Parul® University

NAAC A++ ACCREDITED UNIVERSITY

PARUL INSTITUTE OF ENGINEERING & TECHNOLOGY
FACULTY OF ENGINEERING & TECHNOLOGY
PARUL UNIVERSITY

Unit 4: Relational Data Model

Dr. Vishwanath

Parul® University
NAAC A++ ACCREDITED UNIVERSITY

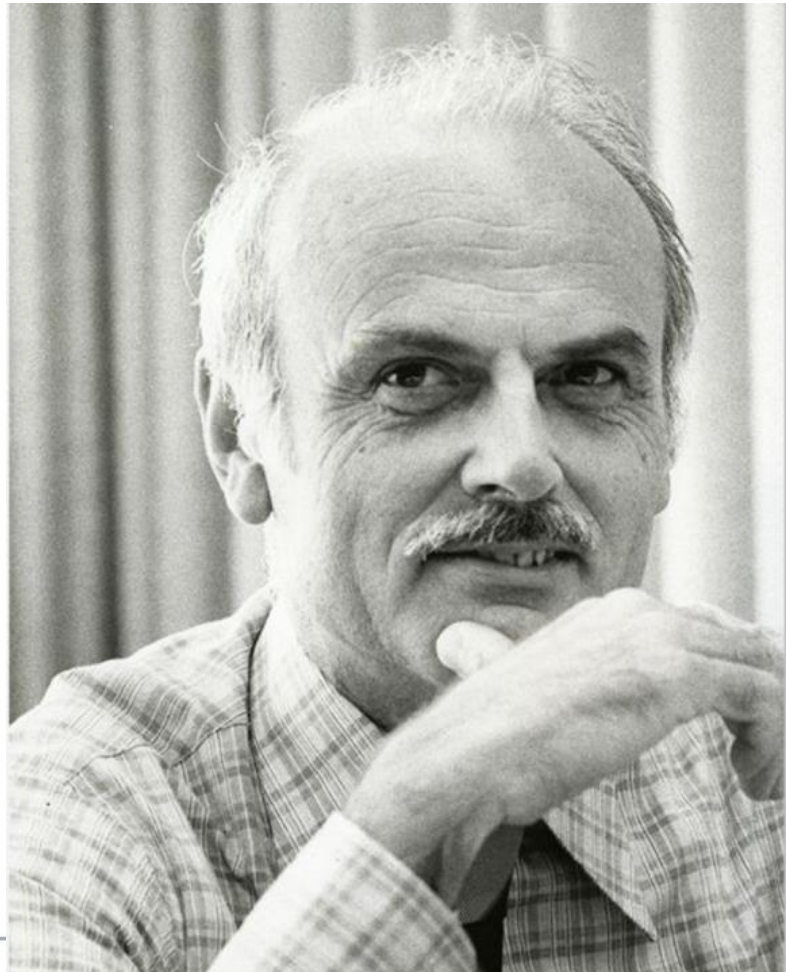
OUTLINE

- Relational Data Model
- Constraints & Keys
- Relational Algebra Operations

Relational Data Model

The relational data model describes the world as "a collection of inter-related relations (or tables)."

The relational data model was introduced by Edgar F. Codd in 1970, while working for IBM.



✓ Currently, it is the most widely used data model.

Relational Data Model

The relational data model has provided the basis for:

- Research on the theory of data/relationship/constraint
- Numerous database design methodologies
- The standard database access language called s*tructured query language (SQL)*
- Almost all modern commercial database management systems.

Relational Data Model

The relational model represents how data is stored in Relational Databases.

- Tables are also known as relations.

Each relation is a collection of columns and rows, where the column represents the attributes of an entity and the rows (or tuples) represent the records.

Student Table (Relation)

Roll Number (Primary Key)	Name	CGPA
001	Vaibhav	9.1
002	Neha	9.5
003	Harsh	8.5
004	Shreya	9.3

Tuples (Rows)

Columns (Attributes)

Relational Data Model

Attribute: Attributes are the properties that define an entity.

e.g.; ROLL_NO, NAME, ADDRESS

Tuple: Each row in the relation is known as a tuple.

ROLL_NO	NAME	ADDRESS	PHONE	AGE
1	RAM	DELHI	9455123451	18
2	RAMESH	GURGAON	9652431543	18
3	SUJIT	ROHTAK	9156253131	20
4	SURESH	DELHI		18

This relation contains 4 tuples.

