

Relational algebra – select (1)

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

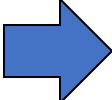
Example: select those tuples of the *instructor* relation where the instructor is in the “Physics” department.

Query:

$\sigma_{dept_name = \text{“Physics”}}(instructor)$

Result:

<i>ID</i>	<i>name</i>	<i>dept_name</i>	<i>salary</i>
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



<i>ID</i>	<i>name</i>	<i>dept_name</i>	<i>salary</i>
22222	Einstein	Physics	95000
33456	Gold	Physics	87000

Relational algebra – select (2)

- We allow comparisons using

$=, \neq, >, \geq, <, \leq$

in the **selection predicate**.

- We can combine several predicates into a larger predicate by using the connectives:

\wedge (**and**), \vee (**or**), \neg (**not**)

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

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

- Then select predicate may include comparisons between two attributes.

Example, find all departments whose name is the same as their building name:

$\sigma_{dept_name = building} (department)$

Relational algebra – project

- A unary operation that returns its argument relation, with **certain attributes left out**.
- Notation:

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

where A_1, A_2 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

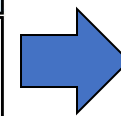
Example: eliminate the *dept_name* attribute of *instructor*

Query:

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

Result:

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



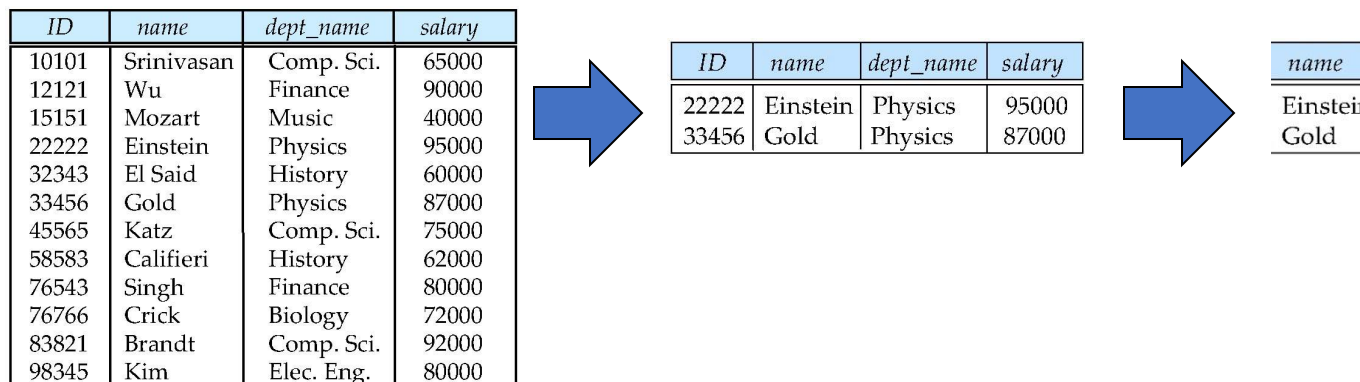
ID	name	salary
10101	Srinivasan	65000
12121	Wu	90000
15151	Mozart	40000
22222	Einstein	95000
32343	El Said	60000
33456	Gold	87000
45565	Katz	75000
58583	Califieri	62000
76543	Singh	80000
76766	Crick	72000
83821	Brandt	92000
98345	Kim	80000

Relational algebra - composition of relational operations

- The result of a relational-algebra operation is relation and therefore 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.

$$\Pi_{name}(\sigma_{dept_name = "Physics"}(instructor))$$

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



Relational algebra - union

- The **union operation** allows us to combine two relations (union of sets)
- Notation: $r \cup s$
- Output the union of tuples from the two input relations
- 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)

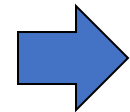
Example: to find all courses taught in the Fall 2009 semester, or in the Spring 2010 semester, or in both

Query:

$$\Pi_{course_id}(\sigma_{semester="Fall" \wedge year=2009}(section)) \cup$$
$$\Pi_{course_id}(\sigma_{semester="Spring" \wedge year=2010}(section))$$

Result:

course_id	sec_id	semester	year	building	room_number	time_slot_id
BIO-101	1	Summer	2009	Painter	514	B
BIO-301	1	Summer	2010	Painter	514	A
CS-101	1	Fall	2009	Packard	101	H
CS-101	1	Spring	2010	Packard	101	F
CS-190	1	Spring	2009	Taylor	3128	E
CS-190	2	Spring	2009	Taylor	3128	A
CS-315	1	Spring	2010	Watson	120	D
CS-319	1	Spring	2010	Watson	100	B
CS-319	2	Spring	2010	Taylor	3128	C
CS-347	1	Fall	2009	Taylor	3128	A
EE-181	1	Spring	2009	Taylor	3128	C
FIN-201	1	Spring	2010	Packard	101	B
HIS-351	1	Spring	2010	Painter	514	C
MU-199	1	Spring	2010	Packard	101	D
PHY-101	1	Fall	2009	Watson	100	A



