



PRESIDENCY COLLEGE
(Autonomous)



Reaccredited by
NAAC with A+

DATABASE MANAGEMENT SYSTEM

By

Mr N Kartik & Mr Jeelan

Assistant Professor, Dept of Computer Applications,
Presidency College, Bangalore-24

Presidency
Group

OVER
40
YEARS
OF ACADEMIC
WISDOM



PRESIDENCY COLLEGE

(AUTONOMOUS)

AFFILIATED TO BENGALURU CITY UNIVERSITY, APPROVED BY AICTE, DELHI & RECOGNISED BY THE GOVT. OF KARNATAKA

RE-ACCREDITED BY NAAC WITH 'A+' GRADE



Unit 3 Syllabus

- Relational Model Concepts
- Relational Model Constraints and Relational Database Schemas and Dealing with Constraint Violations
- Unary Relational Operations: SELECT and PROJECT
- Relational Algebra Operations from SET Theory
- Binary Relational Operations: JOIN and DIVISION
- Examples of Queries in Relational Algebra
- Relational Database Design: Anomalies in a database
- Functional dependency
- Lossless join and dependency
- Normal forms: First Normal form, Second Normal form, Third Normal form and BCNF





Presidency College
(Autonomous)



**Reaccredited by
NAAC with A+**

Unit 3 Syllabus

- SQL- SQL Data Definition and Data Types
- Specifying Constraints in SQL
- Schema Change Statements in SQL, Insert, Delete and Update Statements in SQL
- Basic Queries in SQL, More Complex SQL Queries
- Specifying Constraints as Assertion and Trigger
- Views (Virtual Tables) in SQL
- Embedded SQL, Dynamic SQL
- Introduction to PL/SQL, ODBC

Presidency
Group

OVER
40
YEARS
OF ACADEMIC
WISDOM



Relational Model Concepts

Attribute: Each column in a Table. Attributes are the properties which define a relation. e.g., Student_Rollno, NAME, etc.

Tables – In the Relational model the, relations are saved in the table format. It is stored along with its entities. A table has two properties rows and columns. Rows represent records and columns represent attributes.

Tuple – It is nothing but a single row of a table, which contains a single record.

Relation Schema: A relation schema represents the name of the relation with its attributes.

Degree: The total number of attributes which in the relation is called the degree of the relation.

Cardinality: Total number of rows present in the Table. Ex: 1 to 1, 1 to N, M to N

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




Relational Model Concepts

Entity:

An entity in DBMS (Database management System) is a real-world thing or a real-world object which is distinguishable from other objects in the real world.

For example: If we consider a car entity, it can have its attributes as a car's registration number, car's model, car's name, car's color, number of seats that are there inside the car



	Registration Number	Model	Name	Colour	Number of Seats
Entity 1 →	DL123	LDI	Ritz	White	5
Entity 2 →	DL234	VDI	Swift	Black	5
Entity 3 →	DL345	ZXI	Dzire	Red	5



Introduction

- **Relation instance** – Relation instance is a finite set of tuples in the RDBMS system. Relation instances never have duplicate tuples.
- **Relation key** - Every row has one, two or multiple attributes, which is called relation key.
- **Attribute domain** – Every attribute has some pre-defined value and scope which is known as attribute domain



Table also called Relation

Primary Key

Domain
Ex: NOT NULL

© guru99.com

CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
3	Apple	Inactive

Tuple OR Row
Total # of rows is **Cardinality**

Column OR Attributes
Total # of column is **Degree**



Relational Integrity constraints

- Relational Integrity constraints is referred to conditions which must be present for a valid relation.
- These integrity constraints are derived from the rules in the mini-world that the database represents.
- Constraints on the Relational database management system is mostly divided into three main categories are:
 - Domain constraints
 - Key constraints
 - Referential integrity constraints



Domain Constraints

- These are attribute level constraints.
- An attribute can only take values which lie inside the domain range.
- Ex: If a constraint $AGE > 0$ is applied on STUDENT relation, inserting negative value of AGE will result in failure.
- Domain constraints can be violated if an attribute value is not appearing in the corresponding domain, or it is not of the appropriate data type.