Relational Data Model

Column: The column represents the set of values for a particular attribute.

Null Values: The value which is not known or unavailable. It is represented by blank space.

Degree: The number of attributes in the relation.

The STUDENT relation defined above has degree 5.

Cardinality: The number of tuples in a relation.

The STUDENT relation defined above has cardinality 4.

Relational Data Model

Advantages of the Relational Model

Simple model: It is simple and easy to use in comparison to other languages.
Flexible: It is more flexible than any other relational model present.
Secure: It is more secure than any other relational model.
Data Accuracy: Data is more accurate in the relational data model.
Data Integrity: The integrity of the data is maintained in the relational model.

Disadvantages of the Relational Model

- Relational Database Model is not very good for large databases.
- Sometimes, it becomes difficult to find the relation between tables.
- Because of the complex structure, the response time for queries is high.

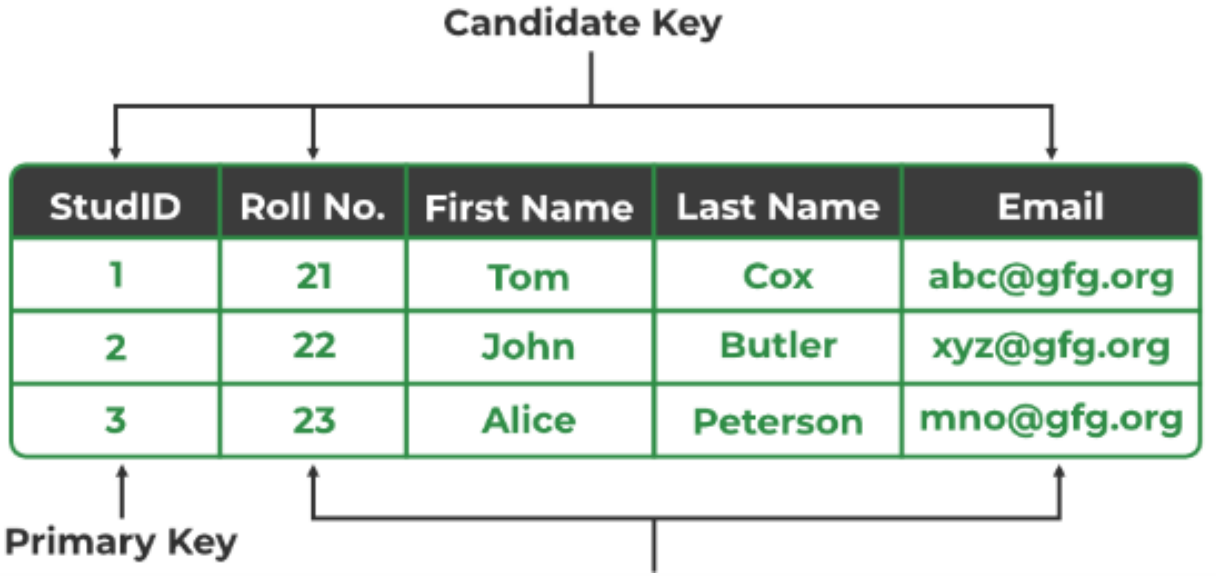
Relation Key

The keys that are used to identify the rows uniquely and also helps in identifying tables.

