The Relational Algebra

Chapter 4 a

Chapter Outline

- The objective of this chapter is to get acquainted with:
- Relational Algebra
  - Unary Relational Operations
    - Select, project, rename,
  - Relational Algebra Operations From Set Theory
    - Union, difference, intersection,cartesian product
  - Binary Relational Operations
    - Join, division
  - Additional Relational Operations
  - Examples of Queries in Relational Algebra
- Example of Queries

Database State for COMPANY

Employee

FName	MInit	LName	SSN	BDate	Address	Sex	Salary	SuperSSN	DNO

Department

DName	DNumber	MGRSSN	MGRStartDate

DeptLocations

DNumber	DLocation

Project

PName	PNumber	PLocation	DNum

WorksOn

ESSN	PNO	Hours

Dependent

ESSN	DependentName	Sex	BDate	Relationship

Relational Algebra

- The basic set of operations for the relational model is known as the relational algebra. These operations enable a user to specify basic retrieval requests.
- The result of a retrieval is a new relation, which may have been formed from one or more relations.

Relational Algebra

- The **algebra operations** thus produce new relations, which can be further manipulated using operations of the same algebra.
- A sequence of relational algebra operations forms a **relational algebra expression**, whose result will also be a relation that represents the result of a database query (or retrieval request).

The Relational Algebra

- The aim is to define a set of operations that allow the user to perform different types of queries
- We talk about ***data manipulation***.
- The relational algebra is the internal language of a Relational DBMS.
  - Two sets of operations are considered
  - Relational Operators:
    - ***Select***, ***Project***, ***Join*** and ***Division***
  - Set Theoretic Operators
    - ***Union***, ***Intersection***, ***Difference***, ***Cartesian Product***

Relational Algebra (Cont.)

- Procedural language
- Six basic operators
  - select
  - project
  - union
  - set difference
  - Cartesian product
  - rename
- The operators take one or more relations as inputs and give a new relation as a result.

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:
  - $E_1 \cup E_2$
  - E_1 - E_2
  - E_1 x E_2
  - $\sigma_p (E_1)$, P is a predicate on attributes in E_1
  - $\prod_s (E_1)$, 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

Unary Relational Operations

SELECT Operation

SELECT operation is used to select a *subset* of the tuples from a relation that satisfy a **selection condition**. It is a filter that keeps only those tuples that satisfy a qualifying condition – those satisfying the condition are selected while others are discarded.

In general, the select operation is denoted by

$$\sigma_{<\text{selection condition}>}(R)$$

where the symbol **σ** (sigma) is used to denote the select operator, and the selection condition is a Boolean expression specified on the attributes of relation R

SELECT Operation

Example:

To select the EMPLOYEE tuples whose department number is four or those whose salary is greater than 15000 the following notation is used:

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

$$\sigma_{SALARY > 15000}\ (EMPLOYEE)$$

SELECT Operation Properties

- The SELECT operation $\sigma_{<\text{selection condition}>}(R)$ produces a relation S that has the same schema as R
- The SELECT operation σ is **commutative;** i.e.,

  $\sigma_{<\text{condition1}>}(\sigma_{<\text{condition2}>}(R)) = \sigma_{<\text{condition2}>}(\sigma_{<\text{condition1}>}(R))$

- A cascaded SELECT operation **may be applied in any order:**

  $\sigma_{<\text{condition1}>}(\sigma_{<\text{condition2}>}(\sigma_{<\text{condition3}>}(R))$
  $= \sigma_{<\text{condition2}>}(\sigma_{<\text{condition3}>}(\sigma_{<\text{condition1}>}(R)))$

- A cascaded SELECT operation may be replaced by a single selection with a conjunction of all the conditions**:**

  $\sigma_{<\text{condition1}>}(\sigma_{<\text{condition2}>}(\sigma_{<\text{condition3}>}(R))$
  $= \sigma_{<\text{condition1}> \text{ AND } <\text{condition2}> \text{ AND } <\text{condition3}>}(R)))$

Select Operation – Example

- Relation *r*

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

- $\sigma_{A=B \wedge D > 5}(r)$

A	B	C	D
α	α	1	7
β	β	23	10

Unary Relational Operations

- **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.

Example: To list each employee's first and last name and salary, the following is used:

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

Example

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

$\pi_{\text{SSD, Fname, Address}}$ (Employee)

$\pi_{\text{Fname, D\#}}$ (Employee)

$\pi_{\text{D\#, Dname}}$ (Department)

$\pi_{\text{D\#, Dname}}$ (Department)

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.

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 (cont.)

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 of <list1>

Project Operation – Example

- 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

Unary Relational Operations

- **Rename Operation**

$x = 5*4 - 6*(9-x)$

$x_1 = 6*(9-x)$

$x_2 = 5*4 - x_1$

To apply several relational algebra operations one after the other, 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.

Rename 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:

Local variable متغير محلي

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

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

Relational Algebra Expression

- Writing a single nested expression :
- $\pi_{\text{Fname, Salary}}(\sigma_{\text{Address='Riyadh'}}(\text{Employee}))$
- $\sigma_{\text{Salary > 1000}}(\pi_{\text{Fname, SSN}}(\sigma_{\text{Address = 'Ryadh'}}(\text{Employee})))$
- Applying one operation at a time and create an intermediate result relation:
- Each intermediate result must have a name.
- $R1 = \sigma_{SSN= 693548}(\text{Employee})$
- $R2 = \pi_{\text{Flame, Address}}(R1)$
- We can also rename the attributes of the result relation:
- $R2(\text{Name, Add}) = \pi_{\text{Name, Address}}(Rl)$

Rename Operation (cont.)

