

Database Management System 11

Relational Algebra

Query Language

Relational Algebra

SELECT Operator(σ)PROJECT
Operator(π)Composition of Relational
OperatorsRENAME Operator(ρ)

Union Compatibility

UNION Operator(\cup)DIFFERENCE
Operator($-$)Cartesian Product
Operator(\times)Intersection
Operator(\cap)JOIN Operator(\bowtie)Division Operator(\div)Assignment
Operator(\leftarrow)

Chittaranjan Pradhan
School of Computer Engineering,
KIIT University

Language in which user requests information from the database are:

- Procedural language
- Nonprocedural language

The categories of different languages are:

- SQL
- Relational Algebra
- Relational Calculus
 - Tuple Relational Calculus
 - Domain Relational Calculus

Query Language

Relational Algebra

SELECT Operator(σ)PROJECT
Operator(π)Composition of Relational
OperatorsRENAME Operator(ρ)

Union Compatibility

UNION Operator(\cup)DIFFERENCE
Operator($-$)Cartesian Product
Operator(\times)Intersection
Operator(\cap)JOIN Operator(\bowtie)Division Operator(\div)Assignment
Operator(\leftarrow)

Relational Algebra

Relational algebra is a procedural language for manipulating relations. Relational algebra operations manipulate relations. That is, these operations use one or two existing relations to create a new relation

- Fundamental operators
 - Unary: SELECT, PROJECT, RENAME
 - Binary: UNION, SET DIFFERENCE, CARTESIAN PRODUCT
- Secondary operators
 - INTERSECTION, NATURAL JOIN, DIVISION, and ASSIGNMENT

Query Language

Relational Algebra

SELECT Operator(σ)PROJECT
Operator(π)Composition of Relational
OperatorsRENAME Operator(ρ)

Union Compatibility

UNION Operator(\cup)DIFFERENCE
Operator($-$)Cartesian Product
Operator(\times)Intersection
Operator(\cap)JOIN Operator(\bowtie)Division Operator(\div)Assignment
Operator(\leftarrow)

The relational schemas used for different operations are:

- **Customer**(cust_name, cust_street, cust_city)
 - used to store customer details
- **Branch**(branch_name, branch_city, assets)
 - used to store branch details
- **Account**(acc_no, branch_name, balance)
 - stores the account details
- **Loan**(loan_no, branch_name, amount)
 - stores the loan details
- **Depositor**(cust_name, acc_no)
 - stores the details about the customers' account
- **Borrower**(cust_name, loan_no)
 - used to store the details about the customers' loan

Query Language

Relational Algebra

SELECT Operator(σ)PROJECT
Operator(π)Composition of Relational
OperatorsRENAME Operator(ρ)

Union Compatibility

UNION Operator(\cup)DIFFERENCE
Operator($-$)Cartesian Product
Operator(\times)Intersection
Operator(\cap)JOIN Operator(\bowtie)Division Operator(\div)Assignment
Operator(\leftarrow)

SELECT Operator(σ)

SELECT Operator(σ)

SELECT operation is used to create a relation from another relation by selecting only those tuples or rows from the original relation that satisfy a specified condition. It is denoted by sigma (σ) symbol. The predicate appears as a subscript to σ

The argument relation is in parenthesis after the σ . The result is a relation that has the same attributes as the relation specified in <relation-name>. The general syntax of select operator is:

σ <selection-condition> (**<relation name>**)

Query: Find the details of the loans taken from 'Bhubaneswar Main' branch.

σ *branch_name='BhubaneswarMain'* (Loan)

- The operators used in selection predicate may be: =, \neq , <, \leq , >, \geq .
- Different predicates can be combined into a larger predicate by using the connectors like: AND(\wedge), OR(\vee), NOT(\neg)

SELECT Operator(σ)...