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

Relational algebra – set difference

- The **set-difference** operation allows us to find tuples that are in one relation but are not in another.
- Notation: $r - s$
- Produce a relation containing those tuples in r but not in 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**

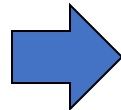
Example: to find all courses taught in the Fall 2009 semester, but not in the Spring 2010 semester

Query:

$$\Pi_{course_id} (\sigma_{semester="Fall" \wedge year=2009}(section)) - \Pi_{course_id} (\sigma_{semester="Spring" \wedge year=2010}(section))$$

Result:

course_id	sec_id	semester	year	building	room_number	time_slot_id
BIO-101	1	Summer	2009	Painter	514	B
BIO-301	1	Summer	2010	Painter	514	A
CS-101	1	Fall	2009	Packard	101	H
CS-101	1	Spring	2010	Packard	101	F
CS-190	1	Spring	2009	Taylor	3128	E
CS-190	2	Spring	2009	Taylor	3128	A
CS-315	1	Spring	2010	Watson	120	D
CS-319	1	Spring	2010	Watson	100	B
CS-319	2	Spring	2010	Taylor	3128	C
CS-347	1	Fall	2009	Taylor	3128	A
EE-181	1	Spring	2009	Taylor	3128	C
FIN-201	1	Spring	2010	Packard	101	B
HIS-351	1	Spring	2010	Painter	514	C
MU-199	1	Spring	2010	Packard	101	D
PHY-101	1	Fall	2009	Watson	100	A



course_id
CS-347
PHY-101

Relational algebra – Cartesian-product (1)

- The **Cartesian-product operation** (denoted by **X**) allows us to combine information from any two relations.
- Notation: **$r \times s$**
- Output all pairs of rows from the two input relations (regardless of whether or not they have the same values on common attributes)
- Assume that attributes of $r(R)$ and $s(S)$ are disjoint. (That is, $R \cap S = \emptyset$).
- If attributes of $r(R)$ and $s(S)$ are not disjoint, then renaming must be used.

Relational algebra – Cartesian-product (2)

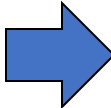
- Example: the Cartesian product of the relations *instructor* and *teaches* is written as:

instructor X *teaches*

- 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*

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

ID	course_id	sec_id	semester	year
10101	CS-101	1	Fall	2009
10101	CS-315	1	Spring	2010
10101	CS-347	1	Fall	2009
12121	FIN-201	1	Spring	2010
15151	MU-199	1	Spring	2010
22222	PHY-101	1	Fall	2009
32343	HIS-351	1	Spring	2010
45565	CS-101	1	Spring	2010
45565	CS-319	1	Spring	2010
76766	BIO-101	1	Summer	2009
76766	BIO-301	1	Summer	2010
83821	CS-190	1	Spring	2009
83821	CS-190	2	Spring	2009
83821	CS-319	2	Spring	2010
98345	EE-181	1	Spring	2009



<i>instructor.ID</i>	name	dept_name	salary	<i>teaches.ID</i>	course_id	sec_id	semester	year
10101	Srinivasan	Comp. Sci.	65000	10101	CS-101	1	Fall	2009
10101	Srinivasan	Comp. Sci.	65000	10101	CS-315	1	Spring	2010
10101	Srinivasan	Comp. Sci.	65000	10101	CS-347	1	Fall	2009
10101	Srinivasan	Comp. Sci.	65000	12121	FIN-201	1	Spring	2010
10101	Srinivasan	Comp. Sci.	65000	15151	MU-199	1	Spring	2010
10101	Srinivasan	Comp. Sci.	65000	22222	PHY-101	1	Fall	2009
...
12121	Wu	Finance	90000	10101	CS-101	1	Fall	2009
12121	Wu	Finance	90000	10101	CS-315	1	Spring	2010
12121	Wu	Finance	90000	10101	CS-347	1	Fall	2009
12121	Wu	Finance	90000	12121	FIN-201	1	Spring	2010
12121	Wu	Finance	90000	15151	MU-199	1	Spring	2010
12121	Wu	Finance	90000	22222	PHY-101	1	Fall	2009
...
15151	Mozart	Music	40000	10101	CS-101	1	Fall	2009
15151	Mozart	Music	40000	10101	CS-315	1	Spring	2010
15151	Mozart	Music	40000	10101	CS-347	1	Fall	2009
15151	Mozart	Music	40000	12121	FIN-201	1	Spring	2010
15151	Mozart	Music	40000	15151	MU-199	1	Spring	2010
15151	Mozart	Music	40000	22222	PHY-101	1	Fall	2009
...
22222	Einstein	Physics	95000	10101	CS-101	1	Fall	2009
22222	Einstein	Physics	95000	10101	CS-315	1	Spring	2010
22222	Einstein	Physics	95000	10101	CS-347	1	Fall	2009
22222	Einstein	Physics	95000	12121	FIN-201	1	Spring	2010
22222	Einstein	Physics	95000	15151	MU-199	1	Spring	2010
22222	Einstein	Physics	95000	22222	PHY-101	1	Fall	2009
...

