ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG VIỆT - HÀN
Vietnam - Korea University of Information and Communication Technology

# Database Systems

## Chapter 2: *Database Design*

## session 3:
## Relational Algebra

# Outline

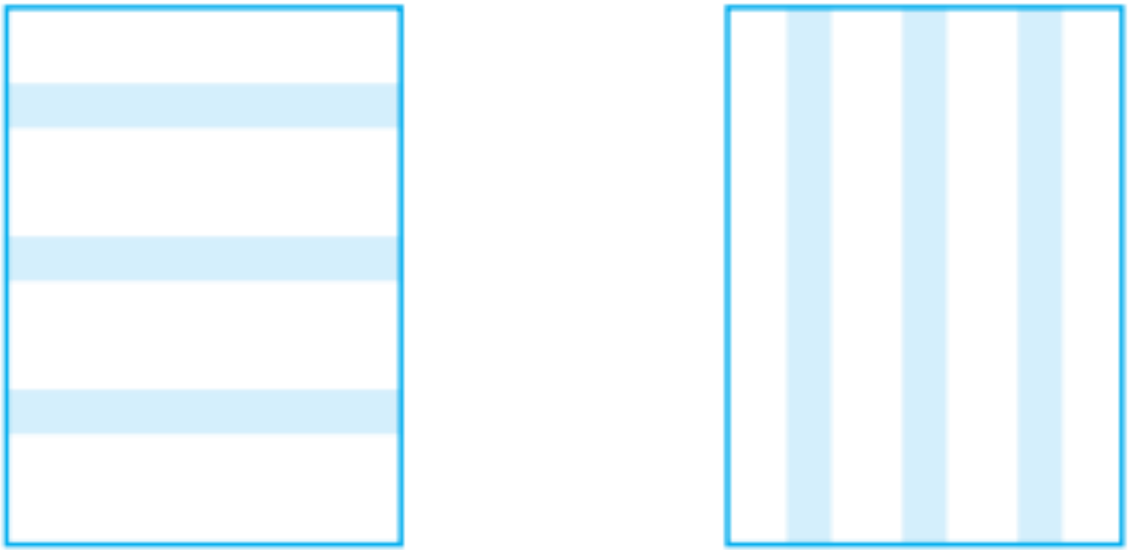
1. Relational Algebra
2. Select and Product
3. Cartesian product
4. Inner Join and Outer Join
5. Set Operations

# Relational Algebra

- The relational algebra consists of a set of operations that take one or two relations as input and produce a new relation as their result.
- Fundamental Operations
  - Unary Operation: operate on one relation
    - Select $\sigma$
    - Project $\prod$
    - Rename $\rho$
  - Binary Operations: operate on pairs of relations
    - Union $\cup$ , Intersection $\cap$
    - Set different –
    - Cartesian product **x**
    - Join operations

# Selection and Project

- Unary operation



(a) Selection (b) Projection

# Select Operation

- The **select** operation selects tuples that satisfy a given predicate.
- Notation: $\sigma_p(r)$
- $p$ is called the **selection predicate**

# Select Operation

**Example**: select those tuples of the *instructor* relation where the instructor is in the "Physics" department.

**Instructor** relation

| *ID* | *name* | *dept_name* | *salary* |
|---|---|---|---|
| 22222 | Einstein | Physics | 95000 |
| 12121 | Wu | Finance | 90000 |
| 32343 | El Said | History | 60000 |
| 45565 | Katz | Comp. Sci. | 75000 |
| 98345 | Kim | Elec. Eng. | 80000 |
| 76766 | Crick | Biology | 72000 |
| 10101 | Srinivasan | Comp. Sci. | 65000 |
| 58583 | Califieri | History | 62000 |
| 83821 | Brandt | Comp. Sci. | 92000 |
| 15151 | Mozart | Music | 40000 |
| 33456 | Gold | Physics | 87000 |
| 76543 | Singh | Finance | 80000 |

- Query

$$\sigma_{\textit{dept\_name}=\text{"Physics"}}(\textit{instructor})$$

- Result

| *ID* | *name* | *dept_name* | *salary* |
|---|---|---|---|
| 22222 | Einstein | Physics | 95000 |
| 33456 | Gold | Physics | 87000 |

# Select Operation

- We allow comparisons using =, ≠, <, ≤, >, and ≥ in the selection predicate.
- we can combine several predicates into a larger predicate by using the connectives *and* ( ∧ ), *or* ( ∨ ), and *not* (¬)