- Primary Key
- Foreign Key
- Super Key
- Candidate Key

Relation Key : Primary Key

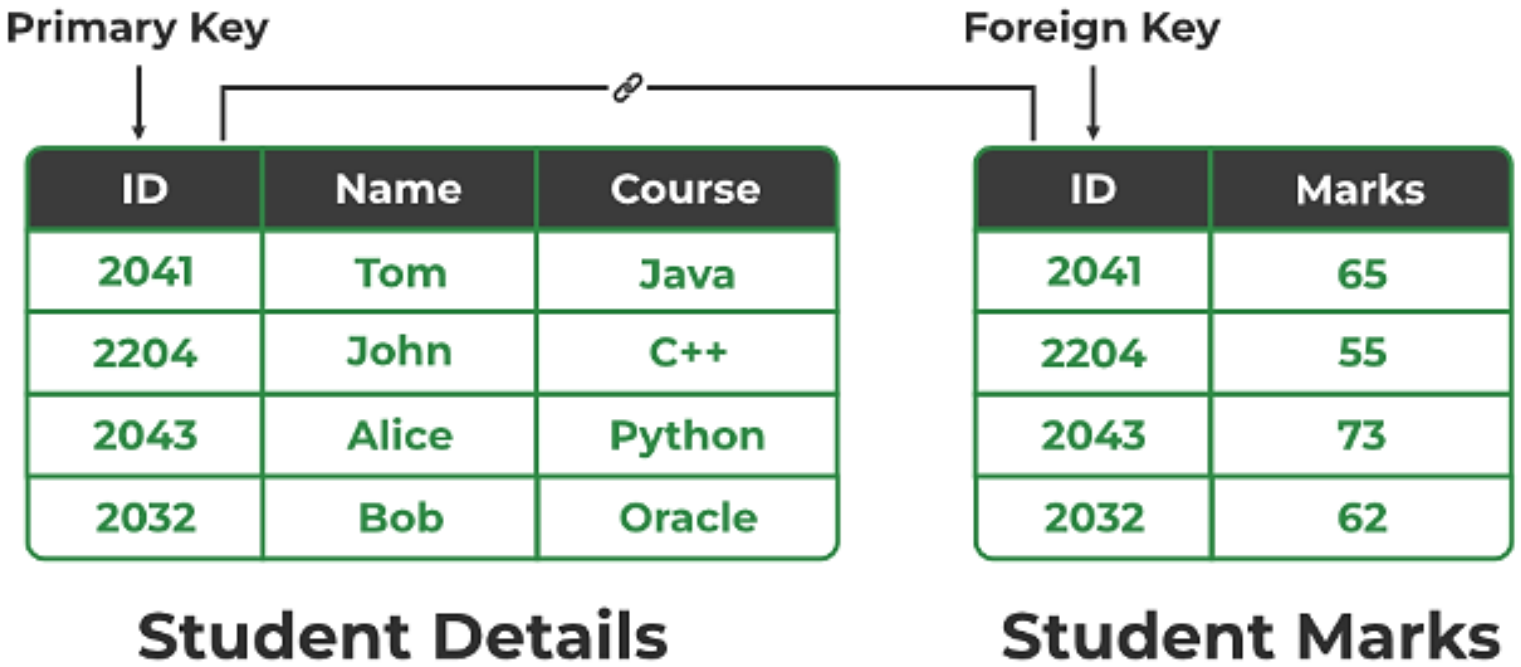
A primary key is a column or a set of columns in a table whose values uniquely identify a row in the table.



Relation Key : Foreign Key

If an attribute can only take the values which are present as values of some other attribute, it will be a foreign key.

- It acts as a primary key in one table and it acts as secondary key in another table.
- It combines two or more relations (tables) at a time.
- They act as a cross-reference between the tables.



ID	Name	Course
2041	Tom	Java
2204	John	C++
2043	Alice	Python
2032	Bob	Oracle

Student Details

ID	Marks
2041	65
2204	55
2043	73
2032	62

Student Marks

Relation Key : Super Key

The set of attributes that can uniquely identify a tuple is known as Super Key.

Example:

1. STUD_NO + STUD_NAME
2. STUD_NO + PHONE

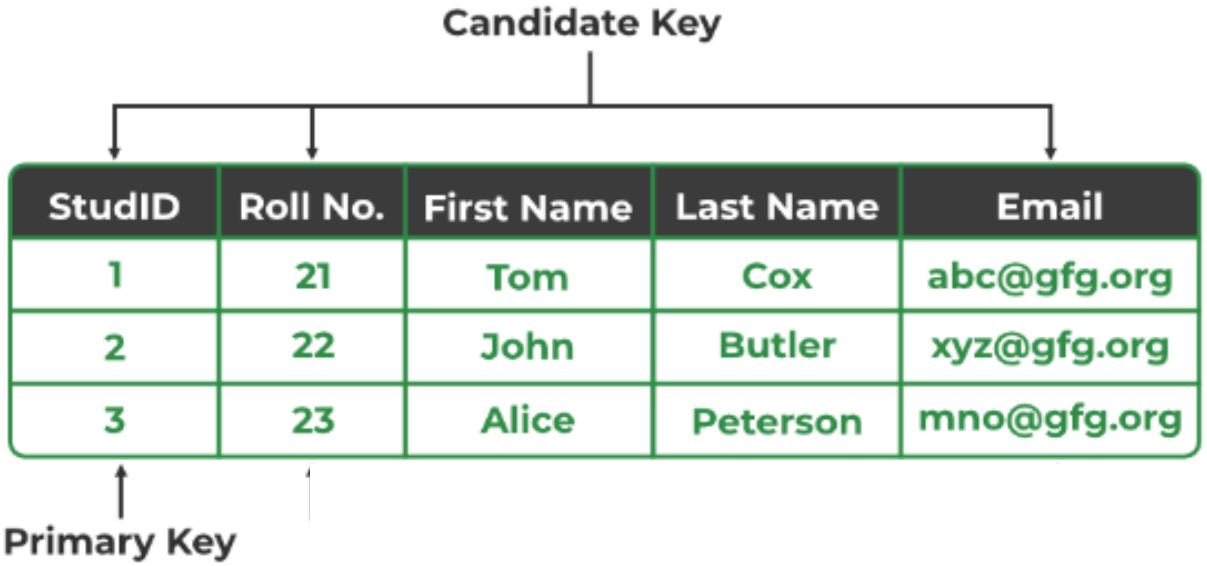
A super key is a group of single or multiple keys that identifies rows in a table.

STUD_NO	SNAME	ADDRESS	PHONE
1	Shyam	Delhi	123456789
2	Rakesh	Kolkata	223365796
3	Suraj	Delhi	175468965

Relation Key : Candidate Key

The minimal set of attributes that can uniquely identify a tuple is known as a candidate key.

Example: Stud ID, Roll No, Email



Not Null Constraint

The constraint is typically used for columns that must contain essential information.

Not Null Constraint: It ensures that a specific column in a table cannot contain null values.

Syntax: CREATE TABLE table_Name(column1 data_type(size) NOT NULL);

Example: CREATE TABLE Doctor(Doctor_ID INT *NOT NULL*,
Doctor_Name VARCHAR(100) *NOT NULL*) ;

- ✓ When you insert or update data in a column with “not null constraint”, you are required to provide a valid value, and leaving it empty is not allowed.

Check Constraint

It allows you to ensure a value meets a certain condition before it is inserted into a table.

Syntax: CREATE TABLE table_name (column1 datatype CONSTRAINT constraint_name CHECK (expression));

Example: CREATE TABLE employees (name varchar(25) CONSTRAINT age_check CHECK (age > 18));

Parul® University
NAAC A++ ACCREDITED UNIVERSITY

Relational Algebra Operations

It is used to perform different operations in relational databases.

It consists of a set of operations enabling users to manipulate and handle data stored in relational databases.

These are similar to the operations of set theory.

for example- union intersection and difference
And some personalized operations for relational databases.

Parul® University
NAAC A++ ACCREDITED UNIVERSITY

Relational Algebra Operations

Types of Relational Algebra Operations:

1. Basic

Selection (σ)
Projection (π)
Rename (ρ)
Cross Product(X)
Set operators (Union, Intersection, Set Difference)

2. Derived/advanced

Join (⋈) (Natural & Outer Join)
Aggregate Functions

Relational Algebra Operations

Here are two tables/relations which will be used for demonstrating the relational algebra operations examples:

1.STUDENT (ROLL, NAME, AGE)

ROLL	NAME	AGE
1	Aman	20
2	Atul	18
3	Baljeet	19
4	Harsh	20
5	Prateek	21
6	Prateek	23

2. EMPLOYEE (EMPLOYEE_NO, NAME, AGE)

EMPLOYEE_NO	NAME	AGE
E-1	Anant	20
E-2	Ashish	23
E-3	Baljeet	25
E-4	Harsh	20
E-5	Pranav	22

Selection (σ)

Selection Operator which is represented by "sigma"(σ).

It is used to retrieve tuples(rows) from the table where the given condition is satisfied.

Example: Suppose we want the row(s) from STUDENT Relation where "AGE" is 20

σ AGE=20 (STUDENT)

ROLL	NAME	AGE
1	Aman	20
4	Harsh	20

Projection (∏)

It pulls out some specific columns (attributes) from a relation.

It is represented by "pi"(∏).

Example: Suppose we want the names of all students from STUDENT Relation.