Relational algebra – query examples

- Find the names of all instructors in the Physics department, along with the *course_id* of all courses they have taught

Query 1

$\Pi_{name, course_id} (\sigma_{instructor.ID=teaches.ID} (\sigma_{dept_name="Physics"} (instructor \times teaches)))$

- There is often more than one way to write a query in relational algebra
- The following two queries are equivalent, i.e., they give the same result

Query 1

$\Pi_{name, course_id} (\sigma_{instructor.ID=teaches.ID} (\sigma_{dept_name="Physics"} (instructor \times teaches)))$

Query 2

$\Pi_{name, course_id} (\sigma_{instructor.ID=teaches.ID} (\sigma_{dept_name="Physics"} (instructor) \times teaches))$

inst.ID	name	dept_name	salary	teaches.ID	course_id	sec_id	semester	year
22222	Einstein	Physics	95000	10101	CS-437	1	Fall	2009
22222	Einstein	Physics	95000	10101	CS-315	1	Spring	2010
22222	Einstein	Physics	95000	12121	FIN-201	1	Spring	2010
22222	Einstein	Physics	95000	15151	MU-199	1	Spring	2010
22222	Einstein	Physics	95000	22222	PHY-101	1	Fall	2009
22222	Einstein	Physics	95000	32343	HIS-351	1	Spring	2010
...
...
33456	Gold	Physics	87000	10101	CS-437	1	Fall	2009
33456	Gold	Physics	87000	10101	CS-315	1	Spring	2010
33456	Gold	Physics	87000	12121	FIN-201	1	Spring	2010
33456	Gold	Physics	87000	15151	MU-199	1	Spring	2010
33456	Gold	Physics	87000	22222	PHY-101	1	Fall	2009
33456	Gold	Physics	87000	32343	HIS-351	1	Spring	2010
...
...



name	course_id
Einstein	PHY-101

Query optimization

Query optimization: the process of choosing a suitable execution strategy for processing a query.

A query, e.g., a SQL, first be scanned, parsed and validated

- The scanner identifies the query tokens, e.g., the keywords, attribute names and relation names
- The parser checks the query syntax
- The validation checks that all attributes and relation names are valid

Relational algebra – rename

- Allows us to name, and therefore to refer to, the results of relational-algebra expressions. Allows us to refer to a relation by more than one name.
- The expression $\rho_x(E)$ returns the result of expression E under the name X
- If a relational-algebra expression E has arity n , then $\rho_{x(A_1, A_2, \dots, A_n)}(E)$ returns the result of expression E under the name X , and with the attributes renamed to A_1, A_2, \dots, A_n .

Example: Find the highest salary in the university

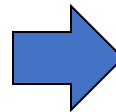
Step 1: find instructor salaries that are less than some other instructor salary (i.e. not the highest). Using a copy of *instructor* under a new name d

$\Pi_{instructor.salary}(\sigma_{instructor.salary < d.salary}(instructor \times \rho_d(instructor)))$

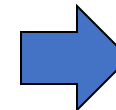
Step 2: Find the highest salary

$\Pi_{salary}(instructor) - \Pi_{instructor.salary}(\sigma_{instructor.salary < d.salary}(instructor \times \rho_d(instructor)))$

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



salary
65000
90000
40000
60000
87000
75000
62000
72000
80000
92000



salary
95000

Relational algebra – additional operations (1)

Set-Intersection

- The set-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
- Note: $r \cap s = r - (r - s)$

Relational algebra – additional operations (2)

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

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

- The **join** operation allows us to combine a **select operation** and a **Cartesian-product operation** into a single operation.
- Consider relations $r(R)$ and $s(S)$. Let “theta” be a predicate on attributes in the schema $R \text{ “union” } S$. The join operation $r \bowtie_{\theta} s$ is defined as follows:

$$r \bowtie_{\theta} s = \sigma_{\theta}(r \times s)$$

- Thus

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

can equivalently be written as: $instructor \bowtie_{instructor.id = teaches.id} teaches$.