# Select Operation

- Example: Find the instructors in Physics with a salary greater $90,000, we write:
  - Query:

$$\sigma_{dept\_name=\text{“Physics”} \wedge salary>90000}(instructor)$$

  - Result:

| *ID* | *name* | *dept_name* | *salary* |
|---|---|---|---|
| 22222 | Einstein | Physics | 95000 |

# Project Operation

- Project operation selects certain *columns* from the table and discards the other columns.
- Notation:

$$\prod_{A_1, A_2, A_3 \dots . A_k} (r)$$

where $A_1, A_2, \dots, A_k$ are attribute names and $r$ is a relation name.

- The result is defined as the relation of $k$ columns obtained by erasing the columns that are not listed
- Duplicate rows removed from result, since relations are sets

# Project Operation

- Example: eliminate the *dept_name* attribute of *instructor* relation
  - Query

$$\Pi_{ID,\ name,\ salary}\ (instructor)$$

| *ID* | *name* | *dept_name* | *salary* |
|---|---|---|---|
| 10101 | Srinivasan | Comp. Sci. | 65000 |
| 12121 | Wu | Finance | 90000 |
| 15151 | Mozart | Music | 40000 |
| 22222 | Einstein | Physics | 95000 |
| 32343 | El Said | History | 60000 |
| 33456 | Gold | Physics | 87000 |
| 45565 | Katz | Comp. Sci. | 75000 |
| 58583 | Califieri | History | 62000 |
| 76543 | Singh | Finance | 80000 |
| 76766 | Crick | Biology | 72000 |
| 83821 | Brandt | Comp. Sci. | 92000 |
| 98345 | Kim | Elec. Eng. | 80000 |

  - Result:

| *ID* | *name* | *salary* |
|---|---|---|
| 10101 | Srinivasan | 65000 |
| 12121 | Wu | 90000 |
| 15151 | Mozart | 40000 |
| 22222 | Einstein | 95000 |
| 32343 | El Said | 60000 |
| 33456 | Gold | 87000 |
| 45565 | Katz | 75000 |
| 58583 | Califieri | 62000 |
| 76543 | Singh | 80000 |
| 76766 | Crick | 72000 |
| 83821 | Brandt | 92000 |
| 98345 | Kim | 80000 |

# Composition of Relational Operations

- The result of a relational-algebra operation is relation and therefore of relational-algebra operations can be composed together into a **relational-algebra expression**.
- Consider the query -- Find the names of all instructors in the Physics department.

$$\prod_{name}(\sigma_{dept\_name\ =\text{“}Physics\text{”}}\ (instructor))$$

- Instead of giving the name of a relation as the argument of the projection operation, we give an expression that evaluates to a relation.

# Cartesian-Product Operation

- The Cartesian-product operation (denoted by X) allows us to combine information from any two relations.
- We write the Cartesian product of relations *P* and *Q* as *P* × *Q*

*P*

| *A* |
|---|
| *a* |
| *b* |

*Q*

| *B* |
|---|
| 1 |
| 2 |
| 3 |

=

*P* × *Q*

| *A* | *B* |
|---|---|
| *a* | 1 |
| *a* | 2 |
| *a* | 3 |
| *b* | 1 |
| *b* | 2 |
| *b* | 3 |

(c) Cartesian product

# Cartesian-Product Operation

- Example: the Cartesian product of the relations *instructor* and *teaches* is written as:

*instructor* X *teaches*

| *ID* | *name* | *dept_name* | *salary* |
|---|---|---|---|
| 10101 | Srinivasan | Comp. Sci. | 65000 |
| 12121 | Wu | Finance | 90000 |
| 15151 | Mozart | Music | 40000 |
| 22222 | Einstein | Physics | 95000 |
| 32343 | El Said | History | 60000 |
| 33456 | Gold | Physics | 87000 |
| 45565 | Katz | Comp. Sci. | 75000 |
| 58583 | Califieri | History | 62000 |
| 76543 | Singh | Finance | 80000 |
| 76766 | Crick | Biology | 72000 |
| 83821 | Brandt | Comp. Sci. | 92000 |
| 98345 | Kim | Elec. Eng. | 80000 |

