 | D102 |

R

| CCode | CName | Semester | TeacherID |
|---|---|---|---|
| M4880 | Digital systems | 2 | D104 |
| F0410 | Databases | 2 | D102 |



8

## Selection: definition

$$R = \sigma_p A$$

- The selection generates a relation R
  - with the same schema as A
  - containing all the tuples of relation A for which predicate *p* is true
- Predicate *p* is a boolean expression (operators $\wedge, \vee, \neg$) combining expressions that compare attributes, or attributes and constants
  - p: City= 'Turin' $\wedge$ Age>18
  - p: ReturnDate>DeliveryDate+10



9

## Selection: example

- *Find the courses held in the second semester*

R
||
$\sigma_{Semester=2}$
|
Courses

$R = \sigma_{Semester=2}$Courses

Courses

| CCode | CName | Semester | TeacherID |
|---|---|---|---|
| M2170 | Computer science | 1 | D102 |
| M4880 | Digital systems | 2 | D104 |
| F1401 | Electronics | 1 | D104 |
| F0410 | Databases | 2 | D102 |

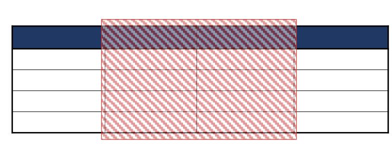
| CCode | CName | Semester | TeacherID |
|---|---|---|---|
| M4880 | Digital systems | 2 | D104 |
| F0410 | Databases | 2 | D102 |



10

## Projection

- The projection extracts a "vertical" subset from the relation
  - it operates a vertical partition of the relation





11

## Projection: example (n. 1)

- *Find the names of teachers*

Teachers

| <u>TeacherID</u> | PName | Department |
|---|---|---|
| D102 | Green | Computer engineering |
| D105 | Black | Computer engineering |
| D104 | White | Department of electronics |

R

| PName |
|---|
| Green |
| Black |
| White |



12

## Projection: definition

$$R = \pi_L A$$

- The projection $\pi_L$ generates a relation R
  - whose schema is the list of attributes L (subset of A's schema)
  - containing all of the tuples present in A
- The duplicates that may be caused by excluding the attributes not contained in L are deleted
  - if L includes a candidate key, there are no duplicates



13

## Projection: example (n. 1)

- *Find the names of teachers*

R
||
$\pi_{PName}$
|
Teachers

$R = \pi_{PName}$Teachers

Teachers

| <u>TeacherID</u> | PName | Department |
|---|---|---|
| D102 | Green | Computer engineering |
| D105 | Black | Computer engineering |
| D104 | White | Department of electronics |



14

## Projection: example (n. 2)

- *Find the names of the departments in which at least one professor is present*

R
||
$\pi_{Department}$
|
Teachers

$R = \pi_{Department}$Teachers

Teachers

| <u>TeacherID</u> | PName | Department |
|---|---|---|
| D102 | Green | Computer engineering |
| D105 | Black | Computer engineering |
| D104 | White | Department of electronics |

R

| Department |
|---|
| Computer engineering |
| Department of electronics |



15

## Selection+projection: example

- *Select the names of courses in the second semester*

Courses

| <u>CCode</u> | CName | Semester | TeacherID |
|---|---|---|---|
| M2170 | Computer science | 1 | D102 |
| M4880 | Digital systems | 2 | D104 |
| F1401 | Electronics | 1 | D104 |
| F0410 | Databases | 2 | D102 |

Selection

| CCode | CName | Semester | TeacherID |
|---|---|---|---|
| M4880 | Digital systems | 2 | D104 |
| F0410 | Databases | 2 | D102 |

Projection

R

| CName |
|---|
| Digital systems |
| Databases |



16

## Selection+projection: example

- *Select the names of courses in the second semester*

Courses

| <u>CCode</u> | CName | Semester | TeacherID |
|---|---|---|---|
| M2170 | Computer science | 1 | D102 |
| M4880 | Digital systems | 2 | D104 |
| F1401 | Electronics | 1 | D104 |
| F0410 | Databases | 2 | D102 |

R
||
$\pi_{CName}$
|
$\sigma_{Semester=2}$
|
Courses

Selection

| CCode | CName | Semester | TeacherID |
|---|---|---|---|
| M4880 | Digital systems | 2 | D104 |
| F0410 | Databases | 2 | D102 |

Projection

R

| CName |
|---|
| Digital systems |
| Databases |



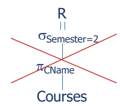
17

## Selection+projection: wrong solution

- *Select the names of courses in the second semester*

| CCode | CName | Semester | TeacherID |
|---|---|---|---|
| M2170 | Computer science | 1 | D102 |
| M4880 | Digital systems | 2 | D104 |
| F1401 | Electronics | 1 | D104 |
| F0410 | Databases | 2 | D102 |

Courses




Projection

| CName |
|---|
| Computer science |
| Digital systems |
| Electronics |
| Databases |

The Semester attribute is not available in the output relation: the selection operation cannot be carried out

DBMG 18

18

# Cartesian product and join

Relation Algebra

DBMG 19

19

## Cartesian product

➢The Cartesian product of two relations A and B generates all the pairs formed by a tuple of A and a tuple of B