The rename operator is ρ

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

- $\rho_{S(B1, B2, \ldots, Bn)}(R)$ is a renamed relation S based on R with column names B1, B1, ...Bn.
  - $\rho_{S}(R)$ is a renamed relation S based on R (which does not specify column names).
  - $\rho_{(B1, B2, \ldots, Bn)}(R)$ is a renamed relation with column names B1, B1, ...Bn which does not specify a new relation name.

Relational Algebra Operations From Set Theory

- **UNION Operation**

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.

Example: To retrieve the social security numbers of all employees who either work in department 5 or directly supervise an employee who works in department 5, we can use the union operation as follows:

UNION Operation

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

$$\mathbf{RESULT1 \leftarrow \pi_{SSN}(DEP5\_EMPS)}$$

$$\mathbf{RESULT2(SSN) \leftarrow \pi_{SUPERSSN}(DEP5\_EMPS)}$$

$$\mathbf{RESULT \leftarrow RESULT1 \cup RESULT2}$$

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

The two operands must be “type compatible”.

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 Operation – Example

- Relations *r*, *s*:

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

$r \cup s$:

A	B
α	1
α	2
β	1
β	3

Relational Algebra Operations From Set Theory (cont.)

- **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"

  Example: The result of the intersection operation (figure below) includes only those who are both students and instructors.

Relational Algebra Operations From Set Theory

- **Set Difference (or MINUS) 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".

  Example: The figure shows the names of students who are not instructors, and the names of instructors who are not students.

Relational Algebra Operations From Set Theory

- 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

  $R - S \neq S - R$

Set Difference Operation – Example

- 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

Relational Algebra Operations From Set Theory

- **CARTESIAN (or cross product) Operation**

It is used to combine tuples from two relations in a combinatorial fashion. The result of $R(A_1, A_2, \ldots, A_n)$ x $S(B_1, B_2, \ldots, B_m)$ 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"

CARTESIAN (or cross product) Operation

Example:

FEMALE_EMPS ← σ $_{\text{SEX='F'}}$**(EMPLOYEE)**

EMPNAMES ← π $_{\text{FNAME, LNAME, SSN}}$ **(FEMALE_EMPS)**

EMP_DEPENDENTS ← EMPNAMES x DEPENDENT

Cartesian-Product Operation-Example

- 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	19	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

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:

    $R \underset{\text{<join condition>}}{\bowtie} S$

Binary Relational Operations (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 $\bowtie$ operation.

DEPT_MGR $\leftarrow$ DEPARTMENT $\underset{\text{MGRSSN=SSN}}{\bowtie}$ EMPLOYEE

Binary Relational Operations (cont.)

- **EQUIJOIN Operation**

The most common use of join involves join conditions with equality comparisons only. Such a join, where the only comparison operator used is =, is called an EQUIJOIN. In the result of an EQUIJOIN we always have one or more pairs of attributes (whose names need not be identical) that have *identical values* in every tuple.

The JOIN seen in the previous example was EQUIJOIN.

NATURAL JOIN Operation

- Because one of each pair of attributes with identical values is superfluous, a new operation called natural join—denoted by *—was created to get rid of the second (superfluous) attribute in an EQUIJOIN condition.

The standard definition of natural join requires that the two join attributes, or each pair of corresponding join attributes, have the **same name** in both relations. If this is not the case, a renaming operation is applied first.

Complete Set of Relational Operations

- The set of operations including **select σ, project π , union ∪, set difference - , 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:

  $\mathbf{R \cap S = (R \cup S) - ((R - S) \cup (S - R))}$

  $R \bowtie_{\text{<join condition>}} S = \sigma_{\text{<join condition>}} (R \times S)$

Binary Relational Operations (cont.)

- **DIVISION Operation**
  - The division operation is applied to two relations

    $R(Z) \div S(X)$, where X subset Z. Let $Y = Z - X$ (and hence $Z = X \cup Y$); that is, let Y be the set of attributes of R that are not attributes of S.

  - The result of DIVISION is a relation T(Y) that includes a tuple t if tuples t_R appear in R with $t_R[Y] = t$, and with

    $t_R[X] = t_s$ *for every tuple* t_s in S.

  - For a tuple t to appear in the result T of the DIVISION, the values in t must appear in R in combination with *every* tuple in S.

Binary Relational Operations (cont.)

SSN_PNOS	ESSN	PNO
	123456789	1
	123456789	2
	666884444	3
	453453453	1
	453453453	2
	333445555	2
	333445555	3
	333445555	10
	333445555	20
	999887777	30
	999887777	10
	987987987	10
	987987987	30
	987654321	30
	987654321	20
	888665555	20

R	A	B
	a1	b1
	a2	b1
	a3	b1
	a4	b1
	a1	b2
	a3	b2
	a2	b3
	a3	b3
	a4	b3
	a1	b4
	a2	b4
	a3	b4

Examples of Queries in Relational Algebra

24-ch5

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

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

  RESEARCH_EMPS ← (RESEARCH_DEPT $\bowtie_{\text{DNUMBER= DNO}}$ EMPLOYEE)

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

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

  ALL_EMPS ← π_{SSN}(EMPLOYEE)

  EMPS_WITH_DEPS(SSN) ← π_{ESSN}(DEPENDENT)

  EMPS_WITHOUT_DEPS ← (ALL_EMPS - EMPS_WITH_DEPS)

  RESULT ← $\pi_{\text{LNAME, FNAME}}$ (EMPS_WITHOUT_DEPS * EMPLOYEE)

Recap of Relational Algebra Operations

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 $R_1 * R_2$
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 $Z = X \cup Y$.	$R_1(Z) \div R_2(Y)$