**instructor**

| *ID* | *course_id* | *sec_id* | *semester* | *year* |
|---|---|---|---|---|
| 10101 | CS-101 | 1 | Fall | 2017 |
| 10101 | CS-315 | 1 | Spring | 2018 |
| 10101 | CS-347 | 1 | Fall | 2017 |
| 12121 | FIN-201 | 1 | Spring | 2018 |
| 15151 | MU-199 | 1 | Spring | 2018 |
| 22222 | PHY-101 | 1 | Fall | 2017 |
| 32343 | HIS-351 | 1 | Spring | 2018 |
| 45565 | CS-101 | 1 | Spring | 2018 |
| 45565 | CS-319 | 1 | Spring | 2018 |
| 76766 | BIO-101 | 1 | Summer | 2017 |
| 76766 | BIO-301 | 1 | Summer | 2018 |
| 83821 | CS-190 | 1 | Spring | 2017 |
| 83821 | CS-190 | 2 | Spring | 2017 |
| 83821 | CS-319 | 2 | Spring | 2018 |
| 98345 | EE-181 | 1 | Spring | 2017 |

**teaches**

# Cartesian-Product Operation

- Result:

| *instructor.ID* | *name* | *dept_name* | *salary* | *teaches.ID* | *course_id* | *sec_id* | *semester* | *year* |
|---|---|---|---|---|---|---|---|---|
| 10101 | Srinivasan | Comp. Sci. | 65000 | 10101 | CS-101 | 1 | Fall | 2017 |
| 10101 | Srinivasan | Comp. Sci. | 65000 | 10101 | CS-315 | 1 | Spring | 2018 |
| 10101 | Srinivasan | Comp. Sci. | 65000 | 10101 | CS-347 | 1 | Fall | 2017 |
| 10101 | Srinivasan | Comp. Sci. | 65000 | 12121 | FIN-201 | 1 | Spring | 2018 |
| 10101 | Srinivasan | Comp. Sci. | 65000 | 15151 | MU-199 | 1 | Spring | 2018 |
| 10101 | Srinivasan | Comp. Sci. | 65000 | 22222 | PHY-101 | 1 | Fall | 2017 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 12121 | Wu | Finance | 90000 | 10101 | CS-101 | 1 | Fall | 2017 |
| 12121 | Wu | Finance | 90000 | 10101 | CS-315 | 1 | Spring | 2018 |
| 12121 | Wu | Finance | 90000 | 10101 | CS-347 | 1 | Fall | 2017 |
| 12121 | Wu | Finance | 90000 | 12121 | FIN-201 | 1 | Spring | 2018 |
| 12121 | Wu | Finance | 90000 | 15151 | MU-199 | 1 | Spring | 2018 |
| 12121 | Wu | Finance | 90000 | 22222 | PHY-101 | 1 | Fall | 2017 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 15151 | Mozart | Music | 40000 | 10101 | CS-101 | 1 | Fall | 2017 |
| 15151 | Mozart | Music | 40000 | 10101 | CS-315 | 1 | Spring | 2018 |
| 15151 | Mozart | Music | 40000 | 10101 | CS-347 | 1 | Fall | 2017 |
| 15151 | Mozart | Music | 40000 | 12121 | FIN-201 | 1 | Spring | 2018 |
| 15151 | Mozart | Music | 40000 | 15151 | MU-199 | 1 | Spring | 2018 |
| 15151 | Mozart | Music | 40000 | 22222 | PHY-101 | 1 | Fall | 2017 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 22222 | Einstein | Physics | 95000 | 10101 | CS-101 | 1 | Fall | 2017 |
| 22222 | Einstein | Physics | 95000 | 10101 | CS-315 | 1 | Spring | 2018 |
| 22222 | Einstein | Physics | 95000 | 10101 | CS-347 | 1 | Fall | 2017 |
| 22222 | Einstein | Physics | 95000 | 12121 | FIN-201 | 1 | Spring | 2018 |
| 22222 | Einstein | Physics | 95000 | 15151 | MU-199 | 1 | Spring | 2018 |
| 22222 | Einstein | Physics | 95000 | 22222 | PHY-101 | 1 | Fall | 2017 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |

# Cartesian-Product Operation

- We construct a tuple of the result out of each possible pair of tuples: one from the instructor relation and one from the teaches relation.
- Since the instructor ID appears in both relations we distinguish between these attribute by attaching to the attribute the name of the relation from which the attribute originally came.
  - instructor.ID
  - teaches.ID
- Assume that we have $n_1$ tuples in *instructor* and $n_2$ tuples in *teaches*, so there are $n_1 * n_2$ tuples in *r*

# Join Operation

- The Cartesian-Product

  *instructor* X *teaches*

associates every tuple of instructor with every tuple of teaches.