DBMG 20

20

## Cartesian product: example

➢*Find the Cartesian product of courses and teachers*

DBMG 21

21

## Cartesian product: example

Courses

| CCode | CName | Semester | TeacherID |
|---|---|---|---|
| M2170 | Computer science | 1 | D102 |
| M4880 | Digital systems | 2 | D104 |
| F1401 | Electronics | 1 | D104 |
| F0410 | Databases | 2 | D102 |

Teachers

| TeacherID | PName | Department |
|---|---|---|
| D102 | Green | Computer engineering |
| D105 | Black | Computer engineering |
| D104 | White | Department of electronics |

DBMG 22

22

## Cartesian product: example

R

| Courses CCode | Courses. CName | Courses. Semester | Courses. TeacherID | Teachers. TeacherID | Teachers. Pname | Teachers. Department |
|---|---|---|---|---|---|---|
| *M2170* | *Computer science* | *1* | *D102* | *D102* | *Green* | *Computer engineering* |
| *M2170* | *Computer science* | *1* | *D102* | *D105* | *Black* | *Computer engineering* |
| *M2170* | *Computer science* | *1* | *D102* | *D104* | *White* | *Department of electronics* |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |

DBMG 23

23

## Cartesian product: example

R

| Courses CCode | Courses. CName | Courses. Semester | Courses. TeacherID | Teachers. TeacherID | Teachers. Pname | Teachers. Department |
|---|---|---|---|---|---|---|
| *M2170* | *Computer science* | *1* | *D102* | *D102* | *Green* | *Computer engineering* |
| *M2170* | *Computer science* | *1* | *D102* | *D105* | *Black* | *Icomputer engineering* |
| *M2170* | *Computer science* | *1* | *D102* | *D104* | *White* | *Department of electronics* |
| *M4880* | *Digital systems* | *2* | *D104* | *D102* | *Green* | *Computer engineering* |
| *M4880* | *Digital systems* | *2* | *D104* | *D105* | *Black* | *Icomputer engineering* |
| *M4880* | *Digital systems* | *2* | *D104* | *D104* | *White* | *Department of electronics* |
| ... | ... | ... | ... | ... | ... | ... |

24

## Cartesian product: example

R

| Courses CCode | Courses. CName | Courses. Semester | Courses. TeacherID | Teachers. TeacherID | Teachers. Pname | Teachers. Department |
|---|---|---|---|---|---|---|
| M2170 | Computer science | 1 | D102 | D102 | Green | Computer engineering |
| M2170 | Computer science | 1 | D102 | D105 | Black | Computer engineering |
| M2170 | Computer science | 1 | D102 | D104 | White | Department of electronics |
| M4880 | Digital systems | 2 | D104 | D102 | Green | Computer engineering |
| M4880 | Digital systems | 2 | D104 | D105 | Black | Computer engineering |
| M4880 | Digital systems | 2 | D104 | D104 | White | Department of electronics |
| F1401 | Electronics | 1 | D104 | D102 | Green | Computer engineering |
| F1401 | Electronics | 1 | D104 | D105 | Black | Computer engineering |
| F1401 | Electronics | 1 | D104 | D104 | White | Department of electronics |
| F0410 | Databases | 2 | D102 | D102 | Green | Computer engineering |
| F0410 | Databases | 2 | D102 | D105 | Black | Computer engineering |
| F0410 | Databases | 2 | D102 | D104 | White | Department of electronics |

25

## Cartesian product: definition

$$R = A \times B$$

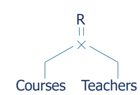
- The Cartesian product of two relations A and B yields a relation R
  - whose schema is the union of the schemas of A and B
  - containing all the pairs formed by a tuple of A and a tuple of B
- The Cartesian product is
  - commutative
    - $A \times B = B \times A$
  - associative
    - $(A \times B) \times C = A \times (B \times C)$

26

## Cartesian product: example

- *Find the Cartesian product of courses and teachers*

R = Courses × Teachers




27

## Link between attributes

R

| Courses CCode | Courses. CName | Courses. Semester | Courses. TeacherID | Teachers. TeacherID | Teachers.P name | Teachers. Department |
|---|---|---|---|---|---|---|
| *M2170* | *Computer science* | *1* | *D102* | *D102* | *Green* | *Computer engineering* |
| *M2170* | *Computer science* | *1* | *D102* | *D105* | *Black* | *Icomputer engineering* |
| *M2170* | *Computer science* | *1* | *D102* | *D104* | *White* | *Department of electronics* |
| *M4880* | *Digital systems* | *2* | *D104* | *D102* | *Green* | *Computer engineering* |
| *M4880* | *Digital systems* | *2* | *D104* | *D105* | *Black* | *Icomputer engineering* |
| *M4880* | *Digital systems* | *2* | *D104* | *D104* | *White* | *Department of electronics* |
| ... | ... | ... | ... | ... | ... | ... |

28

## Join

- The join of two relations A and B generates all the pairs formed by a tuple of A and a tuple of B that are *"semantically linked"*

