Marwadi University
Marwadi Chandarana Group

NAAC A+


# Master of Computer Applications

## MCA Sem : 1

## RDBMS 05MC0105

## INTRODUCTION

1) Overview of SQL,
2) Categories of SQL Commands: DDL(Create, Alter, Truncate, Drop, Rename),
3) DML (Insert, Update, Delete),
4) DCL (Grant, Revoke),
5) TCL (Commit, Rollback, Savepoint),
6) DQL(Select),
7) Constraints (Unique, Not Null, Primary Key, Foreign Key, Check, Default),
8) SQL Operators,
9) Group By and Having,
10) Order By, Types of Joins,
11) Built-in Functions.

## Overview of Transaction Processing

- SQL stand for Structure Query Language. It is available in a number of data base management packages based on the relational model of data, for example, in DB2 of the IBM and UNIFY of the UNIFY.
- The major facilities of SQL, namely, data manipulation, data definition and data control are bound together in one integrated language framework.
- The industry-standard database language used to query and manipulate the data, structures, and permissions in a relational database.

## Overview of Transaction Processing

| Data Types | Description |
|---|---|
| **CHAR** | It is used to store fixed length character data. Maximum size is 255 and default is 1. It has right justified padding. |
| **VARCHAR2** | This data type is used to store variable length alphanumeric data. The maximum size of this data type is 2000 characters. |
| **NUMBER** | This data type is used to store numeric 0, positive, negative fixed or floating-point numbers. Numbers are stored up to 38 digits of precision (Including floating point and signs). If precision is omitted the number is maximum up to 38 digits. |
| **DATE** | This data type is us used to represent date and time. This data type stores information such as century, year, month, day, hour, minute and second. The default format is DD-MON-YY as in 12-JAN-06. |
| **RAW** | It stores binary data up to 255 bytes. |
| **LONG** | This data type is used to store variable length character strings containing up to 2GB. It is generally used to store arrays of binary data in ASCII format. It cannot be indexed and you cannot apply generally character functions such as SUBSTR. It is also known as MEMO data type. |
| **LONG RAW** | It stores binary data up to 2GB. |

## Categories of SQL Commands.

SQL commands can be roughly divide into three categories with regard to their functionality. Firstly, there are those used to create and maintain the database structure. The second category includes those commands that manipulate the data in such structures, and thirdly there are those that control the use of the database.

SQL commands fall under the following categories:

- DDL (Data Definition Language)
- DML (Data Manipulation Language)
- DCL (Data Control Language)
- TCL (Transaction Control Language)

## Categories of SQL Commands.

- **DDL(Create, Alter, Truncate, Drop, Rename),**
  - DDL is used to create and remove database objects. It is also used to alter the structure of the objects. It is also useful in dropping (removing) the objects.
  - Commands available in DDL are a follows:

1. CREATE
2. ALTER
3. TRUNCATE
4. DROP

## Categories of SQL Commands.

### A. Create Command

This command is used to create database objects such as tables, views, indexes etc. Generally it is used to create the database tables. The syntax is as follows:

**Syntax: -**

```
CREATE TABLE <TABLE_NAME>
(COLUMN_1 DATATYPE (SIZE),
COLUMN_2 DATATYPE (SIZE),
COLUMN_N DATATYPE (SIZE));
```

Categories of SQL Commands.

**Example: -**

```
CREATE TABLE EMP
(NO NUMBER (3),
NAME VARCHAR,
B_DATE DATE,
SALARY NUMBER (6,2));
```

## Categories of SQL Commands.

**Rules for Table and Column Names: -**

i. Tables and Column names can be up to 30 characters.

ii. Names must begin with an alphabet.

iii. Names cannot contain quotes.

iv. Names are NOT case sensitive.

v. Names can contain characters, a-z, 0-9, _____(Underscore), $ (Dollar Sign), and # (Hash Sign).

vi. Names cannot be reserved words.

**B. Desc or Describe Command**

This command is used to view the structure of the table.