Loan

| loan_no | branch_name | amount |
|---------|------------------|----------------|
| L201 | Bhubaneswar Main | 50,000,000.00 |
| L202 | Bhubaneswar Main | 5,000,000.00 |
| L203 | Mumbai Main | 100,000,000.00 |
| L204 | Juhu | 60,000,000.00 |

 $\sigma_{branch_name='BhubaneswarMain'} (Loan)$

| loan_no | branch_name | amount |
|---------|------------------|---------------|
| L201 | Bhubaneswar Main | 50,000,000.00 |
| L202 | Bhubaneswar Main | 5,000,000.00 |

 $\sigma_{branch_name='BhubaneswarMain' \text{ AND } amount > 10,000,000} (Loan)$

| loan_no | branch_name | amount |
|---------|------------------|---------------|
| L201 | Bhubaneswar Main | 50,000,000.00 |

PROJECT Operator(π)

PROJECT Operator(π)

PROJECT operation can be thought of as eliminating unwanted columns. It eliminates the duplicate rows. It is denoted by $\pi(\pi)$ symbol. The attributes needed to be appeared in the resultant relation appear as subscript to π .

The argument relation follows in parenthesis. The general syntax of project operator is:

$\pi_{\langle \text{attribute-list} \rangle} (\langle \text{relation name} \rangle)$

Query: Find the loan numbers and respective loan amounts.

$\pi_{\text{loan_no, amount}} (\text{Loan})$

PROJECT Operator(π)...

Query Language

Relational Algebra

SELECT Operator(σ)**PROJECT
Operator(π)**Composition of Relational
OperatorsRENAME Operator(ρ)

Union Compatibility

UNION Operator(\cup)DIFFERENCE
Operator($-$)Cartesian Product
Operator(\times)Intersection
Operator(\cap)JOIN Operator(\bowtie)Division Operator(\div)Assignment
Operator(\leftarrow)

Loan

| <u>loan_no</u> | branch_name | amount |
|----------------|------------------|----------------|
| L201 | Bhubaneswar Main | 50,000,000.00 |
| L202 | Bhubaneswar Main | 5,000,000.00 |
| L203 | Mumbai Main | 100,000,000.00 |
| L204 | Juhu | 60,000,000.00 |

 $\pi_{loan_no, amount} (Loan)$

| <u>loan_no</u> | amount |
|----------------|----------------|
| L201 | 50,000,000.00 |
| L202 | 5,000,000.00 |
| L203 | 100,000,000.00 |
| L204 | 60,000,000.00 |

Composition of Relational Operators

Composition of Relational Operators

Relational algebra operators can be composed together into a relational algebra expression to answer the complex queries

Q: Find the name of the customers who live in Bhubaneswar

| Customer | | |
|------------------|---------------|-------------|
| <u>cust_name</u> | cust_street | cust_city |
| Rishi | India Gate | New Delhi |
| Sarthak | M. G. Road | Bangalore |
| Manas | Shastri Nagar | Bhubaneswar |
| Ramesh | M. G. Road | Bhubaneswar |
| Mahesh | Juhu | Mumbai |

$\pi_{\text{cust_name}} (\sigma_{\text{cust_city} = 'Bhubaneswar'} (\text{Customer}))$

| cust_name |
|-----------|
| Manas |
| Ramesh |

Query Language

Relational Algebra

SELECT Operator(σ)PROJECT
Operator(π)Composition of Relational
OperatorsRENAME Operator(ρ)

Union Compatibility

UNION Operator(\cup)DIFFERENCE
Operator($-$)Cartesian Product
Operator(\times)Intersection
Operator(\cap)JOIN Operator(\Join)Division Operator(\div)Assignment
Operator(\leftarrow)

RENAME Operator(ρ)

RENAME Operator(ρ)

The results of relational algebra expressions do not have a name that can be used to refer them. It is useful to be able to give them names; the rename operator is used for this purpose. It is denoted by rho(ρ) symbol.

The general syntax of rename operator is:

$\rho \times (\mathbf{E})$

Assume E is a relational-algebra expression with arity n. The second form of rename operation is: $\rho \times (b_1, b_2, \dots, b_n) (\mathbf{E})$

$\pi_{cust_name} (\sigma_{cust_city='Bhubaneswar'} (\text{Customer}))$ can be written as:

1. $\rho_{Customer_Bhubaneswar} (\sigma_{cust_city='Bhubaneswar'} (\text{Customer}))$

2. $\pi_{cust_name} (\text{Customer_Bhubaneswar})$

Query Language

Relational Algebra

SELECT Operator(σ)PROJECT
Operator(π)Composition of Relational
OperatorsRENAME Operator(ρ)

Union Compatibility

UNION Operator(\cup)DIFFERENCE
Operator($-$)Cartesian Product
Operator(\times)Intersection
Operator(\cap)JOIN Operator(\bowtie)Division Operator(\div)Assignment
Operator(\leftarrow)

RENAME Operator(ρ)...

