Chapter 8: The Relational Algebra

Relational Algebra Overview

Relational Algebra consists of several groups of operations

- Unary Relational Operations
  - SELECT (symbol: **σ** (sigma))
  - PROJECT (symbol: **π** (pi))
  - RENAME (symbol: **ρ** (rho))
- Relational Algebra Operations From Set Theory
  - UNION (∪), INTERSECTION (∩), DIFFERENCE (or MINUS, **–**)
  - CARTESIAN PRODUCT (**x**)
- Binary Relational Operations
  - JOIN (several variations of JOIN exist)
  - DIVISION
- Additional Relational Operations
  - OUTER JOINS, OUTER UNION
  - AGGREGATE FUNCTIONS (These compute summary of information: for example, SUM, COUNT, AVG, MIN, MAX)

Unary Relational Operations: SELECT

The SELECT operation (denoted by σ (sigma)) is used to select a *subset* of the tuples from a relation based on a **selection condition**.

- The selection condition acts as a **filter**
- Keeps only those tuples that satisfy the qualifying condition
- Tuples satisfying the condition are *selected* whereas the other tuples are discarded (*filtered out*)

Examples:

- Select the EMPLOYEE tuples whose department number is 4:

$$\sigma_{\text{DNO} = 4}\ (\text{EMPLOYEE})$$

- Select the employee tuples whose salary is greater than $30,000:

$$\sigma_{\text{SALARY} > 30{,}000}\ (\text{EMPLOYEE})$$

Unary Relational Operations: SELECT

In general, the *select* operation is denoted by $\boldsymbol{\sigma}_{\text{<selection condition>}}(R)$ where

- the symbol $\boldsymbol{\sigma}$ (sigma) is used to denote the *select* operator
- the selection condition is a Boolean (conditional) expression specified on the attributes of relation R
- tuples that make the condition **true** are selected
    - appear in the result of the operation
- tuples that make the condition **false** are filtered out
    - discarded from the result of the operation

Unary Relational Operations: SELECT (continued)

SELECT Operation Properties

- The SELECT operation $\sigma_{\text{<selection condition>}}(R)$ produces a relation S that has the same schema (same attributes) as R
- SELECT σ is commutative:
  - $\sigma_{\text{<condition1>}}(\sigma_{\text{< condition2>}}(R)) = \sigma_{\text{<condition2>}}(\sigma_{\text{< condition1>}}(R))$
- Because of commutativity property, a cascade (sequence) of SELECT operations may be applied in any order:
  - $\sigma_{\text{<cond1>}}(\sigma_{\text{<cond2>}}(\sigma_{\text{<cond3>}}(R)) = \sigma_{\text{<cond2>}}(\sigma_{\text{<cond3>}}(\sigma_{\text{<cond1>}}(R)))$
- A cascade of SELECT operations may be replaced by a single selection with a conjunction of all the conditions:
  - $\sigma_{\text{<cond1>}}(\sigma_{\text{< cond2>}}(\sigma_{\text{<cond3>}}(R)) = \sigma_{\text{<cond1> AND < cond2> AND < cond3>}}(R)))$
- The number of tuples in the result of a SELECT is less than (or equal to) the number of tuples in the input relation R

Unary Relational Operations: PROJECT

- PROJECT Operation is denoted by π (pi)
- This operation keeps certain *columns* (attributes) from a relation and discards the other columns.
  - PROJECT creates a vertical partitioning
    - The list of specified columns (attributes) is kept in each tuple
    - The other attributes in each tuple are discarded
- Example: To list each employee's first and last name and salary, the following is used:

$$\pi_{\text{LNAME, FNAME,SALARY}}(\text{EMPLOYEE})$$

Unary Relational Operations: PROJECT (cont.)

- The general form of the *project* operation is:

$$\pi_{\text{<attribute list>}}(R)$$

  - π (pi) is the symbol used to represent the *project* operation

    <attribute list> is the desired list of attributes from relation R.
- The project operation *removes any duplicate tuples*
  - This is because the result of the *project* operation must be a *set of tuples*
    - Mathematical sets *do not allow* duplicate elements.

