

#### **Relational Algebra**

MSCI346 Lukasz Golab

Acknowledgement: slides derived from material provided by Sliberschatz, Korth and Sudarshan, copyright 2019, <a href="www.db-book.com">www.db-book.com</a>, and by Prof. Wojciech Golab, ECE356



#### **Learning Outcomes**

- Develop a working knowledge of relational algebra:
  - basic operators
    - select, project, union, difference, Cartesian product, rename
  - additional operators
    - intersection, assignment, natural join
  - formula syntax
- Understand null values and 3-valued logic

Textbook sections (6<sup>th</sup> ed.): 6.1, 6.2, 6.3



#### Relational Algebra

- Procedural language (as opposed to declarative).
- Six basic operators:
  - select: σ
  - project: ∏
  - union: ∪
  - set difference: –
  - Cartesian product: x
  - rename: ρ
- These operators take one or two relations as inputs and output a single relation.



# **Select Operation – Example**

 $\blacksquare$  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 logic 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

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

A	C	A	C
α	1	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.

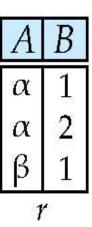
- The result is defined as the relation of *k* columns obtained by erasing the columns that are not listed.
- Note: since relations are sets, duplicate rows "removed" from result.
- Example: To eliminate the dept\_name attribute of instructor

∏ <sub>ID, name, salary</sub> (instructor)



# **Union Operation – Example**

Relations r, s:



A	В
α	2
β	3

ightharpoonup  $r \cup s$ :



#### **Union Operation**

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

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

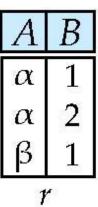
- For  $r \cup s$  to be valid.
  - 1. *r*, *s* must have the *same* **arity** (same number of attributes)
  - 2. The attribute domains must be **compatible**(Example: 2<sup>nd</sup> column of *r* deals with the same type of values as does the 2<sup>nd</sup> column of *s*. Column names may be different.)
- 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*:



$$egin{array}{c|c} A & B \\ \hline $lpha$ & 2 \\ $eta$ & 3 \\ \hline $s$ & \end{array}$$

r - s:

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



## **Set Difference Operation**

- Notation r s
- Defined as:

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

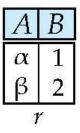
- Set differences must be taken between compatible relations, just like unions.
- Example: to find all courses taught in the Fall 2009 semester, but not in the Spring 2010 semester

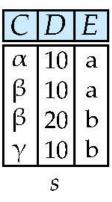
$$\Pi_{course\_id}(\sigma_{semester="Fall"} \land_{year=2009}(section)) - \Pi_{course\_id}(\sigma_{semester="Spring"} \land_{year=2010}(section))$$



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

Relations *r, s*:





**r** x s:

A	В	C	D	E
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:

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

Note: "t q" denotes a tuple obtained by concatenating together t and q.

- Note: the definition assumes 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**:  $\sigma_{A=C}(r x s)$
- rxs

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

 $\bullet$   $\sigma_{A=C}(r x s)$ 

A	В	C	D	E
α	1	α	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
    - $\Pi_{instructor.salary}$   $(\sigma_{instructor.salary} < d.salary < (instructor x <math>\rho_d$  (instructor)))
  - Step 2: find the largest salary

```
 \Pi_{salary} \ (instructor) - \\ \Pi_{instructor.salary} \ (\sigma_{instructor.salary} < d.salary \\ (instructor x \ \rho_d \ (instructor)))
```



## Formal Definition of Relational Algebra

- A basic expression in the relational algebra consists of either a relation in the database (e.g., instructor) or a constant relation (e.g., {(1, Einstein), (2, Crick)}).
- A general **relational algebra expression** is either a basic expression or an expression constructed recursively using one of the following rules, where  $E_1$  and  $E_2$  denote existing relational-algebra expressions:
  - $E_1 \cup E_2$
  - $E_1 E_2$
  - $E_1 \times E_2$
  - $\sigma_p(E_1)$ , where P is a predicate on attributes in  $E_1$
  - $\prod_{S}(E_1)$ , where S is a list comprising a subset of the attributes in  $E_1$
  - $\rho_{X(A_1, A_2, ..., A_n)}(E_1)$ , where  $x(A_1, A_2, ..., A_n)$  is the new name for  $E_1$  and its attributes



#### **Additional Operations**

For convenience, additional relational operators can be defined that do not add any expressive power to the relational algebra but simplify common queries.