∏ NAME(STUDENT)

NAME
Aman
Atul
Baljeet
Harsh
Prateek

Rename (ρ)

It is denoted by "Rho"(ρ).

It is used to rename the output relation.

Example: If we want to rename the STUDENT relation as STUDENT_NAME

ρ(STUDENT_NAME,∏ NAME(STUDENT))

Cross Product(X)

It is denoted by symbol ‘X’.

It is used to perform operation on two relations.

Example: Consider two relations STUDENT(SNO, FNAME, LNAME) and DETAIL(ROLLNO, AGE) below:

S. No.	FNAME	LNAME
1	Albert	Singh
2	Nora	Fatehi

ROLLNO	AGE
5	18
9	21

Cross Product(X)

On applying CROSS PRODUCT on STUDENT and DETAIL:

STUDENT × DETAILS

SNO	FNAME	LNAME	ROLLNO	AGE
1	Albert	Singh	5	18
1	Albert	Singh	9	21
2	Nora	Fatehi	5	18
2	Nora	Fatehi	9	21

Set operators: Union

Union Operator which is represented by "union"(∪), same as the union operator from set theory.

It selects all tuples/rows from both relations/tables but with the exception that; both relations/tables must have the same set of Attributes.

Example: Suppose we want all the names from STUDENT and EMPLOYEE relation.

$$\prod \text{NAME(STUDENT)} \cup \prod \text{NAME(EMPLOYEE)}$$

NAME
Aman
Anant
Ashish
Atul
Baljeet
Harsh
Pranav
Prateek

Set operators: Intersection & Set Difference

Intersection is represented by (∩).

It selects all the tuples which are present in both relations.

Set difference gives the difference between two relations.

It is denoted by a "Hyphen"(-) and it returns all the tuples(rows) which are in relation R but not in relation S.

Set operators: Intersection

Example: If we want the names which are present in STUDENT as well as in EMPLOYEE relation.

$\prod$ NAME(STUDENT) $\cap$ $\prod$ NAME(EMPLOYEE)

NAME
Baljeet
Harsh

Set operators: Set Difference

Example: To know the names of students who are in STUDENT Relation but not in EMPLOYEE Relation.

$\prod$ NAME(STUDENT) - $\prod$ NAME(EMPLOYEE)

NAME
Aman
Atul
Prateek

Join (⋈)

A join is an operation that combines the rows/tuple of two or more tables based on common attribute/columns.

The main purpose of Join is to retrieve the data from multiple tables.

It is denoted by ⋈.

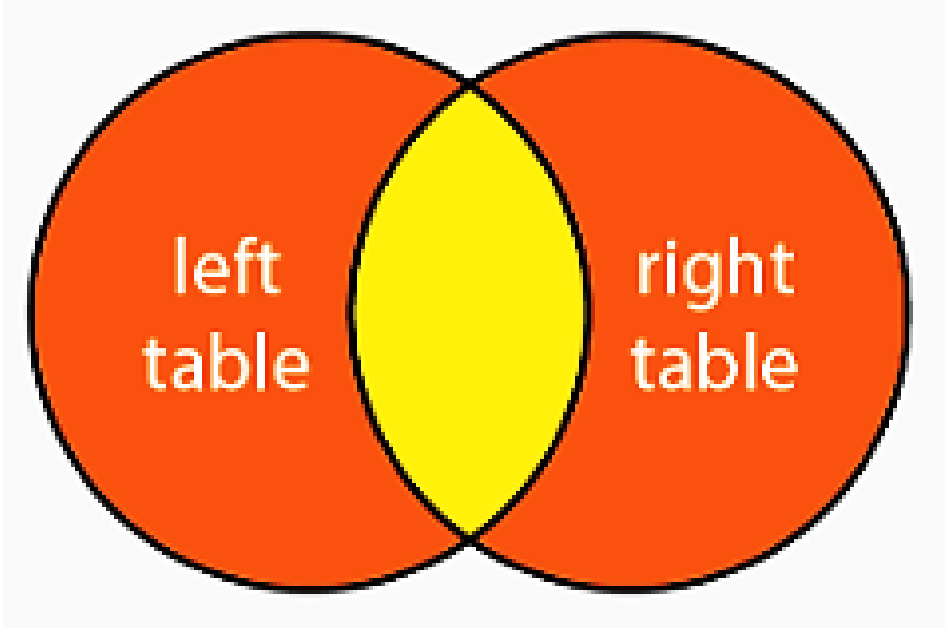
Frequently used JOIN types:

1. Inner Join
2. Outer Join

Inner Join

Inner Join is a join operation that combines two or more tables based on related columns.