Relational algebra – additional operations (3)

Join Operation

- The join operation without predicate is called **natural join**.
- Notation: $r \bowtie s$
- Output pairs of rows from the two input relations that have the **same value** on all attributes that have the **same name**
- Let r and s be relations on schemas R and S respectively.
Then, $r \bowtie s$ is a relation on schema $R \cup S$ obtained as follows:
 - Consider each pair of tuples t_r from r and t_s from s .
 - If t_r and t_s have the same value on each of the attributes in $R \cap S$, add a tuple t to the result, where
 - t has the same value as t_r on r
 - t has the same value as t_s on s

Relational algebra – additional operations (4)

Assignment Operation

- It is convenient at times to write a relational-algebra expression **by assigning parts of it to temporary relation variables**.

- The assignment operation is denoted by \leftarrow and works like assignment in a programming language.

- Example: Find all instructor in the “Physics” and Music department.

$Physics \leftarrow \sigma_{dept_name = \text{“Physics”}}(instructor)$

$Music \leftarrow \sigma_{dept_name = \text{“Music”}}(instructor)$

$Physics \cup Music$

- With the assignment operation, a query can be written as a sequential program consisting of
 - a series of assignments
 - followed by an expression whose value is displayed as a result of the query.

Relational algebra – additional operations (5)

Outer Join Operation

- An extension of the join operation that **avoids loss of information**.
- Computes the join and then **adds tuples** from one relation **that does not match** tuples in the other relation to the result of the join.
- Uses *null* value to signify that the value is unknown or does not exist
- Three forms of outer join:
 - left outer join
 - right outer join
 - full outer join

Relational algebra – additional operations (6)

Outer Join Operation

- Relation *course*

<i>course_id</i>	<i>title</i>	<i>dept_name</i>	<i>credits</i>
BIO-301	Genetics	Biology	4
CS-190	Game Design	Comp. Sci.	4
CS-315	Robotics	Comp. Sci.	3

- Relation *prereq*

<i>course_id</i>	<i>prereq_id</i>
BIO-301	BIO-101
CS-190	CS-101
CS-347	CS-101

- Observe that
 - CS-315 is missing in *prereq* and
 - CS-347 is missing in *course*

Relational algebra – additional operations (7)

Outer Join Operation

- Join

- $course \bowtie prereq$

<i>course_id</i>	<i>title</i>	<i>dept_name</i>	<i>credits</i>	<i>prereq_id</i>
BIO-301	Genetics	Biology	4	BIO-101
CS-190	Game Design	Comp. Sci.	4	CS-101

- Left Outer Join

- $course \ltimes prereq$

<i>course_id</i>	<i>title</i>	<i>dept_name</i>	<i>credits</i>	<i>prereq_id</i>
BIO-301	Genetics	Biology	4	BIO-101
CS-190	Game Design	Comp. Sci.	4	CS-101
CS-315	Robotics	Comp. Sci.	3	<i>null</i>

Outer join can be expressed using basic operations

e.g. $r \ltimes s$ can be written as

$$(r \bowtie s) \cup (r - \Pi_R(r \bowtie s)) \times \{(null, ..., null)\}$$

where the constant relation $\{(null, ..., null)\}$ is on the schema $S - R$

- Right Outer Join

- $course \bowtie\!\!\!\bowtie prereq$

<i>course_id</i>	<i>title</i>	<i>dept_name</i>	<i>credits</i>	<i>prereq_id</i>
BIO-301	Genetics	Biology	4	BIO-101
CS-190	Game Design	Comp. Sci.	4	CS-101
CS-347	<i>null</i>	<i>null</i>	<i>null</i>	CS-101

- Full Outer Join

- $course \ltimes\!\!\!\ltimes prereq$

<i>course_id</i>	<i>title</i>	<i>dept_name</i>	<i>credits</i>	<i>prereq_id</i>
BIO-301	Genetics	Biology	4	BIO-101
CS-190	Game Design	Comp. Sci.	4	CS-101
CS-315	Robotics	Comp. Sci.	3	<i>null</i>
CS-347	<i>null</i>	<i>null</i>	<i>null</i>	CS-101