29

## Join: example

- *Find information about courses and the teachers that hold them*





## Join: example

- *Find information about courses and the teachers that hold them*

Courses

| CCode | CName | Semester | TeacherID |
|---|---|---|---|
| M2170 | Computer science | 1 | D102 |
| M4880 | Digital systems | 2 | D104 |
| F1401 | Electronics | 1 | D104 |
| F0410 | Databases | 2 | D102 |

Teachers

| TeacherID | PName | Department |
|---|---|---|
| D102 | Green | Computer engineering |
| D105 | Black | Computer engineering |
| D104 | White | Department of electronics |





## Join: example

R

| Courses CCode | Courses. CName | Courses. Semester | Courses. TeacherID | Teachers. TeacherID | Teachers. Pname | Teachers. Department |
|---|---|---|---|---|---|---|
| *M2170* | *Computer science* | *1* | *D102* | *D102* | *Green* | *Computer engineering* |
| *M2170* | *Computer science* | *1* | *D102* | *D105* | *Black* | *Icomputer engineering* |
| *M2170* | *Computer science* | *1* | *D102* | *D104* | *White* | *Department of electronics* |
| *M4880* | *Digital systems* | *2* | *D104* | *D102* | *Green* | *Computer engineering* |
| *M4880* | *Digital systems* | *2* | *D104* | *D105* | *Black* | *Icomputer engineering* |
| *M4880* | *Digital systems* | *2* | *D104* | *D104* | *White* | *Department of electronics* |
| ... | ... | ... | ... | ... | ... | ... |





## Join: example

R

| Courses CCode | Courses. CName | Courses. Semester | Courses. TeacherID | Teachers. TeacherID | Teachers. Pname | Teachers. Department |
|---|---|---|---|---|---|---|
| M2170 | Computer science | 1 | D102 | D102 | Green | Computer engineering |
| M4880 | Digital systems | 2 | D104 | D104 | White | Department of electronics |
| F1401 | Electronics | 1 | D104 | D104 | White | Department of electronics |
| F0410 | Databases | 2 | D102 | D102 | Green | Computer engineering |





## Join: example

R

| Courses CCode | Courses. CName | Courses. Semester | Courses. TeacherID | Teachers. TeacherID | Teachers. Pname | Teachers. Department |
|---|---|---|---|---|---|---|
| M2170 | Computer science | 1 | D102 | D102 | Green | Computer engineering |
| M4880 | Digital systems | 2 | D104 | D104 | White | Department of electronics |
| F1401 | Electronics | 1 | D104 | D104 | White | Department of electronics |
| F0410 | Databases | 2 | D102 | D102 | Green | Computer engineering |

- *NB:* Professor (D105,Black,Computer engineering), who does not teach any courses does not appear in the result of the join





## Join: definition

- The join is a derived operator
  - it can be expressed using operators x, $\sigma_p$, $\pi_L$
- The join is defined separately as it expresses synthetically many recurrent operations in database queries
- There are different kinds of joins
  - natural join
  - theta-join (and its special case equi-join)
  - semi-join

# Natural join, theta-join and semi-join

Relation Algebra



36

## Natural join: definition and properties

$R = A \bowtie B$

- The natural join of two relations A and B generates a relation R
  - whose schema is composed of
    - the attributes which are present in A's schema and not in B's
    - the attributes present in B's schema and not in A's
    - a single copy of common attributes (with the same name in the schema of A and B)
  - containing all of the pairs made up of a tuple of A and a tuple of B for which the value of common attributes is the same
- Natural join is commutative and associative

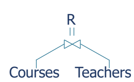


37

## Natural join: example

- *Find information about the courses and the teachers that hold them*

$R = \text{Courses} \bowtie \text{Teachers}$




R R

| Courses CCode | Courses. CName | Courses. Semester | Courses. TeacherID | Teachers. Pname | Teachers. Department |
|---|---|---|---|---|---|
| M2170 | Computer science | 1 | D102 | Green | Computer engineering |
| M4880 | Digital systems | 2 | D104 | White | Department of electronics |
| F1401 | Electronics | 1 | D104 | White | Department of electronics |
| F0410 | Databases | 2 | D102 | Green | Computer engineering |

*Note:* The common attribute TeacherID is present only once in the schema of the resulting relation R



38

## Theta-join: definition

$R = A \bowtie_p B$

- The theta-join of two relations A and B generates all the pairs formed by a tuple of A and B that satisfy a generic *"join/link condition"*
- The theta-join of two relations A and B generates a relation R
  - whose schema is the union of the schemas of A and B
  - containing all the pairs made up of a tuple of A and a tuple of B for which the predicate *p* is true
- The predicate *p* is in the form $X \theta Y$
  - X is an attribute of A, Y is an attribute of B
  - $\theta$ is a comparison operator compatible with the domains of X and of Y
- Theta-join is commutative and associative



39

## Equi-join: definition

$R = A \bowtie_p B$

- Equi-join
  - Particular case of theta-join in which $\theta$ is the equivalence operator (=)



