Chapter 8: Relational Algebra

Outline:

8.1 Introduction
8.2 Unary Relational Operations.
- Select Operator (σ)
- Project Operator (π)
- Rename Operator (ρ)
- Assignment Operator (←)

8.3 Binary Relational Operations.
- Set Operators
  - Union Operator (∪)
  - Intersection Operator (∩)
  - Set Difference or Minus Operator (-)
- Cartesian Product Operator (×)
- Join Operator
  - Theta Join ($\bowtie_{\square}$)
  - Natural Join ($\bowtie$) or (*)

8.4 Examples of Queries in Relational Algebra.

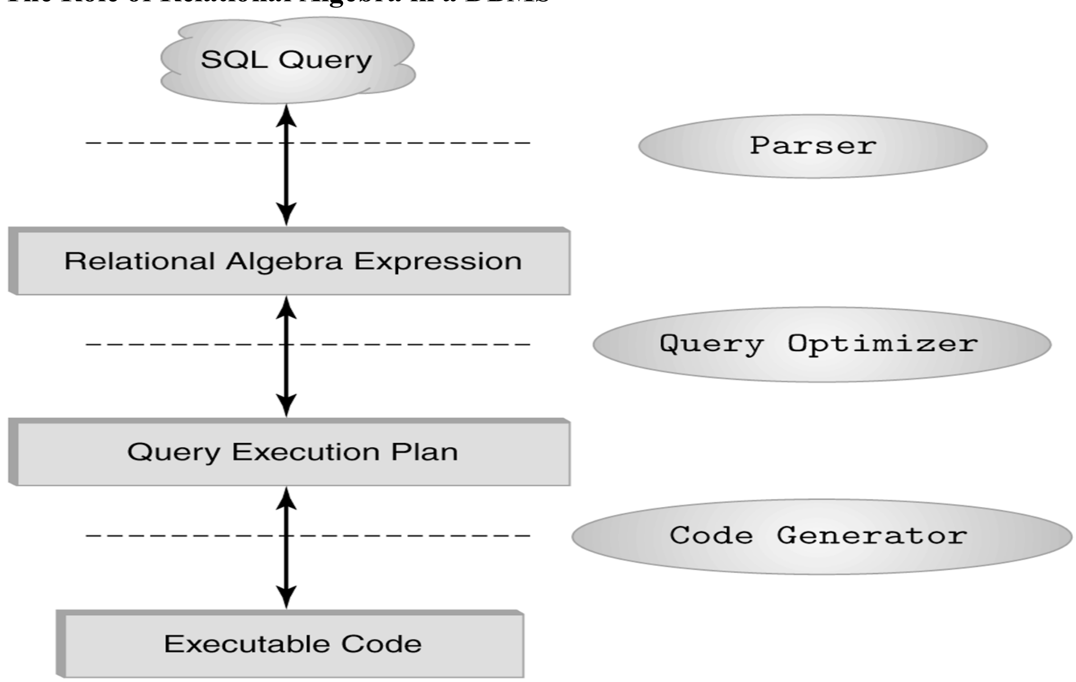
8.1 Introduction

Relational Query Languages

- Languages for describing queries on a relational database
- Structured Query Language (SQL)
  - Declarative (Nonprocedural)
- Relational Algebra
  - Intermediate language used within DBMS
  - Procedural
- Procedural: Relational expression specifies query by describing an algorithm (the sequence in which operators are applied) for determining the result of an expression

- Relational algebra is the basic set of operations for the relational model
- These operations enable a user to specify basic retrieval requests (or queries)
- The result of an operation is a new relation, which may have been formed from one or more input relations
  - This property makes the algebra "closed" (all objects in relational algebra are relations)
- The algebra operations thus produce new relations
  - These can be further manipulated using operations of the same algebra
- A sequence of relational algebra operations forms a relational algebra expression
  - The result of a relational algebra expression is also a relation that represents the result of a database query (or retrieval request)
- The fundamental operations in the relational algebra are select, project, union, set difference, cartesian product, and rename.