Unary Relational Operations: PROJECT (contd.)

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 in the result of PROJECT is *equal* to the number of tuples in R

PROJECT is *not* commutative

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

Relational Algebra Expressions

We may want to apply several relational algebra operations one after the other

- Either we can write the operations as a single **relational algebra expression** by nesting the operations, or
- We can apply one operation at a time and create **intermediate result relations**.

In the latter case, we must give names to the relations that hold the intermediate results.

Single expression versus sequence of relational operations (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:
  - $\text{DEP5\_EMPS} \leftarrow \sigma_{\text{DNO=5}}(\text{EMPLOYEE})$
  - $\text{RESULT} \leftarrow \pi_{\text{FNAME, LNAME, SALARY}}(\text{DEP5\_EMPS})$

Unary Relational Operations: RENAME

- The RENAME operator is denoted by ρ (rho)
- In some cases, we may want to *rename* the attributes of a relation or the relation name or both
    - Useful when a query requires multiple operations
    - Necessary in some cases (see JOIN operation later)

Unary Relational Operations: RENAME (continued)

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

- $\rho_{S\ (B1,\ B2,\ \ldots,\ Bn\ )}(R)$ changes both:
  - the relation name to S, *and*
  - the column (attribute) names to B1, B1, …..Bn
- $\rho_{S}(R)$ changes:
  - the *relation name* only to S
- $\rho_{(B1,\ B2,\ \ldots,\ Bn\ )}(R)$ changes:
  - the *column (attribute) names* only to B1, B1, …..Bn

Unary Relational Operations: RENAME (continued)

For convenience, we also use a *shorthand* for renaming attributes in an intermediate relation:

- If we write:
  - RESULT $\leftarrow \pi_{\text{FNAME, LNAME, SALARY}}$ (DEP5_EMPS)
  - RESULT will have the *same attribute names* as DEP5_EMPS (same attributes as EMPLOYEE)
- If we write:
  - RESULT (F, M, L, S, B, A, SX, SAL, SU, DNO)$\leftarrow$ $\rho_{\text{RESULT (F.M.L.S.B,A,SX,SAL,SU, DNO)}}$(DEP5_EMPS)
  - The 10 attributes of DEP5_EMPS are *renamed* to F, M, L, S, B, A, SX, SAL, SU, DNO, respectively

Note: the $\leftarrow$ symbol is an assignment operator

Relational Algebra Operations from Set Theory: UNION

UNION Operation

- Binary operation, denoted by $\cup$
- The result of $R \cup S$, 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
- The two operand relations R and S must be "type compatible" (or UNION compatible)
  - R and S must have same number of attributes
  - Each pair of corresponding attributes must be type compatible (have same or compatible domains)

Relational Algebra Operations from Set Theory: UNION

Example:

- To retrieve the social security numbers of all employees who either *work in department 5* (RESULT1 below) or *directly supervise an employee who works in department 5* (RESULT2 below)

- We can use the UNION operation as follows:

$$DEP5\_EMPS \leftarrow \sigma_{DNO=5}(EMPLOYEE)$$
$$RESULT1 \leftarrow \pi_{SSN}(DEP5\_EMPS)$$
$$RESULT2(SSN) \leftarrow \pi_{SUPERSSN}(DEP5\_EMPS)$$
$$RESULT \leftarrow RESULT1 \cup RESULT2$$

- The union operation produces the tuples that are in either RESULT1 or RESULT2 or both

RESULT1

Ssn
123456789
333445555
666884444
453453453

RESULT2

Ssn
333445555
888665555

RESULT

Ssn
123456789
333445555
666884444
453453453
888665555

Relational Algebra Operations from Set Theory

- Type Compatibility of operands is required for the binary set operation UNION $\cup$, (also for INTERSECTION $\cap$, and SET DIFFERENCE –, see next slides)
- R1(A1, A2, ..., An) and R2(B1, B2, ..., Bn) are type compatible if:
  - they have the same number of attributes, and
  - the domains of corresponding attributes are type compatible (i.e. dom(Ai)=dom(Bi) for i=1, 2, ..., n).
- The resulting relation for $R1 \cup R2$ (also for $R1 \cap R2$, or R1–R2, see next slides) has the same attribute names as the *first* operand relation R1 (by convention)