Domain Constraints

- Domain constraints specify that within each tuple, and the value of each attribute must be unique.
- This is specified as data types which include standard data types integers, real numbers, characters, Booleans, variable length strings, etc.
- **Example:**
- Create DOMAIN CustomerNameCHECK (value not NULL) The example shown demonstrates creating a domain constraint such that CustomerName is not NULL



Key Constraints

- An attribute that can uniquely identify a tuple in a relation is called the key of the table. The value of the attribute for different tuples in the relation has to be unique.
- **Example:**
- In the given table, CustomerID is a key attribute of Customer Table. It is most likely to have a single key for one customer, CustomerID =1 is only for the CustomerName =" Google".

CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
3	Apple	Inactive



Referential integrity constraints


- Referential integrity constraints is based on the concept of Foreign Keys.
- Referential integrity constraint state happens where relation refers to a key attribute of a different or same relation. However, that key element must exist in the table.

CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
3	Apple	Inactive

Customer

InvoiceNo	CustomerID	Amount
1	1	\$100
2	1	\$200
3	2	\$150

Billing




Operations in Relational Model

- Four basic update operations performed on relational database model are insert, update, delete and select
- **insert** is used to insert data into the relation
- **delete** is used to delete tuples from the table
- **modify** allows you to change the values of some attributes in existing tuples.
- **select** allows you to choose a specific range of data.
- Whenever one of these operations are applied, integrity constraints specified on the relational database schema must never be violated



Relational Algebra

- Relational Algebra is procedural query language, which takes Relation as input and generates relation as output
- Relational algebra mainly provides theoretical foundation for relational databases and SQL
- Relational algebra is a procedural query language, it means that it tells what data to be retrieved and how to be retrieved
- Relational Algebra works on the whole table at once, so we do not have to use loops etc to iterate over all the rows (tuples) of data one by one



Relational Algebra

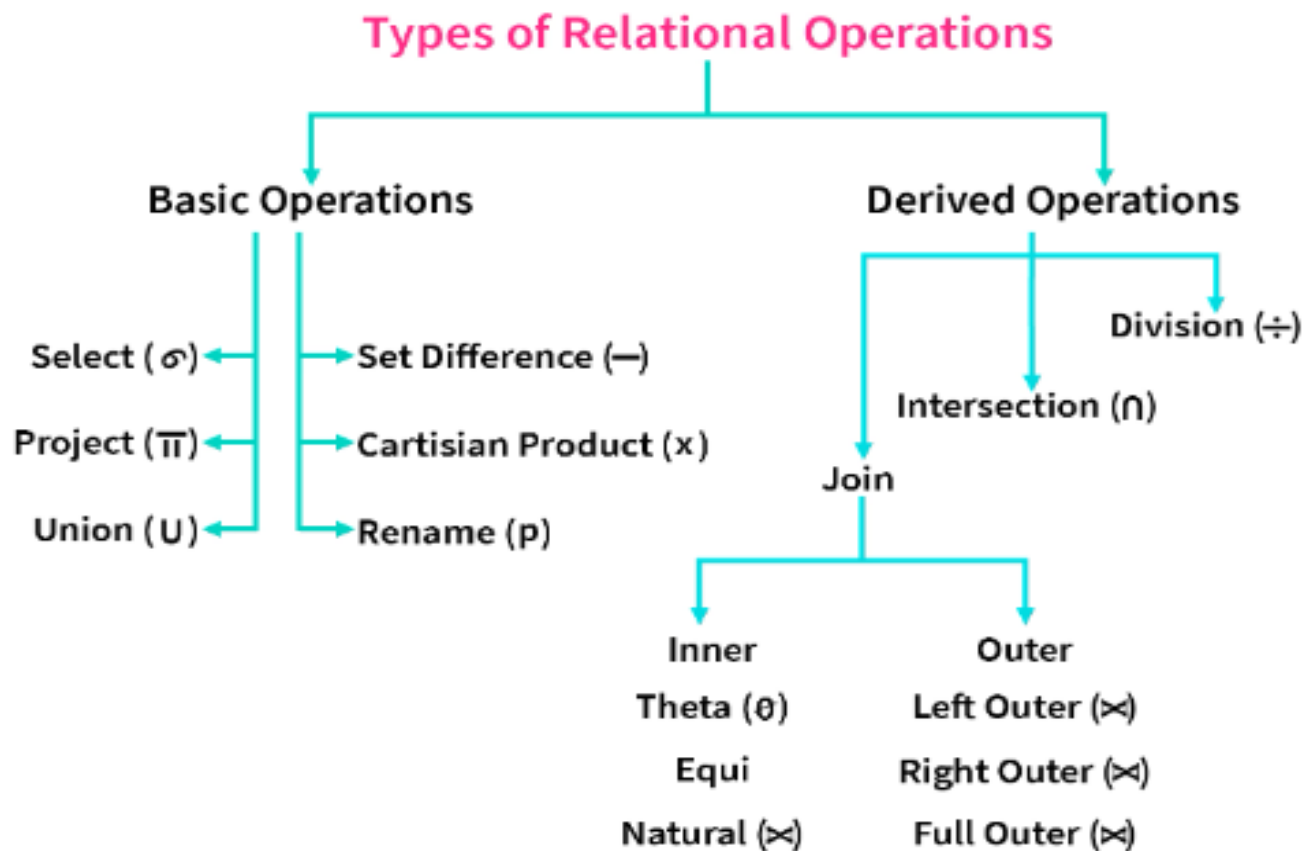
Presidency College
(Autonomous)