The Role of Relational Algebra in a DBMS



8.2 Unary Relational Operations

Select Operator (σ)

- Select a subset of rows from a relation that satisfying a condition.
  Syntax:
  $\sigma_{condition}\,(R)$

  - The symbol σ (sigma) is used to denote the select operator.
  - The selection condition is a Boolean expression specified on the attributes of relation R.
- $\sigma_{condition}\,(R)$ is equivalent to “Select * from R where <condition>;”

- Example: Consider the following relation r.

r

A	B	C	D
X	X	1	8
X	Y	5	7
Y	Y	3	3
Y	Y	12	10

1. $\sigma_{A=B}\,(r)$ = Select * from r where A=B;

A	B	C	D
X	X	1	8
Y	Y	3	3
Y	Y	12	10

2. $\sigma_{D>5}\,(r)$ = Select * from r where D>5;

A	B	C	D
X	X	1	8
X	Y	5	7
Y	Y	12	10

- We can combine several conditions into a larger condition by using the connectives ^ (and), □ (or), and ¬ (not).
- Example:
  $\sigma_{A=B\ \wedge\ D>5}\,(r)$ = Select * from r where A=B and D>5;

A	B	C	D
X	X	1	8
Y	Y	12	10

- Example: Retrieve the Id, Name, Address of students who live in Amman.

Student

Id	Name	Address
1108	Ali	Amman
1453	Ahmad	Salt
1002	Omar	Amman
2603	Anas	Zarqa

$\sigma_{address='Amman'}$ (Student) = Select * from Student where address='Amman';

Id	Name	Address
1108	Ali	Amman
1002	Omar	Amman

- Example: Retrieve the Id, Name, Address of student who's name is Ahmad or students who live in Amman.

  $\sigma_{name='Ahmad'\ \square\ ddress='Amman'}$ (Student) =
  Select * from Student where name = 'Ahmad' or address = 'Amman';

Id	Name	Address
1108	Ali	Amman
1453	Ahmad	Salt
1002	Omar	Amman

- Select Operation Properties:
  $\sigma_{condition1}$ ($\sigma_{condition2}$ (R)) = $\sigma_{condition2}$ ($\sigma_{condition1}$ (R)) = $\sigma_{condition1 \wedge condition2}$ (R)

Project Operator (π)

- Selecting a subset of the attributes of a relation by specifying the name of the required attributes.
- The Project creates a vertical partitioning.
  Syntax:

  $\pi_{<Attribute\ list>}$ (R)

  - The symbol π (pi) is used to denote the project operator.
  - <Attribute list> is the desired list of attributes from the attributes of relation R.
  - $\pi_{<Attribute\ list>}$ (R) is equivalent to "Select Distinct *Attribute_List* from R;"
- The project operation removes any duplicate rows.
- Example: Consider the following relation r.

r

A	B	C	D
X	X	1	8
X	Y	5	7
Y	Y	3	3
Y	Y	12	10

$\pi_{A,B}(r)$ = Select Distinct A, B From r;

A	B
X	X
X	Y
Y	Y

- Project operation properties:
  - $\pi_{list1}(\pi_{list2}(r)) = \pi_{list1}(r)$, where list2 contains the attributes of list1
  - The number of rows in the result of projection $\pi_{list}(r)$ is always less or equal to the number of rows in r.
  - If the list of attributes includes a key of r, the number of rows is equal to the number of rows in r.

- Composition of Relational Operations (Expression)
  - Relational algebra operations can be composed together into a relational algebra expression.
  - Example:

    $\pi_B(\sigma_{C>=3}(r))$ = Select Distinct B From r Where C >= 3;

B
Y

Assignment Operator (⟵)

- We may want to apply several relational algebra operations one after 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.
- Example:

  $r1 \leftarrow \sigma_{C>=3}(r)$

  $\pi_B(r1)$

B
Y

Rename Operator (ρ)

- Allows us to refer to a relation by more than one name.

  Syntax:

  1. $\rho_x(R)$

     Returns the expression R under the name X
  2. $\rho_{x(A1, A2, .., An)}(R)$

     If a relational-algebra expression R has arity n, then returns the result of expression R under the name X, and with the attributes renamed to $A_1, A_2, ..., A_n$.
  3. $\rho_{(A1, A2, .., An)}(R)$

Rename the attributes names without changing the relation name

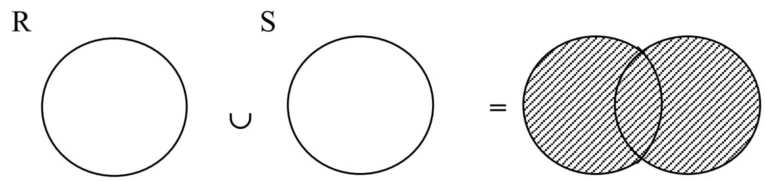
8.3 Binary Relational Operations

Set Operators

- Relation is a set of tuples, so set operations should apply: $\cap$, $\cup$, – (set difference)
- Result of combining two relations with a set operator is a relation (all its elements must be tuples having same structure).