40

## Theta-join: example

- *Find the identifiers of the teachers that hold at least two courses*

Courses C1

| CCode | CName | Semester | TeacherID |
|---|---|---|---|
| M2170 | Computer science | 1 | D102 |
| M4880 | Digital systems | 2 | D104 |
| F1401 | Electronics | 1 | D104 |
| F0410 | Databases | 2 | D102 |

Courses C2

| CCode | CName | Semester | TeacherID |
|---|---|---|---|
| M2170 | Computer science | 1 | D102 |
| M4880 | Digital systems | 2 | D104 |
| F1401 | Electronics | 1 | D104 |
| F0410 | Databases | 2 | D102 |

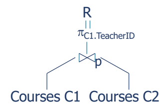


41

## Theta-join: example

- *Find the identifiers of the teachers that hold at least two courses*

$R = \pi_{C1.TeacherID}((\text{Courses } C1) \bowtie_p (\text{Courses } C2))$




p: C1.TeacherID=C2.TeacherID ∧ C1.CCode<>C2.CCode



42

## Theta-join: example

<> =

| Courses C1. CCode | Courses C1. CName | Courses C1 Semester | Courses C1. TeacherID | Courses C2. CCode | Courses C2. CName | Courses C2. Semester | Courses C2. TeacherID |
|---|---|---|---|---|---|---|---|
| M2170 | Computer science | 1 | D102 | M2170 | Computer science | 1 | D102 |
| M2170 | Computer science | 1 | D102 | M4880 | Digital systems | 2 | D104 |
| M2170 | Computer science | 1 | D102 | F1401 | Electronics | 1 | D104 |
| M2170 | Computer science | 1 | D102 | F0410 | Databases | 2 | D102 |
| M4880 | Digital systems | 2 | D104 | M2170 | Computer science | 1 | D102 |
| M4880 | Digital systems | 2 | D104 | M4880 | Digital systems | 2 | D104 |
| M4880 | Digital systems | 2 | D104 | F1401 | Electronics | 1 | D104 |
| M4880 | Digital systems | 2 | D104 | F0410 | Databases | 2 | D102 |
| F1401 | Electronics | 1 | D104 | M2170 | Computer science | 1 | D102 |
| F1401 | Electronics | 1 | D104 | M4880 | Digital systems | 2 | D104 |
| F1401 | Electronics | 1 | D104 | F1401 | Electronics | 1 | D104 |
| F1401 | Electronics | 1 | D104 | F0410 | Databases | 2 | D102 |
| F0410 | Databases | 2 | D102 | M2170 | Computer science | 1 | D102 |
| F0410 | Databases | 2 | D102 | M4880 | Digital systems | 2 | D104 |
| F0410 | Databases | 2 | D102 | F1401 | Electronics | 1 | D104 |
| F0410 | Databases | 2 | D102 | F0410 | Databases | 2 | D102 |



43

## Theta-join: example

| Courses C1. CCode | Courses C1. CName | Courses C1. Semester | Courses C1. TeacherID | Courses C2. CCode | Courses C2. CName | Courses C2. Semester | Courses C2. TeacherID |
|---|---|---|---|---|---|---|---|
| M2170 | Computer science | 1 | *D102* | F0410 | Databases | 2 | D102 |
| M4880 | Digital systems | 2 | *D104* | F1401 | Electronics | 1 | D104 |
| F1401 | Electronics | 1 | *D104* | M4880 | Digital systems | 2 | D104 |
| F0410 | Databases | 2 | *D102* | M2170 | Computer science | 1 | D102 |

Projection

R

| Courses C1. TeacherID |
|---|
| D102 |
| D104 |



44

## Semi-join: definition and properties

$R = A \ltimes_p B$

- The semi-join of two relations A and B selects all the tuples of A that are *"semantically linked"* to at least one tuple of B
  - the information from B does not appear in the result
- The semi-join of two relations A and B generates a relation R
  - which has the same schema as A
  - containing all the tuples of A for which the predicate specified by *p* is true
- The predicate *p* is expressed in the same form as the theta-join (comparison between the attributes of A and B)



45

## Semi-join: properties

- The semi-join can be expressed as a function of the theta-join
  - $A \ltimes_p B = \pi_{schema(A)}(A \bowtie_p B)$
- The semi-join *does not satisfy* the commutative property



46

## Semi-join: example

- *Find information relative to teachers that hold at least one course*

Courses

| CCode | CName | Semester | TeacherID |
|---|---|---|---|
| M2170 | Computer science | 1 | D102 |
| M4880 | Digital systems | 2 | D104 |
| F1401 | Electronics | 1 | D104 |
| F0410 | Databases | 2 | D102 |

Teachers

| TeacherID | PName | Department |
|---|---|---|
| D102 | Green | Computer engineering |
| D105 | Black | Computer engineering |
| D104 | White | Department of electronics |