## Categories of SQL Commands.

**Syntax:**

DESC<TABLE_NAME>;

DESCRIBE <TABLE_NAME>;

**Example:**

DESC EMP;

DESCRIBE EMP;

NOTE: - To show the list of tables, which is created by the user use the following command.

## Categories of SQL Commands.

**SELECT * FROM TAB;**

### C. Alter Command

The ALTER statement allows the user to make some kind of change to structure of some object in the database. The ALTER statement syntax can then be simplified to one of two following statements:

**Syntax: -**

ALTER TABLE <TABLE_NAME>

ADD (NEWCOLUMN_NAME DATATYPE (SIZE),

**OR**

ALTER TABLE<TABLE_NAME>
MODIFY (COLUMN_NAME DATATYPE (NEW_SIZE));

## Categories of SQL Commands.

**Example: -**

```
ALTER TABLE EMP
ADD(GENDER CHAR (1), DEPT VARCHAR2 (15));

ALTER TABLE EMP MODIFY (NO NUMBER (5));
```

**NOTE**: - If the data is already being entered in the table we cannot modify the Data Type and we cannot decrease the size of the column. We can only increase the size of the column.

Categories of SQL Commands.

- **RENAME COMMAND: -**

This command is used to change the name of the table (Rename).

Syntax: -

RENAME <OLD_TABLENAME> TO <NEW_TABLENAME>;

Example: -

RENAME EMP TO EMP_DETAILS;

## Categories of SQL Commands.

### D. Truncate Command

The TRUNCATE statement is similar to the DELETE statement. Both of the statements will delete rows from a table. The main difference is that the DELETE can be more selective (in other words, using a WHERE clause). The TRUNCATE statement simply removes all rows from a table. The TRUNCATE statement will also appear to run faster than a DELETE in most cases.

**Syntax: -**

TRUNCATE TABLE <TABLE_NAME>;

**Example: -**

TRUNCATE TABLE EMP;

NOTE: - With TRUNCATE command Rollback (Recall) is not possible but with DELETE command it is possible.

## Categories of SQL Commands.

### E. Drop Command

When a table is no longer needed, it can be dropped. Both the table definition and the rows in the table are dropped, and the space allocated for the table is made available for other database objects. The syntax for the DROP statement is about as simple as it gets:

**Syntax: -**

DROP TABLE <TABLE_NAME>;

**Example: -**

DROP TABLE EMP;

DML

❖ **DML**

- DML is used to manipulate the data in the tables & database. The commands covered under the DML are used to manipulate the existing objects.
- Commands available in DML are as follows:

**A. Insert**

This is used to add one or more row's (record's) into the table. While using this command values are separated by commas.

**Syntax: -**

INSERT INTO <TABLE_NAME> VALUES (V1, V2, …… );

**OR**

INSERT INTO <TABLE_NAME> VALUES (&COLUMN_NAME, '&COLUMN_NAME', ……) ;

DML

**Example: -**

INSERT INTO EMP VALUES (100,'Ram','16-MAR-06', 10000,'M');

INSERT INTO EMP (NO, NAME) VALUES (200,'Sita');

**OR**

INSERT INTO EMP VALUES (&NO,'&NAME','&DATE', &SALARY,'&GENDER');

## B. Update

An UPDATE statement will change one or more rows in a database table. The basic form of an UPDATE statement must specify which table to update, which column(s) to change, and optionally, whether to change all the rows in the table or just a few. The syntax is as follows:

DML

**Syntax: -**

UPDATE <TABLE_NAME> SET <COLUMN_NAME>=VALUE;

**OR**

UPDATE <TABLE_NAME> SET <COLUMN_NAME1> =VALUE1, <COLUMN_NAME2> = VALUE2
WHERE COLUMN_NAME = VALUE;

**Example: -**

UPDATE EMP SET SALARY = 8000;

NOTE: - The above command will update the Salary's of all the employees to Rs.8000.

UPDATE EMP SET NO=300 WHERE NAME= 'Ravi';

UPDATE EMP SET NO=500, B_DATE='3-June-04' WHERE NAME='Ram';

DML

NOTE: - The above command will update only those rows, which fall under the condition, and not all the rows of the table.