It returns only rows that have matching values among tables.



- Equi Join
- Natural Join

➢ Natural Join joins two tables based on the same attribute name and datatypes.

Natural Join

Department

DEPT_NAME	MANAGER_NAME
IT	ROHAN
SALES	RAHUL
HR	TANMAY
FINANCE	ASHISH
MARKETING	SAMAY

Employee

EMP_ID	EMP_NAME	DEPT_NAME
1	SUMIT	HR
2	JOEL	IT
3	BISWA	MARKETING
4	VAIBHAV	PU
5	SAGAR	SALES

Output:

SELECT * FROM department
NATURAL JOIN employee;

DEPT_NAME	MANAGER_NAME	EMP_ID	EMP_NAME
HR	TANMAY	1	SUMIT
IT	ROHAN	2	JOEL
MARKETING	SAMAY	3	BISWA
SALES	RAHUL	5	SAGAR

Outer Join

Outer joins retrieves matching as well as non-matching records from related tables.

Types:

1. Left outer join / Left Join ⟕
2. Right outer join / Right Join ⟖
3. Full outer join / Full Join ⟗

Left Join

It retrieves all records from the left table and retrieves matching records from the right table.

Example:

Table A

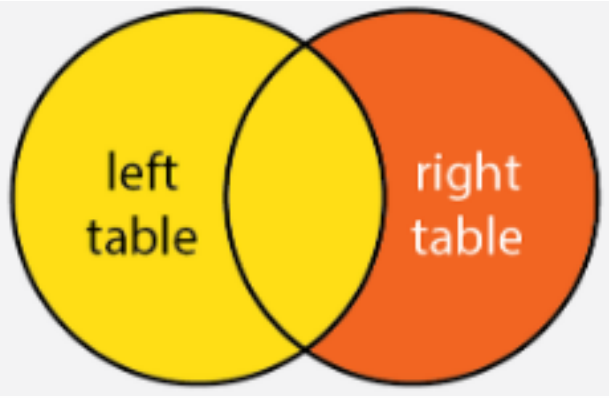
Number	Square
2	4
3	9
4	16

Table B

Number	Cube
2	8
3	27
5	125

$A \bowtie B$

Number	Square	Cube
2	4	8
3	9	27
4	16	-



Left Join

Department

DEPT_NAME	MANAGER_NAME
IT	ROHAN
SALES	RAHUL
HR	TANMAY
FINANCE	ASHISH
MARKETING	SAMAY

Employee

EMP_ID	EMP_NAME	DEPT_NAME
1	SUMIT	HR
2	JOEL	IT
3	BISWA	MARKETING
4	VAIBHAV	PU
5	SAGAR	SALES

Output:

```
SELECT * FROM department
LEFT OUTER JOIN employee ON
department.DEPT_NAME =
employee.DEPT_NAME;
```

DEPT_NAME	MANAGER_NAME	EMP_ID	EMP_NAME	DEPT_NAME
HR	TANMAY	1	SUMIT	HR
IT	ROHAN	2	JOEL	IT
MARKETING	SAMAY	3	BISWA	MARKETING
SALES	RAHUL	5	SAGAR	SALES
FINANCE	ASHISH	-	-	-

Right Join

It retrieves all records from the right table and retrieves matching records from the left table.