Reaccredited by
NAAC with A+

Presidency
Group

OVER
40
YEARS
OF ACADEMIC
WISDOM





Presidency College
(Autonomous)



*Reaccredited by
NAAC with A+*

Presidency
Group

OVER
40
YEARS
OF ACADEMIC
WISDOM

Basic and Fundamental Operations

- Select (σ)
- Project (Π)
- Union (\cup)
- Set Difference ($-$)
- Cartesian product (\times)
- Rename (ρ)



Select

- **Select Operation (σ)** : This is used to fetch rows (tuples) from table(relation) which satisfies a given condition.
- **Syntax:** $\sigma_p(r)$
- σ is the predicate
- r stands for relation which is the name of the table
- p is propositional logic
- ex: $\sigma_{\text{age} > 17}(\text{Student})$
- This will fetch the tuples(rows) from table **Student**, for which **age** will be greater than **17**.
- $\sigma_{\text{age} > 17 \text{ and gender} = \text{'Male'}}(\text{Student})$
- This will return tuples(rows) from table **Student** with information of male students, of age more than 17.



Project

- **Project Operation (Π):**
- Project operation is used to project only a certain set of attributes of a relation. In simple words, If you want to see only the **names** all the students in the **Student** table, then you can use Project Operation.
- It will only project or show the columns or attributes asked for and will also remove duplicate data from the columns.
- **Syntax of Project Operator (Π)**
- Π column_name1, column_name2, ..., column_nameN(table_name)
- Example:

Π Name, Age(Student)

Above statement will show us only the **Name** and **Age** columns for all the rows of data in **Student** table.



Union Operation(\cup)

- Union Operation (\cup):
- This operation is used to fetch data from two relations(tables) or temporary relation(result of another operation).
- For this operation to work, the relations(tables) specified should have same number of attributes(columns) and same attribute domain. Also, the duplicate tuples are automatically eliminated from the result.
- **Syntax:** $R \cup S$
- Where R is the first relation
S is the second relation

Ex:

$\Pi NAME(STUDENT) \cup \Pi NAME(EMPLOYEE)$



Set Difference (-)

- Set Difference as its name indicates is the difference of two relations (R-S). It is denoted by a "Hyphen"(-) and it returns all the tuples(rows) which are in relation R but not in relation S.
- Notation : R - S
Where R is the first relation
S is the second relation
- an example where we would like to know the names of students who are in STUDENT Relation but not in EMPLOYEE Relation.

Ex:

$\Pi \text{ NAME}(\text{STUDENT}) - \Pi \text{ NAME}(\text{EMPLOYEE})$



Cartesian product (X)

- Cartesian product is denoted by the "X" symbol. Let's say we have two relations R and S. Cartesian product will combine every tuple(row) from R with all the tuples from S.
- Notation: $R \times S$
Where R is the first relation
S is the second relation
- to combine the two relations STUDENT and EMPLOYEE.