- Most of the resulting rows have information about instructors who did NOT teach a particular course.

# Join Operation

- To get only those tuples of "*instructor* X *teaches*" that pertain to instructors and the courses that they taught, we write:

$$\sigma_{\,instructor.id\ =\ teaches.id}\ (instructor \times teaches\ ))$$

  - We get only those tuples of "*instructor* X *teaches*" that pertain to instructors and the courses that they taught.

- The result of this expression, shown in the next slide

# Join Operation

- The table corresponding to:

$$\sigma_{instructor.id = teaches.id} (instructor \times teaches))$$

| *instructor.ID* | *name* | *dept_name* | *salary* | *teaches.ID* | *course_id* | *sec_id* | *semester* | *year* |
|---|---|---|---|---|---|---|---|---|
| 10101 | Srinivasan | Comp. Sci. | 65000 | 10101 | CS-101 | 1 | Fall | 2017 |
| 10101 | Srinivasan | Comp. Sci. | 65000 | 10101 | CS-315 | 1 | Spring | 2018 |
| 10101 | Srinivasan | Comp. Sci. | 65000 | 10101 | CS-347 | 1 | Fall | 2017 |
| 12121 | Wu | Finance | 90000 | 12121 | FIN-201 | 1 | Spring | 2018 |
| 15151 | Mozart | Music | 40000 | 15151 | MU-199 | 1 | Spring | 2018 |
| 22222 | Einstein | Physics | 95000 | 22222 | PHY-101 | 1 | Fall | 2017 |
| 32343 | El Said | History | 60000 | 32343 | HIS-351 | 1 | Spring | 2018 |
| 45565 | Katz | Comp. Sci. | 75000 | 45565 | CS-101 | 1 | Spring | 2018 |
| 45565 | Katz | Comp. Sci. | 75000 | 45565 | CS-319 | 1 | Spring | 2018 |
| 76766 | Crick | Biology | 72000 | 76766 | BIO-101 | 1 | Summer | 2017 |
| 76766 | Crick | Biology | 72000 | 76766 | BIO-301 | 1 | Summer | 2018 |
| 83821 | Brandt | Comp. Sci. | 92000 | 83821 | CS-190 | 1 | Spring | 2017 |
| 83821 | Brandt | Comp. Sci. | 92000 | 83821 | CS-190 | 2 | Spring | 2017 |
| 83821 | Brandt | Comp. Sci. | 92000 | 83821 | CS-319 | 2 | Spring | 2018 |
| 98345 | Kim | Elec. Eng. | 80000 | 98345 | EE-181 | 1 | Spring | 2017 |

# Join Operation

- The **join** operation allows us to combine a select operation and a Cartesian-Product operation into a single operation.
- It is denoted by a $\bowtie$
- The general form of a join operation on two relations $R(A_1, A_2, \ldots, A_n)$ and $S(B_1, B_2, \ldots, B_m)$ is:

$$R \bowtie_{\text{<join condition>}} S$$

where R and S can be any relations that result from general *relational algebra expressions*.

# Join Operation

- Thus $\sigma_{instructor.id = teaches.id}(instructor \times teaches))$

Can equivalently be written as

$$instructor \bowtie_{Instructor.id = teaches.id} teaches.$$

- This operation is very important for any relational database with more than a single relation, because it allows us to process relationships among relations.

# Join operator

- Example: Suppose that we want to retrieve the name of the manager of each department
  - To get the manager's name, we need to combine each DEPARTMENT tuple with the EMPLOYEE tuple whose SSN value matches the MGRSSN value in the department tuple.
  - We do this by using the join ⋈ operation.