Relational Algebra Operations from Set Theory: INTERSECTION

- INTERSECTION is denoted by $\cap$
- The result of the operation $R \cap S$, is a relation that includes all tuples that are in both R and S
  - The attribute names in the result will be the same as the attribute names in R
- The two operand relations R and S must be "type compatible"

Relational Algebra Operations from Set Theory: SET DIFFERENCE (cont.)

- SET DIFFERENCE (also called MINUS or EXCEPT) is denoted by –
- The result of R – S, is a relation that includes all tuples that are in R but not in S
    - The attribute names in the result will be the same as the attribute names in R
- The two operand relations R and S must be "type compatible"

Example to illustrate the result of UNION, INTERSECT, and DIFFERENCE

Figure 8.4
The set operations UNION, INTERSECTION, and MINUS. (a) Two union-compatible relations. (b) STUDENT ∪ INSTRUCTOR. (c) STUDENT ∩ INSTRUCTOR. (d) STUDENT – INSTRUCTOR. (e) INSTRUCTOR – STUDENT.

(a) STUDENT

Fn	Ln
Susan	Yao
Ramesh	Shah
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert

INSTRUCTOR

Fname	Lname
John	Smith
Ricardo	Browne
Susan	Yao
Francis	Johnson
Ramesh	Shah

(b)

Fn	Ln
Susan	Yao
Ramesh	Shah
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert
John	Smith
Ricardo	Browne
Francis	Johnson

(c)

Fn	Ln
Susan	Yao
Ramesh	Shah

(d)

Fn	Ln
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert

(e)

Fname	Lname
John	Smith
Ricardo	Browne
Francis	Johnson

Relational Algebra Operations from Set Theory: CARTESIAN PRODUCT

CARTESIAN (or CROSS) PRODUCT Operation

- This operation is used to combine tuples from two relations in a combinatorial fashion.
- Denoted by R(A1, A2, . . ., An) x S(B1, B2, . . ., Bm)
- Result is a relation Q with degree n + m attributes:
  - Q(A1, A2, . . ., An, B1, B2, . . ., Bm), in that order.
- The resulting relation state 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”

Relational Algebra Operations from Set Theory: CARTESIAN PRODUCT (cont.)

- Generally, CROSS PRODUCT is not a meaningful operation
  - Can become meaningful when followed by other operations
- Example (not meaningful):
  - FEMALE_EMPS $\leftarrow \sigma_{\text{Gender='F'}}$(EMPLOYEE)
  - EMPNAMES $\leftarrow \pi_{\text{FNAME, LNAME, SSN}}$ (FEMALE_EMPS)
  - EMP_DEPENDENTS $\leftarrow$ EMPNAMES x DEPENDENT
- EMP_DEPENDENTS will contain every combination of EMPNAMES and DEPENDENT
  - whether or not they are actually related

Relational Algebra Operations from Set Theory: CARTESIAN PRODUCT (cont.)

- To keep only combinations where the DEPENDENT is related to the EMPLOYEE, we add a SELECT operation as follows
- Example (meaningful):
  - FEMALE_EMPS $\leftarrow \sigma_{\text{Gender='F'}}$(EMPLOYEE)
  - EMPNAMES $\leftarrow \pi_{\text{FNAME, LNAME, SSN}}$ (FEMALE_EMPS)
  - EMP_DEPENDENTS $\leftarrow$ EMPNAMES x DEPENDENT
  - ACTUAL_DEPS $\leftarrow \sigma_{\text{SSN=ESSN}}$(EMP_DEPENDENTS)
  - RESULT $\leftarrow \pi_{\text{FNAME, LNAME, DEPENDENT\_NAME}}$ (ACTUAL_DEPS)
- RESULT will now contain the name of female employees and their dependents