The different forms of the rename operation for renaming the relation are:

a. $\rho_S(\mathbf{R})$

b. $\rho_{S(b_1, b_2, \dots, b_n)}(\mathbf{R})$

c. $\rho_{(b_1, b_2, \dots, b_n)}(\mathbf{R})$

For example, the attributes of Customer (cust_name, cust_street, cust_city) can be renamed as:

$\rho_{(name, street, city)}(\mathbf{Customer})$

Union Compatibility

To perform the set operations such as UNION, DIFFERENCE and INTERSECTION, the relations need to be **union compatible** for the result to be a valid relation

Two relations $R_1(a_1, a_2, \dots, a_n)$ and $R_2(b_1, b_2, \dots, b_m)$ are union compatible iff:

- $n = m$, i.e. both relations have same arity
- $\text{dom}(a_i) = \text{dom}(b_i)$ for $1 \leq i \leq n$

Query Language

Relational Algebra

SELECT Operator(σ)PROJECT
Operator(π)Composition of Relational
OperatorsRENAME Operator(ρ)

Union Compatibility

UNION Operator(\cup)DIFFERENCE
Operator($-$)Cartesian Product
Operator(\times)Intersection
Operator(\cap)JOIN Operator(\bowtie)Division Operator(\div)Assignment
Operator(\leftarrow)

UNION Operator(\cup)

UNION Operator(\cup)

The union operation is used to combine data from two relations. It is denoted by union(\cup) symbol. The union of two relations $R_1(a_1, a_2, \dots, a_n)$ and $R_2(b_1, b_2, \dots, b_n)$ is a relation $R_3(c_1, c_2, \dots, c_n)$ such that:

$$\text{dom}(c_i) = \text{dom}(a_i) \cup \text{dom}(b_i), 1 \leq i \leq n$$

$R_1 \cup R_2$ is a relation that includes all tuples that are either present in R_1 or R_2 or in both without duplicate tuples

| Depositor | | Borrower | |
|------------------|---------------|------------------|----------------|
| <u>cust_name</u> | <u>acc_no</u> | <u>cust_name</u> | <u>loan_no</u> |
| Manas | A101 | Ramesh | L201 |
| Ramesh | A102 | Ramesh | L202 |
| Rishi | A103 | Mahesh | L203 |
| Mahesh | A104 | Rishi | L204 |
| Mahesh | A105 | | |

$$\pi_{\text{cust_name}}(\text{Depositor}) \cup \pi_{\text{cust_name}}(\text{Borrower})$$

| <u>cust_name</u> |
|------------------|
| Manas |
| Ramesh |
| Rishi |
| Mahesh |

DIFFERENCE Operator(-)

DIFFERENCE Operator(-)

The difference operation is used to identify the rows that are in one relation and not in another. It is denoted as (-) symbol. The difference of two relations $R_1(a_1, a_2, \dots, a_n)$ and $R_2(b_1, b_2, \dots, b_n)$ is a relation $R_3(c_1, c_2, \dots, c_n)$ such that:

$\text{dom}(c_i) = \text{dom}(a_i) - \text{dom}(b_i)$, $1 \leq i \leq n$

$R_1 - R_2$ is a relation that includes all tuples that are in R_1 , but not in R_2

| Depositor | | Borrower | |
|------------------|---------------|------------------|----------------|
| <u>cust_name</u> | <u>acc_no</u> | <u>cust_name</u> | <u>loan_no</u> |
| Manas | A101 | Ramesh | L201 |
| Ramesh | A102 | Ramesh | L202 |
| Rishi | A103 | Mahesh | L203 |
| Mahesh | A104 | Rishi | L204 |
| Mahesh | A105 | | |