1. **Union Operator ($\cup$)**

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



   - For R $\cup$ S to be valid. (The two operands must be "type compatible")
     1. R, S must have the same arity (same number of attributes).
     2. The attribute domains must be compatible (example: 2nd column of R deals with the same type of values as does the 2nd column of S)

   - R $\cup$ S is equivalent to "Select * From R Union Select * From S;"
   - Example:

     Tables:

     Person (*SSN, Name, Address, Hobby*)

     Professor (*Id, Name, Office, Phone*)

     are **not** union compatible.

     But

     π_{Name} (Person) and π_{Name} (Professor)

     are union compatible so

     π_{Name} (Person) $\cup$ π_{Name} (Professor)

     makes sense.

   - Example: **r**

A	B
X	1
X	2
Y	1

s

A	B
X	2
Y	3

$r \cup s$ = Select * From r Union Select * from s;

A	B
X	1
X	2
Y	1
Y	3

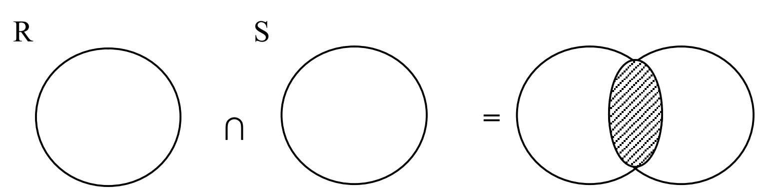
- Example: retrieve the SSNs of all employees who either work in department 5 or directly supervise an employee who works in department 5.

Employee	SSN	EName	Sal	SuperSSN	DNo

$Dep5\_Emps \leftarrow \sigma_{DNo=5}\ (Employee)$
$Result1 \leftarrow \pi_{SSN}\ (Dep5\_Emps)$
$Result2 \leftarrow \pi_{SuperSSN}\ (Dep5\_Emps)$
$Result \leftarrow Result1 \cup Result2$

2. Intersection Operator ($\cap$)

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



- R ∩ S is equivalent to "Select * From R Intersect Select * From S;"

- Example:

r

A	B
X	1
X	2
Y	1

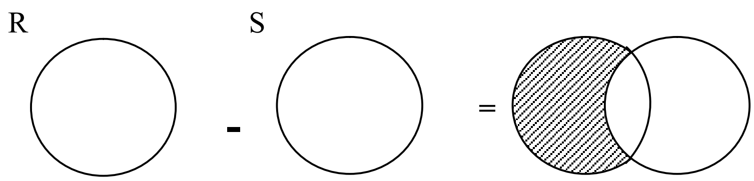
s

A	B
X	2
Y	3

r ∩ s = Select * from r Intersect Select * from s;

A	B
X	2

3. **Intersection Operator (-)**
   - The result of this operation, denoted by R - S, is a relation that includes all rows that are in R but not in S. the two operands must be "type compatible".
   - Example:



   - R - S is equivalent to "Select * From R Minus Select * From S;"
   - Example:

r

A	B
X	1
X	2
Y	1

s

A	B
X	2
Y	3

r – s = Select * from r Minus Select * from s;

A	B
X	1
Y	1

s - r

A	B
Y	3

- **Set Operators Properties:**
  - The union and the intersection are commutative operations

    $R \cup S = S \cup R$, and $R \cap S = S \cap R$
  - The union and the intersection are associative operations

    $R \cup (S \cup T) = (R \cup S) \cup T$, and $R \cap (S \cap T) = (R \cap S) \cap T$
  - The set difference operation is not commutative operation

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

Cartesian (or Cross Product) Operator (×)

- If R and S are two relations, R × S is the set of all concatenated rows <x,y>, where x is a row in R and y is a row in S
  - *R* and *S* need not be type compatible
- $R \times S$ is expensive to compute:
  - Factor of two in the size of each row
  - Quadratic in the number of rows
- If R has n_R rows (denoted as $|R| = n_R$), and S has n_S rows, then R x S will have $n_R * n_S$ rows.
- R × S is equivalent to "Select * From R, S;"
  - Example:

r

A	B
X	1
Y	2

s

C	D	E
X	10	E1
Y	10	E1
Y	20	E2
Z	10	E2

r × s = Select * from r,s;