Binary Relational Operations: JOIN

JOIN Operation (denoted by IXI)

- The sequence of CARTESIAN PRODECT followed by SELECT is used quite commonly to identify and select related tuples from two relations
- A special operation, called JOIN combines this sequence into a single operation
- This operation is very important for any relational database with more than a single relation, because it allows us *combine related tuples* from various relations
- The general form of a join operation on two relations R(A1, A2, . . ., An) and S(B1, B2, . . ., Bm) is:

$$R \text{ IXI}_{\text{<join condition>}} S$$

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

Binary Relational Operations: JOIN (cont.)

- Example: Suppose that we want to retrieve the name of the manager of each department.
  - To get the manager's name, we need to combine each DEPARTMENT tuple with the EMPLOYEE tuple whose SSN value matches the MGRSSN value in the department tuple.
  - We do this by using the join operation.

  - DEPT_MGR ← DEPARTMENT |X| $_{\text{MGRSSN=SSN}}$ EMPLOYEE
- MGRSSN=SSN is the join condition
  - Combines each department record with the employee who manages the department
  - The join condition can also be specified as DEPARTMENT.MGRSSN= EMPLOYEE.SSN

Complete Set of Relational Operations

- The set of operations including SELECT σ, PROJECT π , UNION $\cup$, DIFFERENCE $-$, RENAME ρ, and CARTESIAN PRODUCT X is called a *complete set* because any other relational algebra expression can be expressed by a combination of these five operations.
- For example:
  - $R \cap S = (R \cup S) - ((R - S) \cup (S - R))$

Table 8.1 Operations of Relational Algebra

Table 8.1 Operations of Relational Algebra

OPERATION	PURPOSE	NOTATION
SELECT	Selects all tuples that satisfy the selection condition from a relation R.	$\sigma_{\text{<selection condition>}}(R)$
PROJECT	Produces a new relation with only some of the attributes of R, and removes duplicate tuples.	$\pi_{\text{<attribute list>}}(R)$
THETA JOIN	Produces all combinations of tuples from R_1 and R_2 that satisfy the join condition.	$R_1 \bowtie_{\text{<join condition>}} R_2$
EQUIJOIN	Produces all the combinations of tuples from R_1 and R_2 that satisfy a join condition with only equality comparisons.	$R_1 \bowtie_{\text{<join condition>}} R_2$, OR $R_1 \bowtie_{(\text{<join attributes 1>}),\,(\text{<join attributes 2>})} R_2$
NATURAL JOIN	Same as EQUIJOIN except that the join attributes of R_2 are not included in the resulting relation; if the join attributes have the same names, they do not have to be specified at all.	$R_1 *_{\text{<join condition>}} R_2$, OR $R_1 *_{(\text{<join attributes 1>}),\,(\text{<join attributes 2>})} R_2$ OR $R_1 * R_2$

Table 8.1 Operations of Relational Algebra (continued)

Table 8.1 Operations of Relational Algebra

OPERATION	PURPOSE	NOTATION
UNION	Produces a relation that includes all the tuples in R_1 or R_2 or both R_1 and R_2; R_1 and R_2 must be union compatible.	$R_1 \cup R_2$
INTERSECTION	Produces a relation that includes all the tuples in both R_1 and R_2; R_1 and R_2 must be union compatible.	$R_1 \cap R_2$
DIFFERENCE	Produces a relation that includes all the tuples in R_1 that are not in R_2; R_1 and R_2 must be union compatible.	$R_1 - R_2$
CARTESIAN PRODUCT	Produces a relation that has the attributes of R_1 and R_2 and includes as tuples all possible combinations of tuples from R_1 and R_2.	$R_1 \times R_2$
DIVISION	Produces a relation $R(X)$ that includes all tuples $t[X]$ in $R_1(Z)$ that appear in R_1 in combination with every tuple from $R_2(Y)$, where $Z = X \cup Y$.	$R_1(Z) \div R_2(Y)$