Example: **Table A**

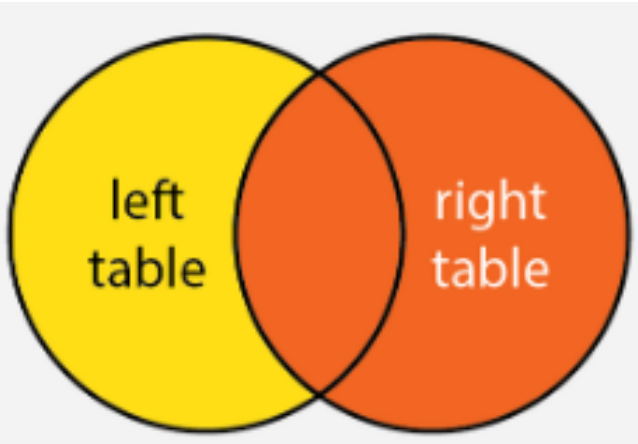
Number	Square
2	4
3	9
4	16

Table B

Number	Cube
2	8
3	27
5	125

$A \bowtie B$

Number	Square	Cube
2	4	8
3	9	27
5	-	125



Right Join

Department

DEPT_NAME	MANAGER_NAME
IT	ROHAN
SALES	RAHUL
HR	TANMAY
FINANCE	ASHISH
MARKETING	SAMAY

Employee

EMP_ID	EMP_NAME	DEPT_NAME
1	SUMIT	HR
2	JOEL	IT
3	BISWA	MARKETING
4	VAIBHAV	PU
5	SAGAR	SALES

Output:

```
SELECT * FROM department
RIGHT OUTER JOIN employee ON
department.DEPT_NAME =
employee.DEPT_NAME;
```

DEPT_NAME	MANAGER_NAME	EMP_ID	EMP_NAME	DEPT_NAME
IT	ROHAN	2	JOEL	IT
SALES	RAHUL	5	SAGAR	SALES
HR	TANMAY	1	SUMIT	HR
MARKETING	SAMAY	3	BISWA	MARKETING
-	-	4	VAIBHAV	PU

Full Join

It combines the results of both LEFT JOIN and RIGHT JOIN. The result set will contain all the rows from both tables.

Example:

Table A

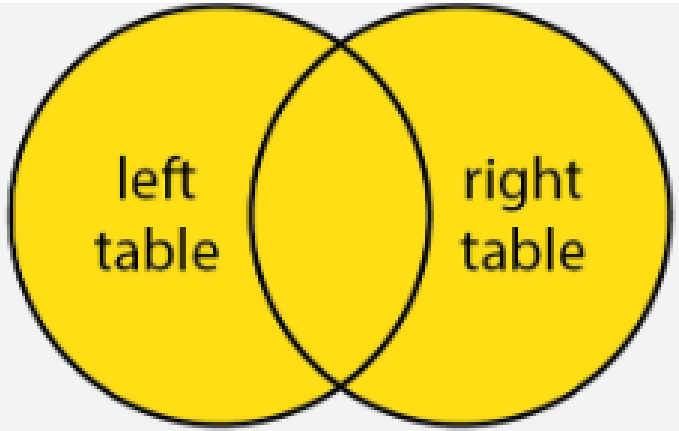
Number	Square
2	4
3	9
4	16

Table B

Number	Cube
2	8
3	27
5	125

$A \bowtie B$

Number	Square	Cube
2	4	8
3	9	27
4	16	NULL
5	NULL	125



Full Join

Department

DEPT_NAME	MANAGER_NAME
IT	ROHAN
SALES	RAHUL
HR	TANMAY
FINANCE	ASHISH
MARKETING	SAMAY

Employee

EMP_ID	EMP_NAME	DEPT_NAME
1	SUMIT	HR
2	JOEL	IT
3	BISWA	MARKETING
4	VAIBHAV	PU
5	SAGAR	SALES

Output:

