#### **Formal Relational Query Languages**

#### Relational Algebra

- Procedural language
- Six basic operators
  - □ select: σ
  - □ project: ∏
  - □ union: ∪
  - set difference: –
  - Cartesian product: x
  - $\square$  rename:  $\rho$
- The operators take one or two relations as inputs and produce a new relation as a result.

# **Select Operation – Example**

Relation r

A	В	C	D
α	α	1	7
α	β	5	7
β	β	12	3
β	β	23	10

$$\bullet$$
  $\sigma_{A=B \land D > 5}(r)$ 

A	В	C	D
α	α	1	7
β	β	23	10

#### **Select Operation**

- □ Notation:  $\sigma_p(r)$
- □ *p* is called the **selection predicate**
- Defined as:

$$\sigma_p(\mathbf{r}) = \{t \mid t \in r \text{ and } p(t)\}$$

Where p is a formula in propositional calculus consisting of **terms** connected by :  $\land$  (**and**),  $\lor$  (**or**),  $\neg$  (**not**) Each **term** is one of:

<attribute> op <attribute> or <constant>

where *op* is one of: =,  $\neq$ , >,  $\geq$ . <.  $\leq$ 

Example of selection:

## **Project Operation – Example**

□ Relation *r*.

A	В	C
α	10	1
$\alpha$	20	1
β	30	1
β	40	2

$$\square$$
  $\prod_{A,C} (r)$ 

A	C		A	C
α	1	60	α	1
α	1		β	1
β	1		β	2
β	2			

#### **Project Operation**

Notation:

$$\prod_{A_1,A_2,\ldots,A_k}(r)$$

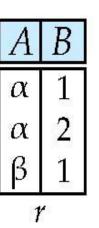
where  $A_1$ ,  $A_2$  are attribute names and r is a relation name.

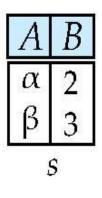
- ☐ The result is defined as the relation of *k* columns obtained by erasing the columns that are not listed
- Duplicate rows removed from result, since relations are sets
- Example: To eliminate the dept\_name attribute of instructor

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

## **Union Operation – Example**

□ Relations *r*, *s*:





 $\square$   $r \cup s$ :

$$\begin{array}{c|c} A & B \\ \hline \alpha & 1 \\ \alpha & 2 \\ \beta & 1 \\ \beta & 3 \\ \end{array}$$

#### **Union Operation**

- □ Notation:  $r \cup s$
- Defined as:

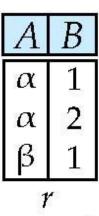
$$r \cup s = \{t \mid t \in r \text{ or } t \in s\}$$

- $\square$  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:  $2^{nd}$  column of r deals with the same type of values as does the  $2^{nd}$  column of s)
- Example: to find all courses taught in the Fall 2009 semester, or in the
   Spring 2010 semester, or in both

$$\Pi_{course\_id}(\sigma_{semester="Fall"} \land year=2009(section)) \cup \Pi_{course\_id}(\sigma_{semester="Spring"} \land year=2010(section))$$

#### Set difference of two relations

□ Relations *r*, *s*:



)
3

 $\Box$  r-s

A	В
α	1
β	1

#### **Set Difference Operation**

- □ Notation r s
- Defined as:

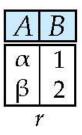
$$r-s = \{t \mid t \in r \text{ and } t \notin s\}$$

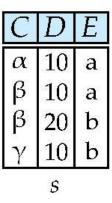
- Set differences must be taken between compatible relations.
  - r and s must have the same arity
  - attribute domains of *r* and *s* must be compatible
- Example: to find all courses taught in the Fall 2009 semester, but not in the Spring 2010 semester

$$\Pi_{course\_id}(\sigma_{semester="Fall"} \land year=2009(section)) - \Pi_{course\_id}(\sigma_{semester="Spring"} \land year=2010(section))$$

#### **Cartesian-Product Operation – Example**

□ Relations *r*, *s*:





 $\square$   $r \times s$ :

A	В	C	D	Ε
α	1	α	10	a
α	1	β	10	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

#### **Cartesian-Product Operation**

- □ Notation *r* x s
- Defined as:

$$r \times s = \{t \mid q \mid t \in r \text{ and } q \in s\}$$

- 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.