47

## Semi-join: example

| Teachers. TeacherID | Teachers. Pname | Teachers. Department | Courses. CCode | Courses.CName | Courses. Semester | Courses. TeacherID |
|---|---|---|---|---|---|---|
| *D102* | *Green* | *Computer engineering* | *M2170* | *Computer science* | *1* | *D102* |
| D102 | Green | Computer engineering | M4880 | Digital systems | 2 | D104 |
| D102 | Green | Computer engineering | F1401 | Electronics | 1 | D104 |
| *D102* | *Green* | *Computer engineering* | *F0410* | *Databases* | *2* | *D102* |
| D105 | Black | Computer engineering | M2170 | Computer science | 1 | D102 |
| D105 | Black | Computer engineering | M4880 | Digital systems | 2 | D104 |
| D105 | Black | Computer engineering | F1401 | Electronics | 1 | D104 |
| *D104* | *White* | *Department of electronics* | *F1401* | *Electronics* | *1* | *D104* |
| ... | ... | ... | ... | ... | ... | ... |

48

48

## Semi-join: example

| Teachers. TeacherID | Teachers. Pname | Teachers. Department | Courses. CCode | Courses. CName | Courses. Semester | Courses. TeacherID |
|---|---|---|---|---|---|---|
| *D102* | *Green* | *Computer engineering* | M2170 | Computer science | 1 | D102 |
| *D102* | *Green* | *Computer engineering* | F0410 | Databases | 2 | D102 |
| *D104* | *White* | *Department of electronics* | M4880 | Digital systems | 2 | D104 |
| *D104* | *White* | *Electronics* | F1401 | Electronics | 3 | D104 |

R

| Teachers. TeacherID | Teachers. Pname | Teachers. Department |
|---|---|---|
| D102 | Green | Computer engineering |
| D104 | White | Department of electronics |

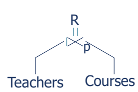
49

49

## Semi-join: example

➢ *Find information relative to teachers that hold at least one course*

R=Teachers $\ltimes_p$ Courses




p: Teachers.TeacherID= Courses.TeacherID

R

| Teachers. TeacherID | Teachers. Pname | Teachers. Department |
|---|---|---|
| D102 | Green | Computer engineering |
| D104 | White | Department of electronics |

50

50

# Outer join

Relation Algebra

51

51

## Outer-join

- Version of join that allows us to preserve the information relative to tuples that are not semantically linked by the join predicate
  - complete the tuples that lack a counterpart with null values
- There are three kinds of outer-join
  - left: only the tuples of the first operand are completed
  - right: only the tuples of the second operand are completed
  - full: the tuples of both operands are completed

52

52

## Left outer-join

- The left outer-join of two relations A and B generates all pairs of
  - a tuple of A and one of B that are *"semantically linked"*

    +
  - a tuple of A *"not semantically linked"* to any tuple of B, completed with null values for all the attributes of B

53

53

## Semi-join: example

- *Find information about teachers and about the courses that they hold*

Courses

| CCode | CName | Semester | TeacherID |
|---|---|---|---|
| M2170 | Computer science | 1 | D102 |
| M4880 | Digital systems | 2 | D104 |
| F1401 | Electronics | 1 | D104 |
| F0410 | Databases | 2 | D102 |

Teachers

| TeacherID | PName | Department |
|---|---|---|
| D102 | Green | Computer engineering |
| D105 | Black | Computer engineering |
| D104 | White | Department of electronics |

DBMG 54

54

## Left outer-join: example

R

| Teachers. TeacherID | Teachers. Pname | Teachers. Department | Courses. CCode | Courses. CName | Courses. Semester | Courses. TeacherID |
|---|---|---|---|---|---|---|
| *D102* | *Green* | *Computer engineering* | *M2170* | *Computer science* | *1* | *D102* |
| *D102* | *Green* | *Computer engineering* | *F0410* | *Databases* | *2* | *D102* |
| *D104* | *White* | *Department of electronics* | *M4880* | *Digital systems* | *2* | *D104* |
| *D104* | *White* | *Electronics* | *F1401* | *Electronics* | *3* | *D104* |

DBMG 55

55

## Left outer-join: example

R

| Teachers. TeacherID | Teachers. Pname | Teachers. Department | Courses. CCode | Courses. CName | Courses. Semester | Courses. TeacherID |
|---|---|---|---|---|---|---|
| *D102* | *Green* | *Computer engineering* | M2170 | Computer science | 1 | D102 |
| *D102* | *Green* | *Computer engineering* | F0410 | Databases | 2 | D102 |
| *D104* | *White* | *Department of electronics* | M4880 | Digital systems | 2 | D104 |
| *D104* | *White* | *Electronics* | F1401 | Electronics | 1 | D104 |
| *D105* | *Black* | *Computer engineering* | *null* | *null* | *null* | *null* |

DBMG 56

56

## Left outer-join: definition

R = A ⟕$_p$ B

- The left outer-join of two relations A and B generates a relation R
  - whose schema is the union of the schemas of A and B
  - containing the pairs made up of:
    - a tuple of A and a tuple of B for which the predicate *p* is true
    - a tuple of A that is not correlated by means of the predicate *p* to any tuple of B completed with null values for all the attributes of B

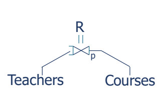
➢The left outer-join *is not* commutative

