Chapter 2: Intro to Relational Model

Edited by Radhika Sukapuram. Original slides by Database System Concepts, 6th Ed.

©Silberschatz, Korth and Sudarshan



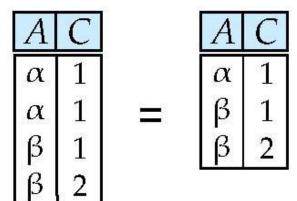
Project Operation – selection of columns (Attributes)

PROJECT Operation: This operation selects certain *columns* from the table and discards the other columns. The PROJECT creates a vertical partitioning — one with the needed columns (attributes) containing results of the operation and other containing the discarded Columns.

□ Relation *r*.

A	В	C
α	10	1
α	20	1
β	30	1
β	40	2

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





Project Operation

- The general form of the project operation is π <attribute list>(r) where π (pi) is the symbol used to represent the project operation and <attribute list> is the desired list of attributes from the attributes of relation r.
- Notation: $\prod_{A_1,A_2,...,A_k}(r)$ 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
- ☐ The project operation *removes any duplicate tuples*, so the result of the project operation is a set of tuples and hence a valid relation.



Unary Relational Operations

PROJECT Operation Properties

- The number of tuples in the result of projection $\pi_{\text{<list>}}(r)$ is always less or equal to the number of tuples in r.
- If the list of attributes includes a key of r, then the number of tuples is equal to the number of tuples in r.

 $\pi_{\text{<list1>}}(\pi_{\text{<list2>}}(r)) = \pi_{\text{<list1>}}(r) \text{ as long as <list2>}$ contains the attributes in list1>



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



Assignment Operation

Example: To retrieve the first name, last name, and salary of all employees who work in department number 5, we must apply a select and a project operation. We can write a single relational algebra expression as follows:

 $\pi_{\text{FNAME, LNAME, SALARY}}(\sigma_{\text{DNO=5}}(\text{EMPLOYEE}))$

OR We can explicitly show the sequence of operations, giving a name to each intermediate relation:

DEP5_EMPS $\leftarrow \sigma_{DNO=5}(EMPLOYEE)$

RESULT $\leftarrow \pi_{\text{FNAME, LNAME, SALARY}}$ (DEP5_EMPS)



Relational Algebra Operations From Set Theory

□ Type Compatibility

□ The operand relations R₁(A₁, A₂, ..., A_n) and R₂(B₁, B₂, ..., B_n) must have the same number of attributes, and the domains of corresponding attributes must be compatible; that is, dom(A_i)=dom(B_i) for i=1, 2, ..., n.

□ The resulting relation for $R_1 \cup R_2$, $R_1 \cap R_2$, or $R_1 - R_2$ has the same attribute names as the *first* operand relation R1 (by convention).



Union of two relations

The result of this operation, denoted by $R \cup S$, is a relation that includes all tuples that are either in R or in S or in both R and S. Duplicate tuples are eliminated.

□ Relations *r*, *s*:

A	В
α	1
α	2
β	1

A	В
α	2
ß	3

 \square r \cup s:

Union of two sets r and s

A	В
α	1
α	2
β	1
β	3



Set Difference Operation

- ☐ The result of this operation, denoted by R S, is a relation that includes all tuples that are in R but not in S. The two operands must be "type compatible".
- □ Notation r s
- Defined as:

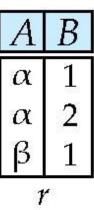
$$r-s = \{t \mid t \in r \text{ and } t \notin s\}$$

- 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



Set difference of two relations

□ Relations *r*, *s*:



A	В
α	2
β	3

 \Box r-s:

A	В
α	1
β	1



Intersection Operation

■ INTERSECTION OPERATION

The result of this operation, denoted by $R \cap S$, is a relation that includes all tuples that are in both R and S. The two operands must be "type compatible"

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



Intersection Operation – Example

□ Relation *r*, *s*:

Α	В
α	1
α	2
β	1

A B α 2 β 3

S

r

 \square $r \cap s$

A B α 2



Relational Algebra Operations From Set Theory (cont.)

□ Notice that both union and intersection are *commutative* operations; that is

$$R \cup S = S \cup R$$
, and $R \cap S = S \cap R$

□ Both union and intersection can be treated as n-ary operations applicable to any number of relations as both are *associative operations;* that is

$$R \cup (S \cup T) = (R \cup S) \cup T$$
, and $(R \cap S) \cap T = R \cap (S \cap T)$

☐ The minus operation is *not commutative*; that is, in general

$$R - S \neq S - R$$



Relational Algebra Operations From Set Theory (cont.)