$$DEPT\_MGR \leftarrow DEPARTMENT \bowtie_{MGRSSN=SSN} EMPLOYEE$$

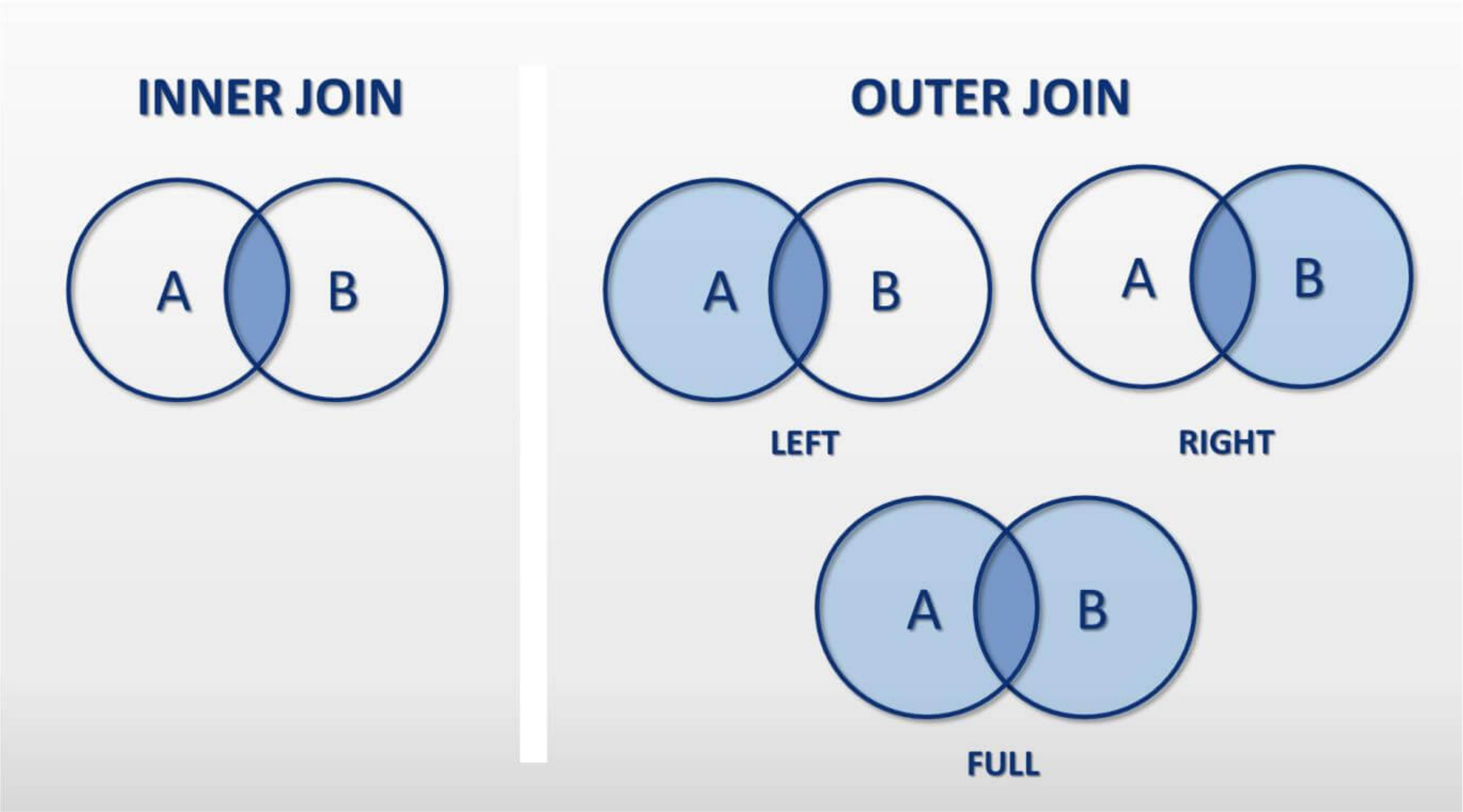
**DEPT_MGR**

| Dname | Dnumber | Mgr_ssn | . . . | Fname | Minit | Lname | Ssn | . . . |
|---|---|---|---|---|---|---|---|---|
| Research | 5 | 333445555 | . . . | Franklin | T | Wong | 333445555 | . . . |
| Administration | 4 | 987654321 | . . . | Jennifer | S | Wallace | 987654321 | . . . |
| Headquarters | 1 | 888665555 | . . . | James | E | Borg | 888665555 | . . . |

# Outer Joins

- In Inner Join, we matched rows are returned and unmatched rows are not returned.
- But, in outer join, we include those tuples which meet the given condition along with that, we also add those tuples which do not meet the required condition.
- There are three types of outer joins:
  - Left outer join ⟕
  - Right outer join ⟖
  - Full outer join ⟗