**C. Delete**

As the name implies, the DELETE statement will remove rows from a database table. You can delete all rows or use a WHERE clause to specify rows, similar to the UPDATE statement. Here's the syntax:

Syntax: -

DELETE FROM <TABLE_NAME>;

DELETE FROM <TABLE_NAME> WHERE COLUMN_NAME=VALUE;

Example: -

DELETE FROM EMP;

DELETE FROM EMP WHERE NO=100;

## DCL

DCL is used to control the data access to the database. DCL concerns with the Recovery and Security issue of database. This category of command allows us to grant and revoke privileges and permissions.

Commands available in DCL are as follows:

### A. Grant

The GRANT statement is almost self-explanatory. GRANT will give a privilege (either object or system) to a user, a role, or to all users. The basic syntax for granting privileges is as follows:

Syntax: -

GRANT sys_privilege [, sys_privilege...]

TO user | role | PUBLIC [, user | role | PUBLIC...];

GRANT obj_privilege [(column list)] ON object

TO user | role | PUBLIC [WITH GRANT OPTION];

DCL

Example: -

GRANT CONNECT TO BCA; GRANT RESORUCE TO BCA; GRANT DBA TO MCA;

GRANT SELECT ON EMP TO MYROLE; GRANT MYROLE TO MCA;

**B. Revoke**

As you would expect, the REVOKE statement is the opposite of the GRANT statement. Either system privileges or object privileges can be revoked with the following basic syntax:

## DCL

Syntax: -

REVOKE obj_privilege | ALL [, obj_privilege] ON object

FROM user | role | PUBLIC [, user | role | PUBLIC...];

REVOKE sys_privilege | ALL [, sys_privilege...] FROM user | role | PUBLIC [, user | role | PUBLIC...];

Example: -
REVOKE CONNECT FROM BCA;
REVOKE REOSOURCE FROM BCA;
REVOKE DBA FROM MCA;

## TCL

Transaction is a sequence of SQL statements that Oracle treats as a single block of unit. To save a set of changes made to a table using DML commands and to get back to the original state before changes have been made we use TCL.

Commands available in DCL are as follows:

1. Commit

A set of changes made to a table using DML (INSERT, UPDATE, DELETE) commands are temporary. To make the changes permanent, COMMIT command must be needed after DML commands.

## TCL

Syntax: -

COMMIT;

Example: -

DELETE FROM EMP WHERE NO=500;

COMMIT;

2. Rollback

The ROLLBACK statement allows you to change your mind about a transaction. It brings back the original state of the tables to the state as of the last COMMIT statement or the beginning of the current transaction.

## TCL

Syntax: -

ROLLBACK [TO SAVEPOINT <Savepoint_Name>];

Example: -

DELETE FROM EMP WHERE NO=500; ROLLBACK;

**NOTE**: - You cannot ROLLBACK the transaction after the COMMIT. So, After the COMMIT you cannot use ROLLBCAK command.

3. Savepoint

Additionally, you can use SAVEPOINT to further subdivide the DML statements within a transaction before the final COMMIT of all DML statements within the transaction. SAVEPOINT essentially allows partial rollbacks within a transaction.

## TCL

Syntax: -

SAVEPOINT <Savepoint_Name>;

ROLLBACK TO SAVEPOINT <Savepoint_Name>;

Example: -

DELETE FROM EMP WHERE NO=500; SAVEPOINT A1;

ROLLBACK TO SAVEPOINT A1;

## DQL (Select)

In its most basic form, the SELECT statement has a list of columns to select from a table, using the SELECT ... FROM syntax. The * means "all columns." The most basic SELECT syntax can be described as follows:

Syntax: -

SELECT [DISTINCT] COLUMN_1,COLUMN_2 , …..,COLUMN_N FROM <TABLE_NAME> [WHERE <Condition>] [Order by <Column> [Asc / Desc}];

NOTE:- It displays the information contained in the column, which is selected in the query only.