Aggregate Function Operation

- Use of the Aggregate Functional operation $\mathcal{F}$
  - $\mathcal{F}_{\text{MAX Salary}}$ (EMPLOYEE) retrieves the maximum salary value from the EMPLOYEE relation
  - $\mathcal{F}_{\text{MIN Salary}}$ (EMPLOYEE) retrieves the minimum Salary value from the EMPLOYEE relation
  - $\mathcal{F}_{\text{SUM Salary}}$ (EMPLOYEE) retrieves the sum of the Salary from the EMPLOYEE relation
  - $\mathcal{F}_{\text{COUNT SSN, AVERAGE Salary}}$ (EMPLOYEE) computes the count (number) of employees and their average salary
    - Note: count just counts the number of rows, without removing duplicates

Using Grouping with Aggregation

- The previous examples all summarized one or more attributes for a set of tuples
  - Maximum Salary or Count (number of) Ssn
- Grouping can be combined with Aggregate Functions
- Example: For each department, retrieve the DNO, COUNT SSN, and AVERAGE SALARY
- A variation of aggregate operation $\mathcal{F}$ allows this:
  - Grouping attribute placed to left of symbol
  - Aggregate functions to right of symbol
  - $_{\text{DNO}}\,\mathcal{F}_{\text{COUNT SSN, AVERAGE Salary}}$ (EMPLOYEE)
- Above operation groups employees by DNO (department number) and computes the count of employees and average salary per department

Figure 8.10 The aggregate function operation.

a. $\rho_{R(\text{Dno, No\_of\_employees, Average\_sal})}(_{\text{Dno}}\,\Im\,_{\text{COUNT Ssn, AVERAGE Salary}}\,(\text{EMPLOYEE}))$.

b. $_{\text{Dno}}\,\Im\,_{\text{COUNT Ssn, AVERAGE Salary}}(\text{EMPLOYEE})$.

c. $\Im\,_{\text{COUNT Ssn, AVERAGE Salary}}(\text{EMPLOYEE})$.

R

(a)

Dno	No_of_employees	Average_sal
5	4	33250
4	3	31000
1	1	55000

(b)

Dno	Count_ssn	Average_salary
5	4	33250
4	3	31000
1	1	55000

(c)

Count_ssn	Average_salary
8	35125

Examples of Queries in Relational Algebra : Procedural Form

- **Q1: Retrieve the name and address of all employees who work for the 'Research' department.**

  RESEARCH_DEPT $\leftarrow \sigma_{\text{DNAME='Research'}}$(DEPARTMENT)

  RESEARCH_EMPS $\leftarrow$ (RESEARCH_DEPT |X| $_{\text{DNUMBER= DNOEMPLOYEE}}$EMPLOYEE)

  RESULT $\leftarrow \pi_{\text{FNAME, LNAME, ADDRESS}}$ (RESEARCH_EMPS)

- **Q6: Retrieve the names of employees who have no dependents.**

  ALL_EMPS $\leftarrow \pi_{\text{SSN}}$(EMPLOYEE)

  EMPS_WITH_DEPS(SSN) $\leftarrow \pi_{\text{ESSN}}$(DEPENDENT)

  EMPS_WITHOUT_DEPS $\leftarrow$ (ALL_EMPS - EMPS_WITH_DEPS)

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

Examples of Queries in Relational Algebra – Single expressions

As a single expression, these queries become:

- **Q1: Retrieve the name and address of all employees who work for the 'Research' department.**

$\pi_{\text{Fname, Lname,Address}}\ (\sigma_{\text{Dname= 'Research'}}$ $(\text{DEPARTMENT} \bowtie_{\text{Dnumber=Dno}} (\text{EMPLOYEE}))$

- **Q6: Retrieve the names of employees who have no dependents.**

$\pi_{\text{Lname, Fname}}((\pi_{\text{Ssn}}\ (\text{EMPLOYEE}) - \rho_{\text{Ssn}}\ (\pi_{\text{Essn}}$ $(\text{DEPENDENT}))) * \text{EMPLOYEE})$