- set intersection: ○
- natural join: ⋈
- theta join:  $\bowtie_{\theta}$
- assignment: ←
- set division: ÷
- outer join (see textbook, covered in a later lecture on SQL)



## **Set-Intersection Operation**

- Notation:  $r \cap s$
- Defined as:

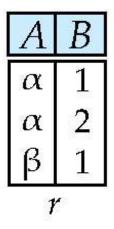
$$r \cap s = \{ t \mid t \in r \text{ and } t \in s \}$$

- Assume:
  - r, s have the same arity
  - attributes of r and s are compatible
- Note:  $r \cap s = r (r s)$



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

Relation *r*, *s*:



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

 $r \cap s$ 



#### **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.
- Example: find the largest salary in the university (in two lines of "code")

```
temp \leftarrow \Pi_{instructor.salary} (\sigma_{instructor.salary} < d.salary (instructor x \rho_d (instructor)))
\Pi_{salary} (instructor) – temp
```



#### **Natural-Join Operation**

- Notation:  $r \bowtie s$
- 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  $t_r$  for attributes in R
    - t has the same value as  $t_s$  for attributes in 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*:

1	α	a
2	γ	a
4	β	b
1	γ	a
2	β	b
	1 2 4 1 2	2 γ

D	Ε
a	α
a	β
a	γ
b	δ
b	3
	a a b

 $r \bowtie s$ 

A	В	C	D	Ε
α	1	$\alpha$	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
  - $\prod_{name, title} (\sigma_{dept\_name="Comp. Sci."} (instructor \bowtie teaches \bowtie course))$
- Natural join is associative
  - (instructor ⋈ teaches) ⋈ course is equivalent to instructor ⋈ (teaches ⋈ course)
- The **theta join** operation  $r \bowtie_{\theta} s$  is defined as
  - $r \bowtie_{\theta} s = \sigma_{\theta} (r \times s)$
  - Example: *instructor* ⋈ *instructor.ID* = *teaches.ID teaches*

(How is this different from *instructor* ⋈ *teaches* ?)



#### **Natural Join and Theta Join (Cont.)**

Example schema:

```
pet (<u>p_id</u>, name, species)
owner (<u>o_id</u>, name, address)
owns (<u>p_id</u>, o_id)
```

- Query: find the name and address of every dog that has an owner.
- Answer:

```
temp \leftarrow (pet \bowtie owns)
temp2 \leftarrow temp \bowtie_{temp.o\_id} = owner.o\_id owner
\prod_{temp.name, address} (O_{species} = "dog" (temp2))
```



#### **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

$$t \times s \subset r$$

- Example:
  - 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
- **Can rewrite**  $r \div s$  as

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



#### **Division Operator (Cont.)**

- Example with real data:
  - student IDs are 1, 2, 3, 4
  - course IDs are 'BIO-101', 'BIO-301', and 'CS-101'
  - $r = \{ (1, 'BIO-101'), (1, 'BIO-301'), (2, 'CS-301') \}$
  - s = { ('BIO-101'), ('BIO-301') }
  - $r \div s = \{ (1) \}$
- The three-line query for  $r \div s$  can be executed as follws:

```
temp1 = \prod_{R-S}(r) = \{ (1), (2) \}
temp1 \times s = \{ (1, 'BIO-101'), (1, 'BIO-301'), (2, 'BIO-101'), (2, 'BIO-301') \}
(temp1 \times s) - r = \{ (2, 'BIO-101'), (2, 'BIO-301') \}
\prod_{R-S}( (temp1 \times s) - r) = \{ (2) \}
temp1 - temp2 = \{ (1) \}
```



#### **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 (not covered in this lecture) simply ignore null values (as in SQL).
- For duplicate elimination and grouping (not covered in this lecture), null is treated like any other value, and two null values are assumed to be the same (as in SQL).