Ex:

STUDENT X EMPLOYEE



Rename (ρ)

- Rename operation is denoted by "Rho"(ρ).
- it is used to rename the output relation. Rename operator too is a **binary operator**.
- Notation: $\rho(R,S)$ Where R is the new relation name
S is the old relation name

Suppose we are fetching the names of students from STUDENT relation. We would like to rename this relation as STUDENT_NAME.

- $\rho(\text{STUDENT_NAME}, \Pi \text{ NAME}(\text{STUDENT}))$



Binary Relational Operations

- JOIN (several variations of JOIN exist)
- DIVISION

Additional Relational Operations

- OUTER JOINS
- OUTER UNION
- AGGREGATE FUNCTIONS



Division

- The division operator is used for queries which involve the 'all'
- $R1 \div R2$ = tuples of R1 associated with all tuples of R2
- Example
- Retrieve the name of the subject that is taught in all courses
- The resulting operation must have all combinations of tuples of relation S that are present in the first relation or R



Division

Presidency College
(Autonomous)



**Reaccredited by
NAAC with A+**

Presidency
Group

OVER
40
YEARS
OF ACADEMIC
WISDOM

Name	Course
System	Btech
Database	Mtech
Database	Btech
Algebra	Btech

÷

Course
Btech
Mtech

=

Name
database



SQL Joins

- SQL Joins are used to relate information in different tables.
- A Join condition is a part of the SQL query that retrieves rows from two or more tables.
- A SQL Join condition is used in the SQL WHERE Clause of select, update, delete statements
- **The Syntax for joining two tables is:**
- `SELECT col1, col2, col3...`
`FROM table_name1, table_name2`
`WHERE table_name1.col2 =`
`table_name2.col1;`



SQL Joins

- SQL Joins can be classified into EQUI join and Non EQUI join.
- **1) SQL EQUI joins**
- It is a simple SQL join condition which uses the equal sign(=) as the comparison operator.
- Two types of EQUI joins are
 - a) SQL Outer join
 - b) SQL Inner join.
- **2) SQL Non EQUI joins (Theta Join (θ))**
- It is a sql join condition which makes use of some comparison operator other than the equal sign like $>$, $<$, $>=$, $<=$



Sample Table

product_id	product_name	supplier_name	unit_price
100	Camera	Nikon	300
101	Television	Onida	100
102	Refrigerator	Vediocon	150
103	Ipod	Apple	75
104	Mobile	Nokia	50

order_id	product_id	total_units	customer
5100	104	30	Infosys
5101	102	5	Satyam
5102	103	25	Wipro
5103	101	10	TCS



SQL Inner Join:

- All the rows returned by the sql query satisfy the sql join condition specified.
- **For example:** If you want to display the product information for each order for this, we need to identify the common column between these two tables, which is the **product_id**.
- **Ex:**
- `SELECT order_id, product_name, unit_price, supplier_name, total_units
FROM product, order_items
WHERE order_items.product_id = product.product_id;`

ORDER_ID	PRODUCT_NAME	UNIT_PRICE	SUPPLIER_NAME	TOTAL_UNITS
5103	Television	100	Onida	10
5101	Refrigerator	150	Vediocon	5
5102	Ipod	75	Apple	25
5100	Mobile	50	Nokia	30



SQL Left Outer Join

- In the SQL outer JOIN all the content of the both tables are integrated together either they are matched or not.
- **Left outer join** (also known as left join): this join returns all the rows from left table combine with the matching rows of the right table. If you get no matching in the right table, it returns NULL values.
- Ex:
- `select product_name,order_id,customer from product left join order_items on product.product_id = order_items.product_id;`

PRODUCT_NAME	ORDER_ID	CUSTOMER
Mobile	5100	Infosys
Refrigerator	5101	Satyam
Ipod	5102	Wipro
Television	5103	TCS
Camera		



SQL Right Outer Join

- **Right outer join** (also known as right join): this join returns all the rows from right table are combined with the matching rows of left table .If you get no column matching in the left table .it returns null value.
- Ex:
- select supplier_name,product_name,order_id,customer from product right join order_items on product.product_id = order_items.product_id;

SUPPLIER_NAME	PRODUCT_NAME	ORDER_ID	CUSTOMER
Onida	Television	5103	TCS
Vediocon	Refrigerator	5101	Satyam
Apple	Ipod	5102	Wipro
Nokia	Mobile	5100	Infosys



Aggregate functions

An aggregate function in SQL performs a calculation on multiple values and returns a single value.