## **Composition of Operations**

- Can build expressions using multiple operations
- □ Example:  $\sigma_{A=C}(r x s)$

	r	X	S
--	---	---	---

A	В	C	D	Ε
α	1	α	10	a
α	1	β	10	a
α	1	β	20	b
α	1	γ	10	b
β	2	$\alpha$	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

$$\Box$$
  $\sigma_{A=C}(r \times s)$ 

A	В	C	D	Ε
α	1	$\alpha$	10	a
β	2	β	10	a
β	2	β	20	b

#### **Rename Operation**

- Allows us to name, and therefore to refer to, the results of relationalalgebra expressions.
- Allows us to refer to a relation by more than one name.
- Example:

$$\rho_X(E)$$

returns the expression *E* under the name *X* 

□ If a relational-algebra expression E has arity n, then

$$\rho_{x(A_1,A_2,...,A_n)}(E)$$

returns the result of expression E under the name X, and with the attributes renamed to  $A_1, A_2, \ldots, A_n$ .

#### **Example Query**

- Find the largest salary in the university
  - Step 1: find instructor salaries that are less than some other instructor salary (i.e. not maximum)
    - using a copy of instructor under a new name d
    - $\Pi_{instructor.salary}$  ( $\sigma_{instructor.salary} < d_{instructor}$ )) (instructor  $\times \rho_{d}$  (instructor)))
  - Step 2: Find the largest salary
    - $\Pi_{salary} \ (instructor) \\ \Pi_{instructor.salary} \ (\sigma_{instructor.salary} < d, salary \\ (instructor x \ \rho_d \ (instructor)))$

#### **Example Queries**

- ☐ Find the names of all instructors in the Physics department, along with the course\_id of all courses they have taught
  - Query 1

```
\prod_{instructor.ID,course\_id} (\sigma_{dept\_name="Physics"} (\sigma_{instructor.ID=teaches.ID} (instructor x teaches)))
```

Query 2

```
\prod_{instructor.ID, course\_id} (\sigma_{instructor.ID=teaches.ID} (\sigma_{dept\_name="Physics"} (instructor) \times teaches))
```

#### **Formal Definition**

- □ A basic expression in the relational algebra consists of either one of the following:
  - A relation in the database
  - A constant relation
- Let  $E_1$  and  $E_2$  be relational-algebra expressions; the following are all relational-algebra expressions:
  - $\Box$   $E_1 \cup E_2$
  - $E_1 E_2$
  - $E_1 \times E_2$
  - $\sigma_p(E_1)$ , P is a predicate on attributes in  $E_1$
  - $\square$   $\prod_{S}(E_1)$ , S is a list consisting of some of the attributes in  $E_1$
  - $\rho_X(E_1)$ , x is the new name for the result of  $E_1$

#### **Additional Operations**

We define additional operations that do not add any power to the relational algebra, but that simplify common queries.

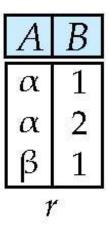
- Set intersection
- Natural join
- Assignment
- Outer join

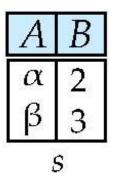
#### **Set-Intersection Operation**

- □ Notation:  $r \cap s$
- Defined as:
- Assume:
  - □ *r*, *s* have the *same arity*
  - attributes of r and s are compatible

#### **Set-Intersection Operation – Example**

□ Relation *r*, *s*:





 $\square$   $r \cap s$ 

### **Natural-Join Operation**

- □ Notation: r ⋈s
- 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<sub>r</sub> on r
    - t has the same value as  $t_S$  on s
- Example:

$$R = (A, B, C, D)$$

$$S = (E, B, D)$$

- □ Result schema = (A, B, C, D, E)
- $\ \ \ r\bowtie s$  is defined as:

$$\prod_{r.A, r.B, r.C, r.D, s.E} (\sigma_{r.B = s.B \land r.D = s.D} (r \times s))$$

## **Natural Join Example**

□ Relations r, s:

A	В	C	D
α	1	α	a
β	2	γ	a
γ	4	β	b
α	1	γ	a
δ	2	β	b

В	D	E
1	a	α
3	a	β
1	a	γ
2	b	δ
3	b	3
	s	

□r⋈s

A	В	C	D	E
α	1	α	a	α
α	1	α	a	γ
α	1	γ	a	α
α	1	γ	a	γ
δ	2	β	b	δ

#### **Natural Join and Theta Join**

- ☐ Find the names of all instructors in the Comp. Sci. department together with the course titles of all the courses that the instructors teach
  - $\ \ \square \ \ \ \ \prod_{name, \ title} (\sigma_{dept\_name="Comp. \ Sci."} \ (instructor \bowtie teaches \bowtie course))$
- Natural join is associative
  - □ (instructor  $\bowtie$  teaches)  $\bowtie$  course is equivalent to instructor  $\bowtie$  (teaches  $\bowtie$  course)
- Natural join is commutative
  - instruct ⋈ teaches is equivalent to teaches ⋈ instructor
- □ The **theta join** operation  $r \bowtie_{\theta} s$  is defined as
  - $r \bowtie_{\theta} s = \sigma_{\theta} (r \times s)$