OR

SELECT * FROM <TABLE_NAME>;

NOTE: - It displays the information contained in all the fields of the table.

**Example: -**

SELECT NO, NAME, GENDER FROM EMP; SELECT * FROM EMP;

SELECT * FROM EMP WHERE NAME = ‘Ram’; SELECT * FROM EMP WHERE SALARY=8000;

## DQL (Select)

**Other Examples: -**

SELECT NAME, NO FROM EMP; SELECT DISTINCT NO FROM EMP;

SELECT DISTINCT NAME FROM EMP;

SELECT DISTINCT NO, NAME FROM EMP;

SELECT * FROM EMP WHERE NO>200;

SELECT * FROM EMP WHERE NO=200;

SELECT * FROM EMP ORDER BY NAME;

SELECT DISTINCT NAME FROM EMP ORDER BY NAME;

SELECT DISTINCT NO, NAME FROM EMP ORDER BY NO;

SELECT DISTINCT NO, NAME FROM EMP ORDER BY NAME;

SELECT * FROM EMP ORDER BY NAME DESC;

# Database Design



## DQL (Select)

```
SELECT * FROM EMP ORDER BY NO, NAME;

SELECT * FROM EMP ORDER BY NO, NAME DESC;

SELECT * FROM EMP ORDER BY NO DESC, NAME;

SELECT * FROM EMP ORDER BY NO DESC, NAME DESC;
```

## Constraints

**INTIGRITY DATA CONSTRAINTS: -**

- Some of the restriction has to be applying on the database to ensure the validity and integrity of database.
- Rule, which are enforced on the data to be entered and prevents the user from entering invalid data into the table. This mechanism is handled by the constraints.
- ORACLE checks the data using entered into the table columns against the data constraints. If it passes the check than data is entered into the table else data is rejected.
- If a single column fails against a constraint entire record is rejected and not stored into the table.
- There are mainly two types of data constraints are there.

## Constraints

**INTIGRITY DATA CONSTRAINTS: -**

- Some of the restriction has to be applying on the database to ensure the validity and integrity of database.
- Rule, which are enforced on the data to be entered and prevents the user from entering invalid data into the table. This mechanism is handled by the constraints.
- ORACLE checks the data using entered into the table columns against the data constraints. If it passes the check than data is entered into the table else data is rejected.
- If a single column fails against a constraint entire record is rejected and not stored into the table.
- There are mainly two types of data constraints are there.

## Constraints

**CATEGORIES OF CONSTRAINTS: -**

1) DOMAIN CONSTRAINT (Associate with Column)
    a. NOT NULL
    b. CHECK
    c. DEFAULT
2) ENTITY CONSTRAINT (Associate with ROW)
    a. PRIMARY KEY
    b. UNIQUE
3) REFERENTIAL INTEGRITY CONSTRAINT (Link between two table)
    a. FOREIGN KEY

## Constraints

**INPUT – OUTPUT CONSTRAINTS:**

It determines the speed at which the data can be inserted or accessed from oracle table. It is dividing into two categories.

**Primary Key Constraints: -**

It is attached to one column or more than one column of the table. The constraints that apply to the columns are unique and not null.

PRIMARY KEY = UNIQUE + NOT NULL.

A single column primary key is called Simple Primary Key. A multi column primary key is called Composite Primary Key.

Only when a record can’t be uniquely identify the value in a single column than composite primary key is required.

## Constraints

**Primary key constraints at column level: -**

Syntax: -

COLUMN-NAME DATA-TYPE (SIZE) PRIMARY KEY

Example:

CREATE TABLE EMP (E_NO NUMBER (3) PRIMARY KEY, E_NAME VARCHAR2 (15));

**Primary key constraints at table level: -**

Syntax: -

PRIMARY KEY (COL1, COL2, ….., COLN);

