DBMS: Lecture 2
(Relational Algebra)

By

Dr. Sumit Kumar Tetarave

Relational Algebra

- Relational Algebra is a set of basic operations used to manipulate the data in relational model.
- These operations enable the user to specify basic retrieval request.
- The result of retrieval is a new relation, formed from one or more relations.
- These operations can be classified in two categories: 1) Basic Set Operations and 2) Relational Operations
  - Basic Set Operations
    1) UNION (∪, ∨, or)
    2) INTERSECTION (∩, ∧, and)
    3) SET DIFFERENCE (-)
    4) CARTESIAN PRODUCT (X)

R0

C
C1
C2

R1

A	B
A1	B1
A2	B2
A3	B3
A4	B4

R2

X	Y
A1	B1
A7	B7
A2	B2
A4	B4

R3

A	B
A1	B1
A2	B2
A3	B3
A4	B4
A7	B7

R3 = R1 ∪ R2 = R2 ∪ R1
Commutative Operation
M = R ∪ (S ∪ T)
= (R ∪ S) ∪ T
Associative Operation

R4

A	B
A1	B1
A2	B2
A4	B4

R4 = R1 ∩ R2
Commutative &
Associative

R5= R1-R2

A	B
A3	B3

R6= R2-R1

A	B
A7	B7

R1-R2 ≠ R2-R1
Not Commutative
&
Not Associative

R7

A	B	C
A1	B1	C1
A1	B1	C2
A2	B2	C1
A2	B2	C2
A3	B3	C1
A3	B3	C2
A4	B4	C1
A4	B4	C2

R7 = R0 X R1
Commutative &
Associative

Degree (R7)
= Degree of (R0) +
Degree (R1).

Note:
These are the binary operations; i.e., each is applied to two sets or relations.
These relations should be union compatible except
- in the case of *Cartesian Product*.

- Two relations R (A_1, A_2,….., A_n) and S (B_1, B_2,……,B_n) are said to be union compatible
  - if they have the ***same degree n*** and domains of the corresponding attributes are also the same

Relational Operations: (1) SELECT, (2) PROJECT, (3) JOIN, (4) DIVISION

1) SELECT : The select operation is used to select some specific records from the database based on some criteria.
2) This is a unary operation mathematically denoted as σ.
   - **Syntax:**
     σ <Selection condition> **(Relation)**
   - The Boolean expression is specified in <Select condition> is made of a number of clauses of the form:
     - <attribute name><comparison operator><constant value> or
     - <attribute name><comparison operator><attribute name>
   - Comparison operators in the set { ≤, ≥, ≠, =, <, <} apply to the attributes whose domains are
     - ***ordered value* like integer.**

Example:

- Consider the relation PERSON. If you want to display details of persons having age less than or equal to 30 than the select operation will be used as follows:

  σ AGE <=30 **(PERSON)**

- The resultant relation will be as follows:

PERSON_ID	NAME	AGE	ADDRESS
3	Rakesh	28	Nandan Vihar

PERSON_ID	NAME	AGE	ADDRESS
1	Sumit	38	Sampark Vihar
2	Amit	35	Sailesh Vihar
3	Rakesh	28	Nandan Vihar

Note:

1) Select operation is commutative; i.e.,

   σ <condition1> (σ <condition2> (R)) = σ <condition2> (σ <condition1> (R))

   Hence, Sequence of select can be applied in any order

2) More than one condition can be applied using Boolean operators AND & OR etc.

Relational Operations…

(2) PROJECT

- The project operation is used to select the records with specified attributes
  - while discarding the others based on some specific criteria.
- This is denoted as Π .

Π List of attribute for project **(Relation)**

- **Example:**
- Consider the relation PERSON. If you want to display only the names of persons then
- the project operation will be used as follows:
- **Π Name (PERSON)**
- The resultant relation will be as follows.

NAME
Sumit
Amit
Rakesh

Note: -

- 1) Π <List1> (Π <list2> (R))= Π <list1> (R)
- As long as <list2> contains attributes in <list1>.

(4) DIVISION

- To perform the division operation R1 ÷ R2,
  - R2 should be a proper subset of R1.
- In the following example
  - R1 contains attributes A and B and
  - R2 contains only attribute B
- So, R2 is a proper subset of R1.
- If we perform R1 ÷ R2 than the resultant relation will contain those values of A from R1 that are-
  - related to all values of B present in R2.

Let R1

A	B
A1	B1
A1	B2
A2	B1
A3	B1
A4	B2
A3	B3
A3	B2

If R2

B
B1
B2

R1 ÷ R2

A
A1
A3

If R2

B
B1

R1 ÷ R2

A
A1
A2
A3

If R2

B

R1 ÷ R2

A
A1
A2
A3
A4

If R2

A

R1 ÷ R2

B
B1
B2
B3

Note:

Degree of relation: Degree (R ÷ S)
=Degree of R – Degree of S.