#### **Assignment Operation**

- □ The assignment operation  $(\leftarrow)$  provides a convenient way to express complex queries.
  - Write query as a sequential program consisting of
    - a series of assignments
    - followed by an expression whose value is displayed as a result of the query.
  - Assignment must always be made to a temporary relation variable.

#### **Outer Join**

- An extension of the join operation that avoids loss of information.
- Computes the join and then adds tuples form one relation that does not match tuples in the other relation to the result of the join.
- □ Uses *null* values:
  - null signifies that the value is unknown or does not exist
  - All comparisons involving *null* are (roughly speaking) **false** by definition.
    - We shall study precise meaning of comparisons with nulls later

## **Outer Join – Example**

Relation instructor1

ID	name	dept_name
10101	Srinivasan	Comp. Sci.
12121	Wu	Finance
15151	Mozart	Music

□ Relation *teaches1* 

ID	course_id	
10101	CS-101	
12121	FIN-201	
76766	BIO-101	

## **Outer Join – Example**

Join

*instructor* ⋈ *teaches* 

ID	name	dept_name	course_id
10101	Srinivasan	Comp. Sci.	CS-101
12121	Wu	Finance	FIN-201

#### Left Outer Join

instructor \textcolor teaches

ID	name	dept_name	course_id
10101	Srinivasan	Comp. Sci.	CS-101
12121	Wu	Finance	FIN-201
15151	Mozart	Music	null

### **Outer Join – Example**

#### □ Right Outer Join

instructor ⋈ teaches

ID	name	dept_name	course_id
10101	Srinivasan	Comp. Sci.	CS-101
12121	Wu	Finance	FIN-201
76766	null	null	BIO-101

#### □ Full Outer Join

instructor □ teaches

ID	name	dept_name	course_id
10101	Srinivasan	Comp. Sci.	CS-101
12121	Wu	Finance	FIN-201
15151	Mozart	Music	null
76766	null	null	BIO-101

#### **Outer Join using Joins**

- Outer join can be expressed using basic operations
  - □ e.g. r | s can be written as

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

#### **Null Values**

- It is possible for tuples to have a null value, denoted by *null*, for some of their attributes
- null signifies an unknown value or that a value does not exist.
- ☐ The result of any arithmetic expression involving *null* is *null*.
- Aggregate functions simply ignore null values (as in SQL)
- For duplicate elimination and grouping, null is treated like any other value, and two nulls are assumed to be the same (as in SQL)

#### **Null Values**

- Comparisons with null values return the special truth value: unknown
  - If *false* was used instead of *unknown*, then not (A < 5) would not be equivalent to A >= 5
- ☐ Three-valued logic using the truth value *unknown*:
  - OR: (unknown or true) = true,(unknown or false) = unknown(unknown or unknown) = unknown
  - AND: (true and unknown) = unknown,(false and unknown) = false,(unknown and unknown) = unknown
  - □ NOT: (**not** *unknown*) = *unknown*
  - In SQL "P is unknown" evaluates to true if predicate P evaluates to unknown
- Result of select predicate is treated as false if it evaluates to unknown

#### **Division Operator**

Given relations r(R) and s(S), such that  $S \subset R$ ,  $r \div s$  is the largest relation t(R-S) such that

$$t \times s \subseteq r$$

- E.g. let  $r(ID, course\_id) = \prod_{ID, course\_id} (takes)$  and  $s(course\_id) = \prod_{course\_id} (\sigma_{dept\_name="Biology"}(course)$  then  $r \div s$  gives us students who have taken all courses in the Biology department
- $\square$  Can write  $r \div s$  as

$$temp1 \leftarrow \prod_{R-S} (r)$$
  
 $temp2 \leftarrow \prod_{R-S} ((temp1 \times s) - \prod_{R-S,S} (r))$   
 $result = temp1 - temp2$ 

- □ The result to the right of the ← is assigned to the relation variable on the left of the ←.
- May use variable in subsequent expressions.