$\pi_{\text{cust_name}}(\text{Depositor}) - \pi_{\text{cust_name}}(\text{Borrower})$

| <u>cust_name</u> |
|------------------|
| Manas |

Cartesian Product Operator(\times)

Cartesian Product Operator(\times)

The Cartesian product of two relations $R_1(a_1, a_2, \dots, a_n)$ with cardinality i and $R_2(b_1, b_2, \dots, b_m)$ with cardinality j is a relation R_3 with

- degree $k = n + m$,
- cardinality $i * j$ and
- attributes $(a_1, a_2, \dots, a_n, b_1, b_2, \dots, b_m)$

$R_1 \times R_2$ is a relation that includes all the possible combinations of tuples from R_1 and R_2 . The Cartesian product is used to combine information from any two relations. It is not a useful operation by itself; but is used in conjunction with other operations.

Query Language

Relational Algebra

SELECT Operator(σ)PROJECT
Operator(π)Composition of Relational
OperatorsRENAME Operator(ρ)

Union Compatibility

UNION Operator(\cup)DIFFERENCE
Operator($-$)Cartesian Product
Operator(\times)Intersection
Operator(\cap)JOIN Operator(\bowtie)Division Operator(\div)Assignment
Operator(\leftarrow)

Cartesian Product Operator(\times)...

Borrower Loan

| cust_name | loan_no | loan_no | branch_name | amount |
|-----------|---------|---------|------------------|----------------|
| Ramesh | L201 | L201 | Bhubaneswar Main | 50,000,000.00 |
| Ramesh | L202 | L202 | Bhubaneswar Main | 5,000,000.00 |
| Mahesh | L203 | L203 | Mumbai Main | 100,000,000.00 |
| Rishi | L204 | L204 | Juhu | 60,000,000.00 |

Borrower \times Loan

| cust_name | Borrower.loan_no | Loan.loan_no | branch_name | amount |
|-----------|------------------|--------------|------------------|----------------|
| Ramesh | L201 | L201 | Bhubaneswar Main | 50,000,000.00 |
| Ramesh | L201 | L202 | Bhubaneswar Main | 5,000,000.00 |
| Ramesh | L201 | L203 | Mumbai Main | 100,000,000.00 |
| Ramesh | L201 | L204 | Juhu | 60,000,000.00 |
| Ramesh | L202 | L201 | Bhubaneswar Main | 50,000,000.00 |
| Ramesh | L202 | L202 | Bhubaneswar Main | 5,000,000.00 |
| Ramesh | L202 | L203 | Mumbai Main | 100,000,000.00 |
| Ramesh | L202 | L204 | Juhu | 60,000,000.00 |
| Mahesh | L203 | L201 | Bhubaneswar Main | 50,000,000.00 |
| Mahesh | L203 | L202 | Bhubaneswar Main | 5,000,000.00 |
| Mahesh | L203 | L203 | Mumbai Main | 100,000,000.00 |
| Mahesh | L203 | L204 | Juhu | 60,000,000.00 |
| Rishi | L204 | L201 | Bhubaneswar Main | 50,000,000.00 |
| Rishi | L204 | L202 | Bhubaneswar Main | 5,000,000.00 |
| Rishi | L204 | L203 | Mumbai Main | 100,000,000.00 |
| Rishi | L204 | L204 | Juhu | 60,000,000.00 |

Query Language

Relational Algebra

SELECT Operator(σ)

PROJECT
Operator(π)

Composition of Relational
Operators

RENAME Operator(ρ)

Union Compatibility

UNION Operator(\cup)

DIFFERENCE
Operator($-$)

Cartesian Product
Operator(\times)

Intersection
Operator(\cap)

JOIN Operator(\bowtie)

Division Operator(\div)

Assignment
Operator(\leftarrow)

Cartesian Product Operator(\times)...

Query: Find out the customer and their loan details taken from Bhubaneswar Main branch.