# Outer Joins



# Left Outer Join

Left Outer Join : A $⟕_{\text{<join condition>}}$ B

- Ensures that all tuples in the in the relation A are present in the result set.
- The tuples in A without matching tuples in B are filled with *null* values for B's attributes

# Left Outer Join example

Students

| *stud#* | *name* | *course* |
|---|---|---|
| 100 | Fred | PH |
| 200 | Dave | CM |
| 400 | Peter | EN |

Courses

| *course#* | *name* |
|---|---|
| PH | Pharmacy |
| CM | Computing |
| CH | Chemistry |

Students ⟕ <course = course#> Courses

Result:

| *stud#* | *Students.name* | *course* | *course#* | *Courses.name* |
|---|---|---|---|---|
| 100 | Fred | PH | PH | Pharmacy |
| 200 | Dave | CM | CM | Computing |
| 400 | Peter | EN | NULL | NULL |

# Right Outer Join

Right Outer Join: A ⟖$_{\langle\text{join condition}\rangle}$ B

- ❑Reverse of left outer join.
- ❑Retrieves all tuples of B and null values for attributes of A in non-matching tuples of B

⟗

# Right Outer Join example

Students

| stud# | name | course |
|---|---|---|
| 100 | Fred | PH |
| 200 | Dave | CM |
| 400 | Peter | EN |

Courses

| course# | name |
|---|---|
| PH | Pharmacy |
| CM | Computing |
| CH | Chemistry |

Students ⟖ <course = course#> Courses

Result:

| stud# | Students.name | course | course# | Courses.name |
|---|---|---|---|---|
| 100 | Fred | PH | PH | Pharmacy |
| 200 | Dave | CM | CM | Computing |
| NULL | NULL | NULL | CH | Chemistry |

# Full Outer Join

Full Outer Join: $A ⟗_{\text{<join condition>}} B$

- Ensures that all tuples of A and B are present in the result set

# Full Outer Join Example

Students

| *stud#* | *name* | *course* |
|---|---|---|
| 100 | Fred | PH |
| 200 | Dave | CM |
| 400 | Peter | EN |

Courses

| *course#* | *name* |
|---|---|
| PH | Pharmacy |
| CM | Computing |
| CH | Chemistry |

Students $⟗$ <course = course#> Courses

Result:

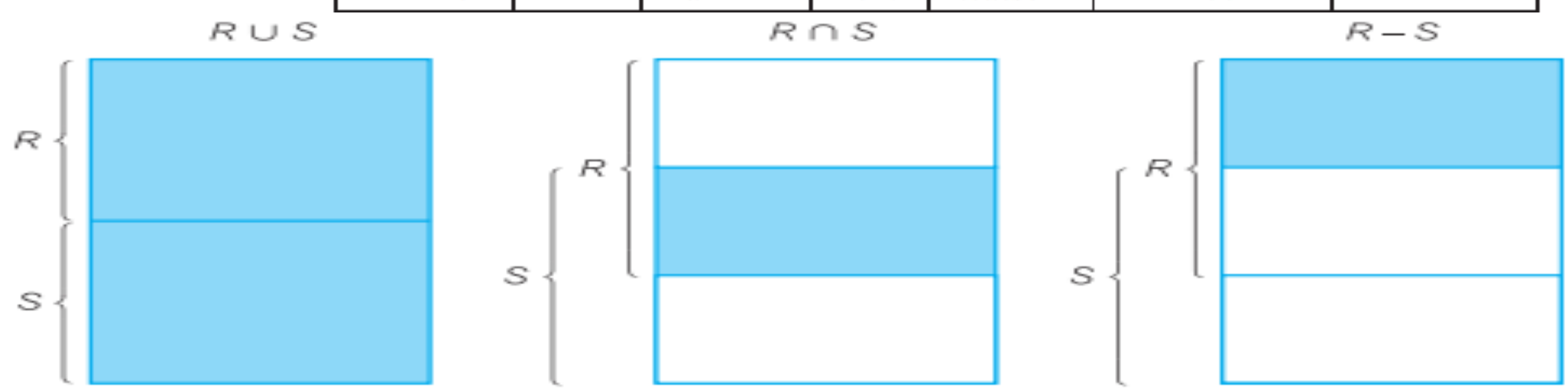
| *stud#* | *Students.name* | *course* | *course#* | *Courses.name* |
|---|---|---|---|---|
| 100 | Fred | PH | PH | Pharmacy |
| 200 | Dave | CM | CM | Computing |
| 400 | Peter | CN | NULL | NULL |
| NULL | NULL | NULL | CH | Chemistry |

