Chapter 2: Intro to Relational Model

Edited by Radhika Sukapuram. Original slides by

Database System Concepts, 6th Ed.

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	B	C
α	10	1
α	20	1
β	30	1
β	40	2

- $\Pi_{A,C}\,(r)$

A	C
α	1
α	1
β	1
β	2

=

A	C
α	1
β	1
β	2

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

- $\pi_{\text{<list1>}}(\pi_{\text{<list2>}}(r)) = \pi_{\text{<list1>}}(r)$ 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 $\leftarrow$ is assigned to the relation variable on the left of the $\leftarrow$.
  - 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:

$$\text{DEP5\_EMPS} \leftarrow \sigma_{\text{DNO=5}}(\text{EMPLOYEE})$$

$$\text{RESULT} \leftarrow \pi_{\text{FNAME, LNAME, SALARY}}(\text{DEP5\_EMPS})$$

Relational Algebra Operations From Set Theory

- **Type Compatibility**
  - The operand relations $R_1(A_1, A_2, ..., A_n)$ and $R_2(B_1, B_2, ..., B_n)$ 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 ∪ 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	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

- r ∪ s:

A	B
α	1
α	2
β	1
β	3

- Union of two sets r and s

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:

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

- 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	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

- *r* – *s*:

A	B
α	1
β	1

Intersection Operation

- **INTERSECTION OPERATION**

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

  Defined as $r \cap s = \{ t \mid t \in r \textbf{ and } t \in s \}$
- Assume:
  - *r*, *s* have the *same arity*
  - attributes of *r* and *s* are compatible

Intersection Operation – Example

- Relation *r*, *s*:

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

- $r \cap s$

A	B
α	2

Relational Algebra Operations From Set Theory (cont.)

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

  $$\mathbf{R \cup S = S \cup R, \text{ and } R \cap S = S \cap R}$$

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

  $$\mathbf{R \cup (S \cup T) = (R \cup S) \cup T, \text{ and } (R \cap S) \cap T = R \cap (S \cap T)}$$

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

  $$\mathbf{R - S \neq S - R}$$

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 $R(A_1, A_2, \ldots, A_n) \times S(B_1, B_2, \ldots, B_m)$ is a relation Q with degree n + m attributes $Q(A_1, A_2, \ldots, A_n, B_1, B_2, \ldots, B_m)$, 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

    $| R \times S |$ 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*:

A	B
α	1
β	2

r

C	D	E
α	10	a
β	10	a
β	20	b
γ	10	b

s

- *r* x *s*:

A	B	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 – naming issue

- Relations *r, s*:

A	B
α	1
β	2

r

B	D	E
α	10	a
β	10	a
β	20	b
γ	10	b

s

- *r* x *s*:

A	r.B	s.B	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

Rename Operation

The rename operator is ρ

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

- $\rho_{S(B_1, B_2, \ldots, B_n)}(R)$ 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).
- $\rho_{(B_1, B_2, \ldots, B_n)}(R)$ is a renamed relation with column names B_1, B_1, …..B_n 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	B
α	1
β	2

r

- $r \times \rho_{s}(r)$

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: $\sigma_{A=C}(r \times s)$
- *r x s*

A	B	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

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

A	B	C	D	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 $\bowtie$
  - 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 $R(A_1, A_2, \ldots, A_n)$ and $S(B_1, B_2, \ldots, B_m)$ 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 $R \cup S$ 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*

Natural Join Example

- Relations r, s:

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

r

B	D	E
1	a	α
3	a	β
1	a	γ
2	b	δ
3	b	ε

s

- Natural Join
  - $r \bowtie s$

A	B	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?

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

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)	$\sigma_{\text{salary} >= 85000}(instructor)$
	Return rows of the input relation that satisfy the predicate.
Π (Projection)	$\Pi_{ID,\ salary}(instructor)$
	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).
$\cup$ (Union)	$\Pi_{name}(instructor) \cup \Pi_{name}(student)$
	Output the union of tuples from the *two* input relations.
– (Set Difference)	$\Pi_{name}(instructor)$ -- $\Pi_{name}(student)$
	Output the set difference of tuples from the two input relations.
$\bowtie$ (Natural Join)	*instructor* $\bowtie$ *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