Relational Operations… (3) JOIN

- When we want to select **related tuples** from two given relation JOIN is used.
  - This is denoted as ⋈.
- The join operation requires that both the joined relations must have at least
  - one domain compatible attributes.
- Syntax:

  R1⋈<join condition>R2

  is used to combine related tuples from two relations R1 and R2 into a single tuple.
- <join condition> is of the form:

  <condition>**AND**<condition>**AND**……..**AND**<condition>.
- Degree of Relation:

  Degree (R1⋈<join condition>R2) <= Degree (R1) + Degree (R2).
- Example:

EMPLOYEE

EMP_CODE	EMP_NAME
101	Stephan
102	Jack
103	Harry

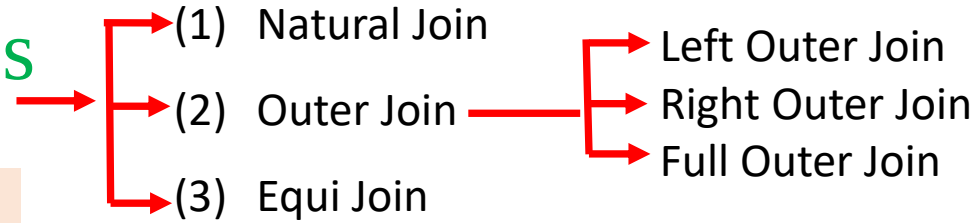
SALARY

EMP_CODE	SALARY
101	50000
102	30000
103	25000

Operation: (EMPLOYEE ⋈ SALARY)

EMP_CODE	EMP_NAME	SALARY
101	Stephan	50000
102	Jack	30000
103	Harry	25000

Relational Operations… JOIN Types



1. Natural Join:

- A natural join is the set of tuples of all combinations in R and S
  - that are equal on their common attribute names.
- It is denoted by ⋈.
- Example:

$\prod_{EMP\_NAME,\ SALARY}(EMPLOYEE \bowtie SALARY)$

EMP_NAME	SALARY
Stephan	50000
Jack	30000
Harry	25000

Syntax

SELECT EMPLOYEE. EMP_NAME, SALARY.SALARY FROM EMPLOYEE NATURAL JOIN SALARY;

2. Outer Join:

- The outer join operation is an extension of the join operation.
- It is used to deal with missing information.

3. Equi Join:

- A join that is formed as a result of the equality condition between the two same columns of multiple tables is called an Equi join.
- This is also known as simple join.
- The comparison operator to denote the equi join is "=".
- **Example:**

```
SELECT * FROM Student JOIN Marks ON Student.RollNo = Marks.RollNo;
```

A natural join is a kind of equi join that occurs when a common column with the same name in a different table gets compared and appears only once in the output.

EMPLOYEE

EMP_NAME	STREET	CITY
Ram	Civil line	Mumbai
Shyam	Park S	Kolkata
Ravi	M.G. S.	Delhi
Hari	Nehru N	Hyderabad

FACT_WORKERS

EMP_NAME	BRANCH	SALARY
Ram	Infosys	10000
Shyam	Wipro	20000
Kuber	HCL	30000
Hari	TCS	50000

OUTER JOIN

a. Left Outer Join
b. Right Outer Join
c. Full Outer Join

(R) EMPLOYEE

EMP_NAME	STREET	CITY
Ram	Civil line	Mumbai
Shyam	Park S	Kolkata
Ravi	M.G. S.	Delhi
Hari	Nehru N	Hyderabad

(S) FACT_WORKERS

EMP_NAME	BRANCH	SALARY
Ram	Infosys	10000
Shyam	Wipro	20000
Kuber	HCL	30000
Hari	TCS	50000

• a. Left outer join:

- Left outer join contains the set of tuples of all combinations in R and S that are equal on their common attribute names.
- In the left outer join, tuples in R have no matching tuples in S.
- It is denoted by ⟕.

EMP_NAME	STREET	CITY	BRANCH	SALARY
Ram	Civil line	Mumbai	Infosys	10000
Shyam	Park street	Kolkata	Wipro	20000
Hari	Nehru street	Hyderabad	TCS	50000
Ravi	M.G. Street	Delhi	NULL	NULL

• b. Right outer join:

- Right outer join contains the set of tuples of all combinations in R and S that are equal on their common attribute names.
- In right outer join, tuples in S have no matching tuples in R.
- It is denoted by ⟖.