DBMG 57

57

## Left outer-join: example

- *Find information about teachers and about the courses that they hold*




R = Teachers ⟕$_p$ Courses

p: Teachers.TeacherID=Courses.TeacherID

| Teachers. TeacherID | Teachers. Pname | Teachers. Department | Courses. CCode | Courses. CName | Courses. Semester | Courses. TeacherID |
|---|---|---|---|---|---|---|
| D102 | Green | Computer engineering | M2170 | Computer science | 1 | D102 |
| D102 | Green | Computer engineering | F0410 | Databases | 2 | D102 |
| D104 | White | Department of electronics | M4880 | Digital systems | 2 | D104 |
| D104 | White | Electronics | F1401 | Electronics | 1 | D104 |
| D105 | Black | Computer engineering | null | null | null | null |

DBMG 58

58

## Right outer-join: definition

R = A ⟖$_p$ B

- The right outer-join of two relations A and B generates a relation R
  - whose schema is the union of the schemas of A and B
  - containing the pairs made up of
    - a tuple of A and a tuple of B for which the predicate *p* is true
    - a tuple of B that is not correlated by means of the predicate *p* to any tuple of A completed with null values for all the attributes of A

➢The right outer-join *is not* commutative

DBMG 59

59

## Full outer-join: definition and properties

$R = A ⟗_p B$

- The full outer-join of two relations A and B generates the relation R
  - whose schema is the union of the schemas of A and B
- containing the pairs formed by:
  - a tuple of A and a tuple of B for which predicate *p* is true
  - a tuple of A that is not correlated by means of the predicate *p* to any tuple of B completed with null values for all the attributes of B
  - a tuple of B that is not correlated by means of the predicate *p* to any tuple of A completed with null values for all the attributes of A
- The full outer-join is commutative

DBMG 60

60

# Union and intersection

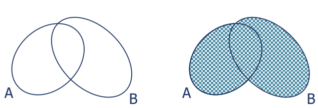
Relation Algebra

DBMG 61

61

## Union

- The union of two relations A and B selects all the tuples present in at least one of the two relations




DBMG 62

62

## Union: example

- *Find information relative to the teachers of bachelor's degree or master's degree courses*

BachelorTeachers

| TeacherID | PName | Department |
|---|---|---|
| D102 | Green | Computer engineering |
| D105 | Black | Computer engineering |
| D104 | White | Department of electronics |

MasterTeachers

| TeacherID | PName | Department |
|---|---|---|
| D102 | Green | Computer engineering |
| D101 | Rossi | Department of electrics |

DBMG 63

63

## Union: example

- *Find information relative to the teachers of bachelor's degree or master's degree courses*

BachelorTeachers

| TeacherID | PName | Department |
|---|---|---|
| D102 | Green | Computer engineering |
| D105 | Black | Computer engineering |
| D104 | White | Department of electronics |

MasterTeachers

| TeacherID | PName | Department |
|---|---|---|
| D102 | Green | Computer engineering |
| D101 | Rossi | Department of electrics |

| TeacherID | PName | Department |
|---|---|---|
| D102 | Green | Computer engineering |
| D105 | Black | Computer engineering |
| D104 | White | Department of electronics |
| D101 | Rossi | Department of electrics |

Note: Duplicate tuples are deleted

DBMG 64

64

## Union: definition and properties

$R = A \cup B$

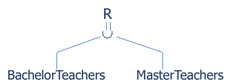
- The union of two relations A and B generates the relation R
  - which has the same schema of A and B
  - containing all the tuples belonging to A and all the tuples belonging to B (or both)
- *Compatibility*
  - the relations A and B must have the same schema (number and type of attributes)
- Duplicate tuples are eliminated
- The union is commutative and associative

DBMG 65

65

## Union: example

- *Find information relative to the teachers of bachelor's degree or master's degree courses*




R = BachelorTeachers ∪ MasterTeachers

R

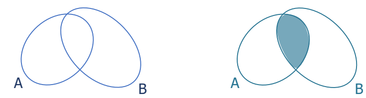
| TeacherID | PName | Department |
|---|---|---|
| D102 | Green | Computer engineering |
| D105 | Black | Computer engineering |
| D104 | White | Department of electronics |
| D101 | Rossi | Department of electrics |

66

66

## Intersection

- The intersection of two relations A and B selects all the tuples present in both relations




67

67

## Intersection: example

- *Find information relative to the teachers of both bachelor's degree and master's degree courses*

BachelorTeachers

| TeacherID | PName | Department |
|---|---|---|
| D102 | Green | Computer engineering |
| D105 | Black | Computer engineering |
| D104 | White | Department of electronics |

MasterTeachers

| TeacherID | PName | Department |
|---|---|---|
| D102 | Green | Computer engineering |
| D101 | Rossi | Department of electrics |

68

68

## Intersection: example

- *Find information relative to the teachers of both bachelor's degree and master's degree courses*

BachelorTeachers

| TeacherID | PName | Department |
|---|---|---|
| D102 | Green | Computer engineering |
| D105 | Black | Computer engineering |
| D104 | White | Department of electronics |

MasterTeachers

| TeacherID | PName | Department |
|---|---|---|
| D102 | Green | Computer engineering |
| D101 | Rossi | Department of electrics |

| TeacherID | PName | Department |
|---|---|---|
| D102 | Green | Computer engineering |