SQL provides many aggregate functions that include avg, count, sum, min, max, etc.

An aggregate function ignores NULL values when it performs the calculation, except for the count function.

We often use aggregate functions with the GROUP BY and HAVING clauses of the SELECT statement.

Various types of SQL aggregate functions are:

Count()

Sum()

Avg()

Min()

Max()



Aggregate functions

The COUNT() function returns the number of rows in a database table.

Syntax:

COUNT(*)

or

COUNT(columnname) ;

Ex:

Select count(product_id) from Products;

SUM() Function

The SUM() function returns the total sum of a numeric column.

Syntax:

SUM(columnname) ;

Ex:

Select sum(unit_price) from Products;



Aggregate functions

AVG() Function

The AVG() function calculates the average of a set of values.

Syntax:

AVG(column name) ;

Ex:

Select AVG (quantity_in_stock) from Products;

MIN() Function

The MIN() aggregate function returns the lowest value (minimum) in a set of non-NULL values.

Syntax: MIN(column name)

Ex:

Select MIN(quantity_in_stock) from Products;

MAX() Function

The MAX() aggregate function returns the highest value (maximum) in a set of non-NULL values.

Syntax:

MAX(column name)

Ex:

Select MAX(quantity_in_stock) from Products;





Presidency College
(Autonomous)



**Reaccredited by
NAAC with A+**

Presidency
Group

OVER
40
YEARS
OF ACADEMIC
WISDOM

Relational Database Languages

- The name SQL is derived from structured query language
- SQL is a non-procedural language, means that SQL specifies what data to be retrieved from the database rather than how to retrieve the data.
- SQL is a comprehensive database language; it has statements for data definition, query and update.



SQL Statement

- The SQL statements can be grouped into following categories:
 1. Data Definition Language [DDL]
 2. Data Manipulation Language [DML]
 3. Data Control Language [DCL]
 4. Transaction Control Language [TCL]
 5. Data Query Language [DQL]



DDL

1. **Data Definition Language [DDL]:** Provides commands for defining schema, tables, indexes, sequences etc. Commands are:

CREATE: For creating schema, table, indexes, sequences.

ALTER: For altering table, indexes, sequences.

DROP: For dropping table, indexes, sequences



2. Data Manipulation Language [DML]: Provides commands for manipulating the data.

INSERT: - For inserting values or data into the table.

DELETE: - For deleting the data from the table.

UPDATE: - For updating the data in the table



3. Data Control Language [DCL]: These statements are used to give permissions to the user, take back the given permission from the user, lock certain permission from the user.

GRANT: - Giving permissions on object privileges to user.

REVOKE: - Take back the given permission from the user.

4. Transaction Control Language [TCL]: It is used to control transactions.

COMMIT: - To permanently save the changes made to transaction.

ROLLBACK: - To undo or cancel (discards) the changes up to the previous commit point.

5. Data Query Language [DQL]: It is used to retrieve data from the database

SELECT:

SQL uses the terms tables, row and column for relation, tuple and attribute respectively.

An SQL schema is identified by a schema name, and includes an authorization identifier to indicate the user or account who owns the schema, as well as descriptors for each element

DATA TYPE	DESCRIPTION
CHAR(size)	Fixed length character Size- number of characters, Max size = 200 Characters
VARCHAR(size)	Varying length characters. Size- number of characters. Max. size = 400 characters. This is also specified as VARCHAR2(Size)
NUMBER(size)	Integer number without decimal point Max. Size = 40 digits
NUMBER(size, d)	Integer numbers with decimal point or floating points numbers. Size = total number of digits and d- number of digits after decimal point.

INTEGER OR INT	Integer number. Size can't be specified.
SMALL INT	Integer number
DECIMAL (i, j)	<p>Integer numbers with decimal point or real numbers or floating point numbers.</p> <p>i- Precision is the total number of decimal digits.</p> <p>j- Scale, is the number of digits after decimal point.</p>



Date and time

DATE	For representing date, which has 10 positions and its components are year, month and day typically in the form YYYY-MM-DD.
TIME	For representing time, which has 8 positions and its components are hour, minutes and second. Typically in the form HH: MM:SS.