Ans: $\sigma_{branch_name='BhubaneswarMain' \wedge Borrower.loan_no=Loan.loan_no}$
(Borrower \times Loan)

| cust_name | Borrower.loan_no | Loan.loan_no | branch_name | amount |
|-----------|------------------|--------------|------------------|---------------|
| Ramesh | L201 | L201 | Bhubaneswar Main | 50,000,000.00 |
| Ramesh | L202 | L202 | Bhubaneswar Main | 5,000,000.00 |

[Query Language](#)[Relational Algebra](#)[SELECT Operator\(\$\sigma\$ \)](#)[PROJECT Operator\(\$\pi\$ \)](#)[Composition of Relational Operators](#)[RENAME Operator\(\$\rho\$ \)](#)[Union Compatibility](#)[UNION Operator\(\$\cup\$ \)](#)[DIFFERENCE Operator\(\$-\$ \)](#)[Cartesian Product Operator\(\$\times\$ \)](#)[Intersection Operator\(\$\cap\$ \)](#)[JOIN Operator\(\$\bowtie\$ \)](#)[Division Operator\(\$\div\$ \)](#)[Assignment Operator\(\$\leftarrow\$ \)](#)

Intersection Operator(\cap)

Intersection Operator(\cap)

The intersection operation is used to identify the rows that are common to two relations. It is denoted by (\cap) symbol. The intersection of two relations $R_1(a_1, a_2, \dots, a_n)$ and $R_2(b_1, b_2, \dots, b_n)$ is a relation $R_3(c_1, c_2, \dots, c_n)$ such that:

$\text{dom}(c_i) = \text{dom}(a_i) \cap \text{dom}(b_i), 1 \leq i \leq n$

$R_1 \cap R_2$ is a relation that includes all tuples that are present in both R_1 and R_2

The intersection operation can be rewritten by a pair of set difference operations as **$R \cap S = R - (R - S)$**

| Depositor | | Borrower | |
|------------------|---------------|------------------|----------------|
| <u>cust_name</u> | <u>acc_no</u> | <u>cust_name</u> | <u>loan_no</u> |
| Manas | A101 | Ramesh | L201 |
| Ramesh | A102 | Ramesh | L202 |
| Rishi | A103 | Mahesh | L203 |
| Mahesh | A104 | Rishi | L204 |
| Mahesh | A105 | | |

$\pi_{\text{cust_name}}(\text{Depositor}) \cap \pi_{\text{cust_name}}(\text{Borrower})$

| <u>cust_name</u> |
|------------------|
| Ramesh |
| Mahesh |
| Rishi |

JOIN Operator(\bowtie)

JOIN Operator(\bowtie)

The join is a binary operation that is used to combine certain selections and a Cartesian product into one operation. It is denoted by join (\bowtie) symbol.

The join operation forms a Cartesian product of its two arguments, performs a selection forcing equality on those attributes that appear in both relations, and finally removes the duplicate attributes

Query: Find the names of customers who have a loan at the bank, along with the loan number and the loan amount.

Ans: This query can be solved by using the PROJECT, SELECT and CARTESIAN PRODUCT operators as:

$\pi_{cust_name, Loan.loan_no, amount} (\sigma_{Borrower.loan_no=Loan.loan_no} (Borrower \times Loan))$

This same expression can be simplified by using the JOIN as:

$\pi_{cust_name, loan_no, amount} (Borrower \bowtie Loan)$

JOIN Operator(\bowtie)...

Query Language

Relational Algebra

SELECT Operator(σ)PROJECT
Operator(π)Composition of Relational
OperatorsRENAME Operator(ρ)

Union Compatibility

UNION Operator(\cup)DIFFERENCE
Operator($-$)Cartesian Product
Operator(\times)Intersection
Operator(\cap)JOIN Operator(\bowtie)Division Operator(\div)Assignment
Operator(\leftarrow)

Borrower Loan

| <u>cust_name</u> | <u>loan_no</u> | <u>loan_no</u> | branch_name | amount |
|------------------|----------------|----------------|------------------|----------------|
| Ramesh | L201 | L201 | Bhubaneswar Main | 50,000,000.00 |
| Ramesh | L202 | L202 | Bhubaneswar Main | 5,000,000.00 |
| Mahesh | L203 | L203 | Mumbai Main | 100,000,000.00 |
| Rishi | L204 | L204 | Juhu | 60,000,000.00 |