EMP_NAME	BRANCH	SALARY	STREET	CITY
Ram	Infosys	10000	Civil line	Mumbai
Shyam	Wipro	20000	Park street	Kolkata
Hari	TCS	50000	Nehru street	Hyderabad
Kuber	HCL	30000	NULL	NULL

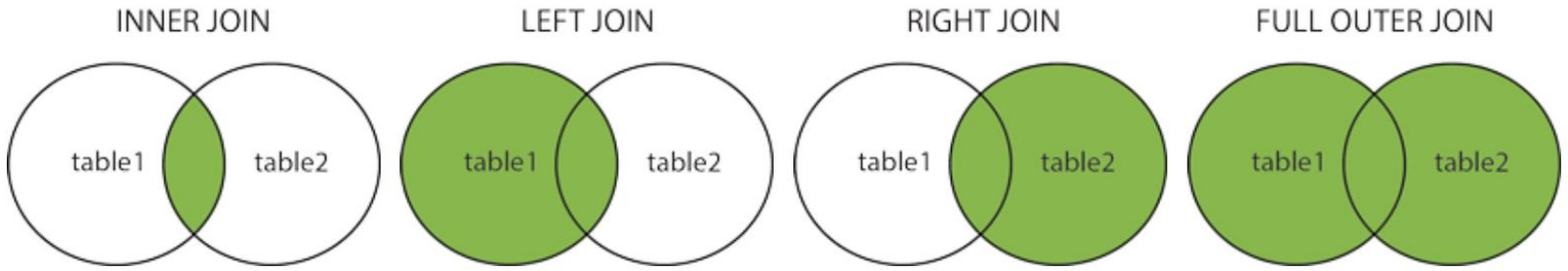
• c. Full outer join:

- Full outer join is like a left or right join except that it contains all rows from both tables.
- In full outer join, tuples in R that have no matching tuples in S and tuples in S that have no matching tuples in R in their common attribute name.
- It is denoted by ⟗.

EMP_NAME	STREET	CITY	BRANCH	SALARY
Ram	Civil line	Mumbai	Infosys	10000
Shyam	Park street	Kolkata	Wipro	20000
Hari	Nehru street	Hyderabad	TCS	50000
Ravi	M.G. Street	Delhi	NULL	NULL
Kuber	NULL	NULL	HCL	30000

Different Types of SQL JOINs: A summary

- INNER **JOIN**: Returns records that have matching values in both tables.
- **LEFT** (OUTER) **JOIN**: Returns all records from the left table, and the matched records from the right table.
- **RIGHT** (OUTER) **JOIN**: Returns all records from the right table, and the matched records from the left table.
- **FULL** (OUTER) **JOIN**: Returns all records when there is a match in either left or right table.



Student

ROLL_NO	NAME	ADDRESS	PHONE	Age
1	HARSH	DELHI	xxxxxxxxxx	18
2	PRATIK	BIHAR	xxxxxxxxxx	19
3	RIYANKA	SILIGURI	xxxxxxxxxx	20
4	DEEP	RAMNAGAR	xxxxxxxxxx	18
5	SAPTARHI	KOLKATA	xxxxxxxxxx	19
6	DHANRAJ	BARABAJAR	xxxxxxxxxx	20
7	ROHIT	BALURGHAT	xxxxxxxxxx	18
8	NIRAJ	ALIPUR	xxxxxxxxxx	19

StudentCourse

COURSE_ID	ROLL_NO
1	1
2	2
2	3
3	4
1	5
4	9
5	10
4	11

```
SELECT
StudentCourse.COURSE_ID,
Student.NAME, Student.AGE FROM
Student INNER JOIN
StudentCourse ON
Student.ROLL_NO =
StudentCourse.ROLL_NO;
```

COURSE_ID	NAME	Age
1	HARSH	18
2	PRATIK	19
2	RIYANKA	20
3	DEEP	18
1	SAPTARHI	19

```
SELECT
Student.NAME,StudentCourse.COU
RSE_ID FROM Student LEFT JOIN
StudentCourse ON
StudentCourse.ROLL_NO =
Student.ROLL_NO;
```

NAME	COURSE_ID
HARSH	1
PRATIK	2
RIYANKA	2
DEEP	3
SAPTARHI	1
DHANRAJ	*NULL*
ROHIT	*NULL*
NIRAJ	*NULL*

```
SELECT
Student.NAME,StudentCourse.COUR
SE_ID FROM Student RIGHT JOIN
StudentCourse ON
StudentCourse.ROLL_NO =
Student.ROLL_NO;
```

NAME	COURSE_ID
HARSH	1
PRATIK	2
RIYANKA	2
DEEP	3
SAPTARHI	1
NULL	4
NULL	5
NULL	4

```
SELECT
Student.NAME,StudentCourse.COURSE
_ID FROM Student FULL JOIN
StudentCourse ON
StudentCourse.ROLL_NO =
Student.ROLL_NO;
```

NAME	COURSE_ID
HARSH	1
PRATIK	2
RIYANKA	2
DEEP	3
SAPTARHI	1
DHANRAJ	NULL
ROHIT	NULL
NIRAJ	NULL
NULL	4
NULL	5
NULL	4

Thank You!!!