A	B	C	D	E
X	1	X	10	E1
X	1	Y	10	E1
X	1	Y	20	E2
X	1	Z	10	E2
Y	2	X	10	E1
Y	2	Y	10	E1
Y	2	Y	20	E2
Y	2	Z	10	E2

$\sigma_{A=C}$ (r × s) = select * from r,s where A=C;

A	B	C	D	E
X	1	X	10	E1
Y	2	Y	10	E1
Y	2	Y	20	E2

- Generally, CROSS PRODUCT is not a meaningful operation

- Can become meaningful when followed by other operations
- Example (not meaningful):

Employee

SSN	FName	LName	Gender	SuperSSN	DNo

Dependent

ESSN	Dependent_Name	Gender	BDate	Relationship

$\text{Female\_Emps} \leftarrow \sigma_{\text{Gender='F'}}(\text{Employee})$
$\text{EmpNames} \leftarrow \pi_{\text{FNAME, LNAME, SSN}}(\text{Female\_Emps})$
$\text{Emp\_Dependents} \leftarrow \text{EmpNames x Dependent}$

- Emp_Dependents will contain every combination of EmpNames and Dependent
- whether or not they are actually related
- To keep only combinations where the Dependent is related to the Employee, we add a SELECT operation as follows
  $\text{Actual\_Deps} \leftarrow \sigma_{\text{SSN=ESSN}}(\text{Emp\_Dependents})$
  $\text{Result} \leftarrow \pi_{\text{FNAME, LNAME, DEPENDENT\_NAME}}(\text{Actual\_Deps})$
- Result will now contain the name of female employees and their dependents

- Example: Display employees names for employees who work in accounting department

Employee

SSN	Ename	Sal	Gender	SuperSSN	DNo

Department

DNo	DName	Location

$\pi_{\text{ENAME}}(\sigma_{\text{employee.dno=department.dno}}(\sigma_{\text{dname='Accounting'}}(\text{Employee} \times \text{Department}))) =$

Select Ename from Employee, Department
where employee.dno = department.dno and dname = 'Accounting';

- Example: find the names of all customers who live on the same street and in the same city as smith

Customer

SSN	Cname	City	Street

$\pi_{\text{CName}}(\sigma_{\text{street = s} \wedge \text{city = c}}($
$\text{Customer} \times (\rho_{\text{smith\_add(s,c)}}(\pi_{\text{street, city}}(\sigma_{\text{CName = 'Smith'}}(\text{Customer}))))) =$

Select c1.CName from customer c1, customer c2
where c2.CName='Smith' and c1.city = c2.city and c1.street = c2.street;

- Example: find the largest account balance in the bank

Account

AccNo	Balance	Date

π_{Balance} (Account) –
$\pi_{\text{Account.Balance}}$ ($\sigma_{\text{account.balance} < \text{d.balance}}$ (Account × (ρ_d (Account))))

Join Operator

- The JOIN operation is used to combine related rows from two relations into a single row.

1. **Theta Join Operator ($\bowtie_{\square}$)**

  - The Theta-Join is a specialized product containing only pairs that match on a supplied condition called join-condition.
  - A theta join of R and S is the expression
    $R \bowtie_{\square} S$
    where $\square$ is a conjunction of terms: A_i oper B_i
    in which A_i is an attribute of R; B_i is an attribute of S; and oper is one of =, <, >, ≥ ≠, ≤.
  - $R \bowtie_{\square} S = \sigma_{\square} (R \times S)$ = Select * from R,S where $\square$;

  - Example: **R**

A	B	C	D
a	1	x	4
b	2	y	5
c	4	z	4
d	8	x	5
e	1	y	4

S

E	F	G
5	a	x
4	b	y
3	c	y
2	a	x

$R \bowtie_{A <> \text{'a'} \wedge D < E} S$

A	B	C	D	E	F	G
c	4	z	4	5	a	x
e	1	y	4	5	a	x

$R \bowtie_{B = E} S$ (Equi_Join)

A	B	C	D	E	F	G
b	2	y	5	2	a	x
c	4	z	4	4	b	y

- Example: Display the names of all employees who earn more than their managers.