$\pi_{\text{cust_name, loan_no, amount}} (\text{Borrower} \bowtie \text{Loan})$

| <u>cust_name</u> | <u>loan_no</u> | amount |
|------------------|----------------|----------------|
| Ramesh | L201 | 50,000,000.00 |
| Ramesh | L202 | 5,000,000.00 |
| Mahesh | L203 | 100,000,000.00 |
| Rishi | L204 | 60,000,000.00 |

Division Operator(\div)

Division Operator(\div)

The division operation creates a new relation by selecting the rows in one relation that match every row in another relation. The division operation requires that we look at an entire relation at once. It is denoted by division (\div) symbol

Let A, B, C are three relations and we desire $B \div C$ to give A as the result. This operation is possible iff:

- The columns of C must be a subset of the columns of B. The columns of A are all and only those columns of B that are not columns of C
- A row is placed in A if and only if it is associated with B and with every row of C

The division operation is the reverse of the Cartesian product operation as: $B = (B \times C) \div C$

Division operator is suited to queries that include the phrase **every** or **all** as part of the condition

Query Language

Relational Algebra

SELECT Operator(σ)PROJECT
Operator(π)Composition of Relational
OperatorsRENAME Operator(ρ)

Union Compatibility

UNION Operator(\cup)DIFFERENCE
Operator($-$)Cartesian Product
Operator(\times)Intersection
Operator(\cap)JOIN Operator(\bowtie)Division Operator(\div)Assignment
Operator(\leftarrow)

Division Operator(\div)...

| Depositor | | Account | | |
|-----------|--------|---------|------------------|---------------|
| cust_name | acc_no | acc_no | branch_name | balance |
| Manas | A101 | A101 | Bhubaneswar Main | 100,000.00 |
| Ramesh | A102 | A102 | Shastri Nagar | 50,000.00 |
| Rishi | A103 | A103 | India Gate | 5,000,000.00 |
| Mahesh | A104 | A104 | Juhu | 600,000.00 |
| Mahesh | A105 | A105 | Mumbai Main | 10,000,000.00 |

Branch

| branch_name | branch_city | assets |
|------------------|-------------|-----------|
| Bhubaneswar Main | Bhubaneswar | Gold |
| Shastri Nagar | Bhubaneswar | Mines |
| India Gate | New Delhi | Gold |
| Juhu | Mumbai | Sea Shore |
| Mumbai Main | Mumbai | Movie |

Query: Find all the customers who have an account at all the branches located in Mumbai

$\pi_{\text{cust_name, branch_name}} (\text{Depositor} \bowtie \text{Account}) \div \pi_{\text{branch_name}} (\sigma_{\text{branch_city}='Mumbai'} (\text{Branch}))$

| cust_name |
|-----------|
| Mahesh |

Query Language

Relational Algebra

SELECT Operator(σ)

PROJECT Operator(π)

Composition of Relational Operators

RENAME Operator(ρ)

Union Compatibility

UNION Operator(\cup)

DIFFERENCE Operator($-$)

Cartesian Product Operator(\times)

Intersection Operator(\cap)

JOIN Operator(\bowtie)

Division Operator(\div)

Assignment Operator(\leftarrow)

Assignment Operator(\leftarrow)

Assignment Operator(\leftarrow)

It works like assignment in a programming language. In relational algebra, the assignment operator gives a name to a relation. It is denoted by (\leftarrow) symbol

Assignment must always be made to a temporary relation variable. The result of the right of the \leftarrow symbol is assigned to the relation variable on the left of the \leftarrow symbol

With the assignment operator, a query can be written as a sequential program consisting of:

- a series of assignment,
- followed by an expression whose value is displayed as a result of the query

$\pi_{cust_name, branch_name} (Depositor \bowtie Account) \div \pi_{branch_name} (\sigma_{branch_city='Mumbai'} (Branch))$
can be simplified as:

Temp1 $\leftarrow \pi_{cust_name, branch_name} (Depositor \bowtie Account)$

Temp2 $\leftarrow \pi_{branch_name} (\sigma_{branch_city='Mumbai'} (Branch))$

Result = Temp1 \div Temp2