# Set operations

- ❑Example

**Section** relation

| course_id | sec_id | semester | year | building | room_number | time_slot_id |
|---|---|---|---|---|---|---|
| BIO-101 | 1 | Summer | 2017 | Painter | 514 | B |
| BIO-301 | 1 | Summer | 2018 | Painter | 514 | A |
| CS-101 | 1 | Fall | 2017 | Packard | 101 | H |
| CS-101 | 1 | Spring | 2018 | Packard | 101 | F |
| CS-190 | 1 | Spring | 2017 | Taylor | 3128 | E |
| CS-190 | 2 | Spring | 2017 | Taylor | 3128 | A |
| CS-315 | 1 | Spring | 2018 | Watson | 120 | D |
| CS-319 | 1 | Spring | 2018 | Watson | 100 | B |
| CS-319 | 2 | Spring | 2018 | Taylor | 3128 | C |
| CS-347 | 1 | Fall | 2017 | Taylor | 3128 | A |
| EE-181 | 1 | Spring | 2017 | Taylor | 3128 | C |
| FIN-201 | 1 | Spring | 2018 | Packard | 101 | B |
| HIS-351 | 1 | Spring | 2018 | Painter | 514 | C |
| MU-199 | 1 | Spring | 2018 | Packard | 101 | D |
| PHY-101 | 1 | Fall | 2017 | Watson | 100 | A |




(d) Union (e) Intersection (f) Set difference

# Union Operation

- The union operation allows us to combine two relations
- Notation: $r \cup s$
- For $r \cup s$ to be valid.

1. *r, s* must have the *same* **arity** (same number of attributes)
2. The attribute domains must be **compatible** (example: 2nd column of *r* deals with the same type of values as does the 2nd column of *s*)

# Union Operation

- Example: to find all courses taught in the Fall 2017 semester, or in the Spring 2018 semester, or in both

$$\Pi_{course\_id}\,(\sigma_{semester=\text{"Fall"}\ \land\ year=2017}\,(section))\ \cup$$

$$\Pi_{course\_id}\,(\sigma_{semester=\text{"Spring"}\ \land\ year=2018}\,(section))$$

- The result of this expression, shown in the next slide

# Union Operation

**section** relation

| *course_id* | *sec_id* | *semester* | *year* | *building* | *room_number* | *time_slot_id* |
|---|---|---|---|---|---|---|
| BIO-101 | 1 | Summer | 2017 | Painter | 514 | B |
| BIO-301 | 1 | Summer | 2018 | Painter | 514 | A |
| CS-101 | 1 | Fall | 2017 | Packard | 101 | H |
| CS-101 | 1 | Spring | 2018 | Packard | 101 | F |
| CS-190 | 1 | Spring | 2017 | Taylor | 3128 | E |
| CS-190 | 2 | Spring | 2017 | Taylor | 3128 | A |
| CS-315 | 1 | Spring | 2018 | Watson | 120 | D |
| CS-319 | 1 | Spring | 2018 | Watson | 100 | B |
| CS-319 | 2 | Spring | 2018 | Taylor | 3128 | C |
| CS-347 | 1 | Fall | 2017 | Taylor | 3128 | A |
| EE-181 | 1 | Spring | 2017 | Taylor | 3128 | C |
| FIN-201 | 1 | Spring | 2018 | Packard | 101 | B |
| HIS-351 | 1 | Spring | 2018 | Painter | 514 | C |
| MU-199 | 1 | Spring | 2018 | Packard | 101 | D |
| PHY-101 | 1 | Fall | 2017 | Watson | 100 | A |

result

| *course_id* |
|---|
| CS-101 |
| CS-315 |
| CS-319 |
| CS-347 |
| FIN-201 |
| HIS-351 |
| MU-199 |
| PHY-101 |

# Set-Intersection Operation

- The set-intersection or intersection operation allows us to find tuples that are in both the input relations.
- Notation: $r \cap s$
- Assume:
  - *r*, *s* have the *same arity*
  - attributes of *r* and *s* are compatible

# Set-Intersection Operation

- Example: Find the set of all courses taught in both the Fall 2017 and the Spring 2018 semesters.

$$\Pi_{course\_id}\left(\sigma_{semester=\text{“Fall”} \ \wedge\ year=2017}(section)\right) \cap \Pi_{course\_id}\left(\sigma_{semester=\text{“Spring”} \ \wedge\ year=2018}(section)\right)$$

- Result:

| *course_id* |
| --- |
| CS-101 |

# Set Difference Operation

- The set-difference operation allows us to find tuples that are in one relation but are not in another.