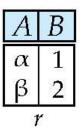
☐ CARTESIAN (or cross product) Operation

- This operation is used to combine tuples from two relations in a combinatorial fashion. In general, the result of R(A₁, A₂, ..., A_n) x S(B₁, B₂, ..., B_m) is a relation Q with degree n + m attributes Q(A₁, A₂, ..., A_n, B₁, B₂, ..., B_m), in that order. The resulting relation Q has one tuple for each combination of tuples—one from R and one from S.
- □ Hence, if R has n_R tuples (denoted as $|R| = n_R$), and S has n_S tuples, then
 - | R x S | will have n_R * n_S tuples.
- The two operands do NOT have to be "type compatible"
- 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.



joining two relations -- Cartesian-product

Relations *r, s*:



C	D	E
α	10	a
β	10	a
β	20	b
γ	10	b
	s	

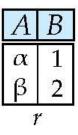
 \square $r \times s$:

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 – naming issue

Relations *r, s*:



\bar{B}	D	E
α	10	a
β	10	a
β	20	b
γ	10	b
	s	

 \square $r \times s$:

A	r.B	s.B	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



Rename Operation

The rename operator is ρ

The general Rename operation can be expressed by any of the following forms:

- $\rho_{S(B_1, B_2, ..., B_n)}$ (R) is a renamed relation S based on R with column names B_1, B_1,B_n.
- $-\rho_{S}(R)$ is a renamed relation S based on R (which does not specify column names).
- $\qquad \qquad \rho_{\,(B_1,\,B_2,\,...,\,B_n\,)} \,\,(\,\,R) \ \, \text{is a renamed relation with column names} \,\,_{B_1},\,_{B_1},\,....\,_{B_n} \,\,\text{which}$ does not specify a new relation name.



Renaming a Table

☐ Allows us to refer to a relation, (say E) by more than one name.

$$\rho_x(E)$$

returns the expression E under the name X

□ Relations *r*

A	В
α	1
β	2
1	

 \Box $r \times \rho_s(r)$

r.A	r.B	s.A	s.B
α	1	α	1
α	1	β	2
β	2	α	1
β	2	β	2



Composition of Operations

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

A	В	C	D	Е
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

 \Box $\sigma_{A=C}(rxs)$

A	В	C		E
α	1	α	10	a
β	2	β	10	a
β	2	β	20	b



Binary Relational Operations

JOIN Operation

- □ The sequence of cartesian product followed by select is used quite commonly to identify and select related tuples from two relations, a special operation, called **JOIN**. It is denoted by a ⋈
- This operation is very important for any relational database with more than a single relation, because it allows us to process relationships among relations.
- □ The general form of a join operation on two relations $R(A_1, A_2, ..., A_n)$ and $S(B_1, B_2, ..., B_m)$ is:

 $R \bowtie S$

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





Joining two relations – Natural Join

- Let r and s be relations on schemas R and S respectively. Then, the "natural join" of relations R and S is a relation on schema R ∪ 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



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

<u>D</u>	E
a	α
a	β
a	γ
b	δ
b	3
	a a b

- Natural Join
 - □ 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

Question: Can this be expressed in terms of other fundamental operators?

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



Notes about Relational Languages

- Each Query input is a table (or set of tables)
- Each query output is a table.
- □ All data in the output table appears in one of the input tables
- Relational Algebra is not Turing complete
- Can we compute:
 - SUM
 - AVG
 - MAX
 - MIN



Summary of Relational Algebra Operators

Symbol (Name)	Example of Use	
σ (Selection)	σ salary $>$ = 85000 (instructor)	
	Return rows of the input relation that satisfy the predicate.	
П (Projection)	П ID, salary ^(instructor)	
	Output specified attributes from all rows of the input relation. Remove duplicate tuples from the output.	
X (Cartesian Product)	instructor x department	
	Output all pairs of rows from the two input relations (regardless of whether they have the same value on all attributes that have the same name).	
∪ (Union)	Π name $(instructor) \cup \Pi$ name $(student)$	
	Output the union of tuples from the <i>two</i> input relations.	
- (Set Difference) П name (instructor) П name (student)		
	Output the set difference of tuples from the two input relations.	
⋈ (Natural Join)	instructor ⋈ department	
	Output pairs of rows from the two input relations that have the same value on all attributes that have the same name.	

End of Chapter 2

Edited by Radhika Sukapuram. Original slides by Database System Concepts, 6th Ed.

©Silberschatz, Korth and Sudarshan