69

69

## Intersection: definition and properties

R = A ∩ B

- The intersection of two relations A and B generates a relation R
  - with the same schema of A and B
  - containing all the tuples belonging to both A and B
- *Compatibility*
  - relations A and B must have the same schema (number and type of attributes)
- Intersection is commutative and associative

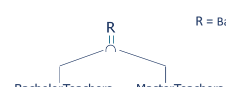
70

70

## Intersection: example

- *Find information relative to the teachers of both bachelor's degree and master's degree courses*

R = BachelorTeachers ∩ MasterTeachers




R

| TeacherID | PName | Department |
|---|---|---|
| D102 | Green | Computer engineering |

71

71

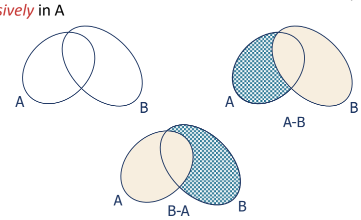
# Difference and anti-join

Relation Algebra



## Difference

- The difference of two relations A and B selects all the tuples present *exclusively* in A




A-B≠B-A



## Difference: example (n. 1)

- *Find information relative to professors who teach bachelor's degree courses, but not master's degree courses*

BachelorTeachers

| TeacherID | PName | Department |
|---|---|---|
| D102 | Green | Computer engineering |
| D105 | Black | Computer engineering |
| D104 | White | Department of electronics |

MasterTeachers

| TeacherID | PName | Department |
|---|---|---|
| D102 | Green | Computer engineering |
| D101 | Rossi | Department of electrics |

| TeacherID | PName | Department |
|---|---|---|
| D105 | Black | Computer engineering |
| D104 | White | Department of electronics |



## Difference: definition and properties

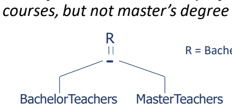
R = A - B

- The difference of two relations A and B generates a relation R
  - with the same schema of A and B
  - containing all tuples belonging to A that do not belong to B
- *Compatibility*
  - relations A and B must have the same schema (number and type of attributes)
- The difference *does not satisfy* the commutative property, nor the associative property



## Difference: example (n. 1)

- *Find information relative to professors who teach bachelor's degree courses, but not master's degree courses*




R = BachelorTeachers - MasterTeachers

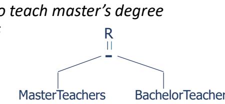
R

| TeacherID | PName | Department |
|---|---|---|
| D105 | Black | Computer engineering |
| D104 | White | Department of electronics |



## Difference: example (n. 2)

- *Find information relative to professors who teach master's degree courses, but not bachelor's degree courses*




R = MasterTeachers - BachelorTeachers

MasterTeachers

| TeacherID | PName | Department |
|---|---|---|
| D102 | Green | Computer engineering |
| D101 | Rossi | Department of electrics |

BachelorTeachers

| TeacherID | PName | Department |
|---|---|---|
| D102 | Green | Computer engineering |
| D105 | Black | Computer engineering |
| D104 | White | Department of electronics |

| TeacherID | PName | Department |
|---|---|---|
| D101 | Rossi | Department of electrics |

## Difference: example (n. 3)

- *Find identifier, name and department of teachers that are not holding any courses*

Courses

| CCode | CName | Semester | TeacherID |
|---|---|---|---|
| M2170 | Computer science | 1 | D102 |
| M4880 | Digital systems | 2 | D104 |
| F1401 | Electronics | 1 | D104 |
| F0410 | Databases | 2 | D102 |

Teachers

| TeacherID | PName | Department |
|---|---|---|
| D102 | Green | Computer engineering |
| D105 | Black | Computer engineering |
| D104 | White | Department of electronics |

78

## Difference: example (n. 3)

- *Find identifier, name and department of teachers that are not holding any courses*

Projection
Teacher identifier

Teachers

| TeacherID | PName | Department |
|---|---|---|
| *D102* | Green | Computer engineering |
| *D105* | Black | Computer engineering |
| *D104* | White | Department of electronics |

Courses

| CCode | CName | Semester | TeacherID |
|---|---|---|---|
| M2170 | Computer science | 1 | *D102* |
| M4880 | Digital systems | 2 | *D104* |
| F1401 | Electronics | 1 | *D104* |
| F0410 | Databases | 2 | *D102* |

Projection
Identifiers of teachers who hold at least one course

79

## Difference: example (n. 3)

| TeacherID |
|---|
| D102 |
| D105 |
| D104 |

| TeacherID |
|---|
| D102 |
| D104 |

Difference

| TeacherID |
|---|
| *D105* |

Natural Join

Teachers

| TeacherID | PName | Department |
|---|---|---|
| D102 | Green | Computer engineering |
| *D105* | *Black* | *Computer engineering* |
| D104 | White | Department of electronics |

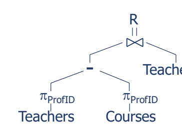
R

| TeacherID | PName | Department |
|---|---|---|
| *D105* | *Black* | *Computer engineering* |