- Notation $r - s$
- Set differences must be taken between **compatible** relations.
  - *r* and *s* must have the same arity
  - attribute domains of *r* and *s* must be compatible

# Set Difference Operation

- Example: to find all courses taught in the Fall 2017 semester, but not in the Spring 2018 semester

$$\prod_{course\_id} (\sigma_{semester=\text{“Fall”} \ \wedge \ year=2017} (section)) \ - \prod_{course\_id} (\sigma_{semester=\text{“Spring”} \ \wedge \ year=2018} (section))$$

- Result:

| *course_id* |
|---|
| CS-347 |
| PHY-101 |

# The Rename Operation

- The results of relational-algebra expressions do not have a name that we can use to refer to them. The rename operator, $\rho$, is provided for that purpose
- The expression:

$$\rho_x (E)$$

returns the result of expression $E$ under the name $x$

- Another form of the rename operation:

$$\rho_{x(A1,A2, .. An)} (E)$$

# Equivalent Queries

- There is more than one way to write a query in relational algebra.
- Example: Find information about courses taught by instructors in the Physics department with salary greater than 90,000
- Query 1

$$\sigma_{dept\_name=\text{“Physics”} \wedge salary > 90{,}000}(instructor)$$

- Query 2

$$\sigma_{dept\_name=\text{“Physics”}}(\sigma_{salary > 90.000}(instructor))$$

The two queries are not identical; they are, however, equivalent -- they give the same result on any database.

# Equivalent Queries

- Example: Find information about courses taught by instructors in the Physics department
- Query 1

$$\sigma_{dept\_name=\text{“Physics”}}(instructor \bowtie_{instructor.ID = teaches.ID} teaches)$$

- Query 2

$$(\sigma_{dept\_name=\text{“Physics”}}(instructor)) \bowtie_{instructor.ID = teaches.ID} teaches$$

The two queries are not identical; they are, however, equivalent -- they give the same result on any database.

# Exercises

- Ex1:

**Example: Customer**

| c_no | title | sname | inits | street | city | postc | cred_lim | balance |
|---|---|---|---|---|---|---|---|---|
| 1 | Mrs | Sallaway | G.R. | 12 Fax Rd | London | WC1 | £1,000.00 | £16.26 |
| 2 | Miss | Lauri | P. | 5 Dux St | London | N1 | £500.00 | £200.00 |
| 3 | Mr | Jackson | R. | 2 Lux Ave | Leeds | LE1 2AB | £500.00 | £510.00 |
| 4 | Mr | Dziduch | M. | 31 Low St | Dover | DO2 9CD | £100.00 | £149.23 |
| 5 | Ms | Woods | S.Q. | 17 Nax Rd | London | E18 4WW | £1,000.00 | £350.10 |
| 6 | Mrs | Williams | C. | 41 Cax St | Dover | DO2 8WD | | £412.21 |

- *Query 1:* List customers whose cred_lim is greater than £500.
- *Query 2:* List customers whose cred_lim is greater than £500 and lives in London.

# Exercises

- Ex2

*Reserves*

| sid | bid | day |
|---|---|---|
| 22 | 101 | 10/10/11 |
| 58 | 103 | 11/12/11 |

| sid | sname | rating | age |
|---|---|---|---|
| 22 | Jesly | 7 | 45.0 |
| 31 | Mishail | 8 | 55.5 |
| 58 | Raj | 10 | 35.0 |

*sailors*

| bid | bname | color |
|---|---|---|
| 101 | Interlake | Blue |
| 102 | Interlake | Red |
| 103 | Clipper | Green |
| 104 | Marine | Red |

*Boats*

1. Find names of sailors who've reserved boat #103
2. Find names of sailors who've reserved a red boat
3. Find sailors who've reserved a red or a green boat
4. Find sailors who've reserved a red <u>and</u> a green boat
5. Find the names of sailors who've reserved all boats

# Thank You !