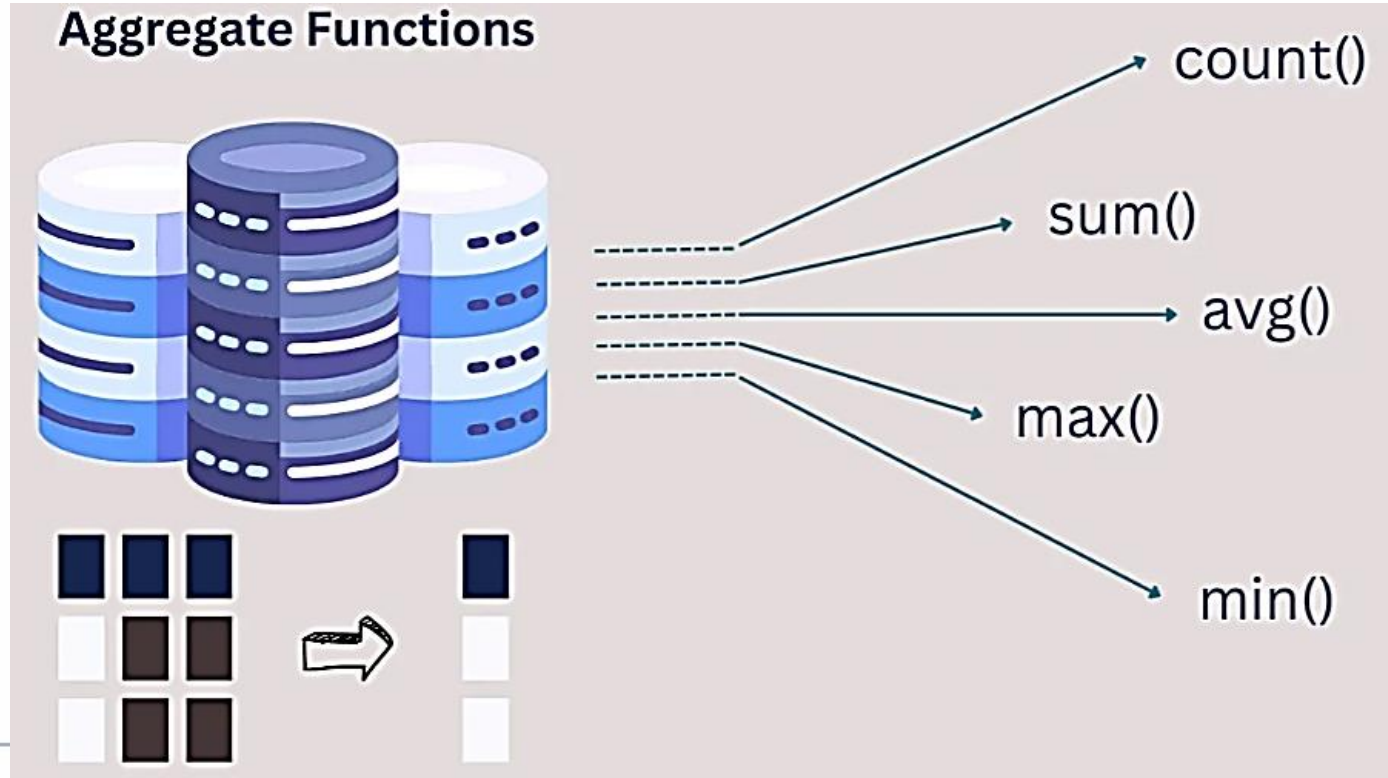
```
SELECT * FROM department
FULL OUTER JOIN employee ON
department.DEPT_NAME =
employee.DEPT_NAME;
```

DEPT_NAME	MANAGER_NAME	EMP_ID	EMP_NAME	DEPT_NAME
HR	TANMAY	1	SUMIT	HR
IT	ROHAN	2	JOEL	IT
MARKETING	SAMAY	3	BISWA	MARKETING
-	-	4	VAIBHAV	PU
SALES	RAHUL	5	SAGAR	SALES
FINANCE	ASHISH	-	-	-

Aggregate Function

Aggregation function is used to perform the calculations on multiple rows of a single column of a table.

It returns a single value.



Aggregate Function

```
CREATE TABLE Employee (Id INT, Name CHAR(1),
Salary DECIMAL(10));
INSERT INTO Employee (Id, Name, Salary)
VALUES      (1, 'A', 802),
            (2, 'B', 403),
            (3, 'C', 604),
            (4, 'D', 705),
            (5, 'E', 606),
            (6, 'F', NULL);
```

Id	Name	Salary
1	A	802
2	B	403
3	C	604
4	D	705
5	E	606
6	F	NULL

Aggregate Function

Aggregate Function Example:

```
--Count the number of employees
SELECT COUNT(*) AS Total Employees FROM Employee;
-- Calculate the total salary
SELECT SUM(Salary) AS Total Salary FROM Employee;
-- Find the average salary
SELECT AVG(Salary) AS Average Salary FROM Employee;
-- Get the highest salary
SELECT MAX(Salary) AS Highest Salary FROM Employee;
-- Determine the lowest salary
SELECT MIN(Salary) AS Lowest Salary FROM Employee;
```

Total Employees: 6

Total Salary: 3120

Average Salary: 624

Highest Salary: 802

Lowest Salary: 403

Parul® University

NAAC A++ ACCREDITED UNIVERSITY

Thank You!!!

DIGITAL LEARNING CONTENT

Parul® University

www.paruluniversity.ac.in