80

## Difference: example (n. 3)

- *Find identifier, name and department of teachers that are not holding any courses*




$$R = \text{Teachers} \bowtie ((\pi_{ProfID}\text{Teachers}) - (\pi_{ProfID}\text{Courses}))$$

81

## Anti-join: definition and properties

$$R = A \triangleright_p B$$

- The anti-join of two relations A and B selects all the tuples of A that are *"not semantically linked"* to tuples of B
  - the information of B does not appear in the result
- The anti-join of two relations A and B generates a relation R
  - with the same schema of A
  - containing all the tuples of A for which there is no tuple of B for which the predicate *p* is true
- The predicate *p* is expressed in the same way as for the theta-join and the semi-join
- The anti-join *does not satisfy* the commutative property, nor the associative property

82

## Anti-join: example

- *Find identifier, name and department of teachers that are not holding any courses*

Courses

| CCode | CName | Semester | TeacherID |
|---|---|---|---|
| M2170 | Computer science | 1 | D102 |
| M4880 | Digital systems | 2 | D104 |
| F1401 | Electronics | 1 | D104 |
| F0410 | Databases | 2 | D102 |

Teachers

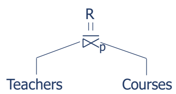
| TeacherID | PName | Department |
|---|---|---|
| D102 | Green | Computer engineering |
| D105 | Black | Computer engineering |
| D104 | White | Department of electronics |

| TeacherID | PName | Department |
|---|---|---|
| D105 | Black | Computer engineering |

83

## Anti-join: example

- *Find identifier, name and department of teachers that are not holding any courses*




R = Teachers $\overline{\ltimes}_p$ Courses

p: Teachers.TeacherID=Courses.TeacherID

R

| TeacherID | PName | Department |
|---|---|---|
| D105 | Black | Computer engineering |

84

84

## Division and other operators

Relation Algebra

85

85

## Division: example

- *Find the students that have passed the exams of all the courses in the first year*

PassedExams

| StudentID | CCode |
|---|---|
| S1 | C1 |
| S1 | C2 |
| S1 | C3 |
| S1 | C4 |
| S1 | C5 |
| S1 | C6 |
| S2 | C1 |
| S2 | C2 |
| S3 | C2 |
| S4 | C2 |
| S4 | C4 |
| S4 | C5 |

FirstYearCourses

| CCode |
|---|
| ... |
| ... |
| ... |
| ... |

86

86

## Division: example

- *Find the students that have passed the exams of all the courses in the first year*

PassedExams

| StudentID | CCode |
|---|---|
| S1 | C1 |
| S1 | C2 |
| S1 | C3 |
| S1 | C4 |
| S1 | C5 |
| S1 | C6 |
| S2 | C1 |
| S2 | C2 |
| S3 | C2 |
| S4 | C2 |
| S4 | C4 |
| S4 | C5 |

FirstYearCourses

| CCode |
|---|
| C1 |

R

| StudentID |
|---|
| S1 |
| S2 |

87

87

## Division: example

- *Find the students that have passed the exams of all the courses in the first year*

PassedExams

| StudentID | CCode |
|---|---|
| S1 | C1 |
| S1 | C2 |
| S1 | C3 |
| S1 | C4 |
| S1 | C5 |
| S1 | C6 |
| S2 | C1 |
| S2 | C2 |
| S3 | C2 |
| S4 | C2 |
| S4 | C4 |
| S4 | C5 |

FirstYearCourses

| CCode |
|---|
| C2 |
| C4 |

R

| StudentID |
|---|
| S1 |
| S4 |

88

88

## Division: example

- *Find the students that have passed the exams of all the courses in the first year*

PassedExams

| StudentID | CCode |
|---|---|
| S1 | C1 |
| S1 | C2 |
| S1 | C3 |
| S1 | C4 |
| S1 | C5 |
| S1 | C6 |
| S2 | C1 |
| S2 | C2 |
| S3 | C2 |
| S4 | C2 |
| S4 | C4 |
| S4 | C5 |

FirstYearCourses

| CCode |
|---|
| C1 |
| C2 |
| C3 |
| C4 |
| C5 |
| C6 |

R

| StudentID |
|---|
| S1 |

89

89

## Division: definition and properties

R = A / B

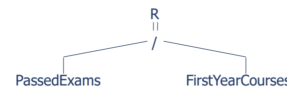
- The division of relation A by relation B generates a relation R
  - whose schema is *schema(A) - schema(B)*
  - containing all the tuples of A such that for each tuple (Y:y) present in B there is a tuple (X:x, Y:y) in A
- Division *does not satisfy* the commutative property, nor the associative property



90

## Division: example

- *Find the students that have passed the exams of all the courses in the first year*




R = PassedExams / FirstYearCourses



91

## Other operators

- Various other operators have been proposed so as to extend the expressive power of relational algebra
  - extending relations with a new attribute, defined by a scalar expression
    - GROSS_WEIGHT=NET_WEIGHT+TARE
  - calculating aggregate function
    - max, min, avg, count, sum
    - possibly defining subsets in which to group the data (GROUP BY of SQL)



92