Employee

<u>ID</u>	Ename	Salary	MgrId

Manager

<u>ID</u>	MName	Salary

$$\pi_{EName}(\text{Employee} \bowtie_{MgrId=Manager.Id \text{ AND } Employee.Salary>Manager.Salary} \text{Manager})$$

2. Natural Join Operator ($\bowtie$)

- Special case of Equi_Join.
- 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.
- Natural join removes duplicate attributes.
- $r \bowtie s = \pi_{\text{Attribute\_List}} (\sigma_{\text{Join\_Condition}} (r \times s))$

  where

  Attribute_List = *attributes* (r) $\cup$ *attributes* (s)

  (duplicates are eliminated) and *Join-Condition* has the form:

  $A_1 = A_1$ AND … AND $A_n = A_n$

  where $\{A_1 \ldots A_n\} = attributes(r) \cap attributes(s)$

- Note: let r(R) and s(S) be relations without any attributes in common; that is,

  $R \cap S = \square$. Then $r \bowtie s = r \times s$.

- Example: **R**

A	B	C	D
a	1	x	4
b	2	y	5
c	4	z	4
d	8	x	5
e	1	y	4

S

E	F	G
5	a	x
4	b	y
3	c	y
2	a	x

R $\bowtie \rho_{(B, F, G)}$ **(S)** = Select R.*, F, G from R, S where B=E;

A	B	C	D	F	G
b	2	y	5	a	x
c	4	z	4	b	y

- Example: **R**

A	B	C	D
x	1	x	a
y	2	z	a
z	4	y	b
x	1	z	a
w	2	y	b

S

B	D	E
1	a	x
3	a	y
1	a	z
2	b	w
3	b	e

$$\mathbf{R \bowtie S = \pi_{R.A,\ R.B,\ R.C,\ R.D,\ S.E}\ (\sigma_{R.B = S.B \wedge R.D = S.D}\ (R \times S))}$$

A	B	C	D	E
x	1	x	a	x
x	1	x	a	z
x	1	z	a	x
x	1	z	a	z
w	2	y	b	w

Complete Set of Relational Operations

- The set of operations including π (Projection), σ (Selection), – (Difference), ρ (Rename), (Union) and $\times$ (Cartesian Product) is called a complete set, because any other relational algebra expression can be expressed by combination of these five operations.
  - $R \cap S = (R \cup S) - ((R - S) \cup (S - R))$
  - $R \cap S = R - (R - S)$
  - $R \bowtie_{Condition} S = \sigma_{Condition} (R \times S)$

Examples of Queries in Relational Algebra

8.4 Examples of Queries in Relational Algebra

Banking Example:

Branch (branch_name, branch_city)

Customer (customer_name, customer_city, customer_street)

Account (account_number, branch_name, balance)

Loan (loan_number, branch_name, amount)

Depositor (customer_name, account_number)

Borrower (customer_name, loan_number)

Q1: Find all loans of over $1200.

Q2: Find the loan number for each loan of an amount greater than $1200.

Q3: Find the names of all customers who have a loan, an account, or both from the bank.

Q4: Find the names of all customers who have a loan and an account at bank.

Q5: Find the names of all customers who have a loan at the Perryridge branch.

Q6: Find the names of all customers who have a loan at the Perryridge branch but do not have an account at any branch of the bank.

Q7: Find the names of all branches with customers who have an account in the bank and who live in Harrison.

Q8: Find all customers who have an account from at least the "Downtown" and the "uptown" branches.

Company Example:

Employee (fname, minit, lname, SSN, address, sex, salary, superSSN, DNo)

Department (Dname, Dnumber, MGRSSN, MGRStartDate)

Dept_Locations (DNumber, DLocation)

Project (PName, PNumber, PLocation, DNum)

Works_On (ESSN, PNo, Hours)

Dependent (ESSN, Dependent_Name, Sex, BDate,Relationship)

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

Q2: for every project located in 'Stafford', list the project number, the controlling department number, and the department manager's last name, address, and birthdate.

Q3: make a list of project numbers for projects that involve an employee whose last name is 'smith', either as a worker or as a manager of the department that controls the project.

Q4: Retrieve the names of employees who have no dependents.

Q5: List the names of managers who have at least one dependent.