## Constraints

Example:

```
CREATE TABLE EMP
(E_NO NUMBER (3),
E_NAME VARCHAR2 (15),
PRIMARY KEY (E_NO));
```

**To ADD constraints when the table is already there: -**

Syntax: -

```
ALTER TABLE <TABLE-NAME> ADD PRIMARY KEY (COL-NAME)
```

Example: -

```
ALTER TABLE EMP ADD PRIMARY KEY (E_NO);
```

## Constraints

**To DROP a constraints: -**

Syntax: -

ALTER TABLE <TABLE-NAME> DROP <CONSTRAINT-NAME>;

Example:

ALTER TABLE EMP DROP PRIMARY KEY;

**To give the constraints name on the Primary key: -**

Syntax: -

COLUMN-NAME DATA-TYPE (SIZE) CONSTRAINT CONS NAME PRIMARY KEY

# Database Design

## Constraints

Example: -

```
CREATE TABLE EMP

(E_NO NUMBER (3) CONSTRAINT P_KEY PRIMARY KEY,

E_NAME VARCHAR2 (15));
```

## Constraints

**Foreign Key Constraints: -**

It creates the relationship between the master table and detail table and it ensures.

Records can't be inserted into the detail table, if corresponding record in the master table doesn't exist.

Record of the master table can't be deleted, if corresponding record in the detail table exist.

A Foreign key is a column or group of columns whose values are derive from the primary key or unique key of some other table.

The table in which the foreign key is defined is called foreign key table or Detail table.

## Constraints

The table which define primary key is called Primary key table or Master table.

It creates the relationship between the Master table and Detail table.

It ensures:

- Records can’t be inserted into the detail table, if corresponding record in the master table doesn’t exist.
- Record of the master table can’t be deleted, if corresponding record in the detail table exist.

## Constraints

**Foreign key constraints at column level: -**

Syntax: -

COLUMN-NAME DATA-TYPE (SIZE) REFERANCES <TABLE-NAME>

[COL-NAME] [ON DELETE CASECADE]

Example: -

CREATE TABLE EMP_DETAIL (ENO NUMBER (3) REFERENCES ENAME VARCHAR2 (15));

## Constraints

**Foreign key constraints at table level: -**

Syntax: -

FOREIGN KEY (COL1, COL2, ….., COLN) REFERENCES <TABLE-NAME> [COL-NAME] ;

Example: -

CREATE TABLE EMP_DETAIL (ENO NUMBER (3), ENAME VARCHAR2 (15)

FOREIGN KEY (ENO) REFERANCES EMP (E_NO));

Constraints

**To ADD constraints when the table is already there.**

Syntax: -

ALTER TABLE EMP ADD PRIMARY KEY (E_NO);

Example: -

ALTER TABLE EMP_DETAIL CONSTRIANT FORAIGN KEY REFERANCES EMP (E_NO);

**NOT NULL: -**

This constraint is applied only at column level and not for table level.

This constraint can be applied to single column or more than one column.

Because of that the NULL value is restricted to be entered into the column.

## Constraints

**Principles of NULL value: -**

1. Setting a NULL value is appropriate when the actual value is unknown.
2. A NULL value is not equivalent to “zero” for the “number” data-type or “space” for “Character” data-type.
3. A NULL value will evaluated to null in any expression. Example: - NULL * 10 = NULL
4. NULL value can be inserted into the columns of any data-type.
5. If a column has NULL values the oracle ignores the ‘UNIQUE’, ‘FOREIGN KEY’, and ‘CHECK’ constraints.

Syntax: -

COLUMN-NAME DATA-TYPE (SIZE) NOT NULL

Example: -

CREATE TABLE EMP (E_NO NUMBER (3) NOT NULL, E_NAME VARCHAR2 (15));

## Constraints

**To ADD constraints when the table is already there: -**

Example: -
ALTER TABLE EMP ADD CONSTRIANT EMPNO (E_NO) NOT NULL;

**UNIQUE: -**

The purpose of UNIQUE constraint is to ensure the information in the column must be UNIQUE.

One table can contain many UNIQUE constraints.

The main aim of the UNIQUE constraint is to uniquely identify each row from the table.

## Constraints

**Unique constraints at column level: -**

Syntax: -

COLUMN-NAME DATA-TYPE (SIZE) [CONSTRAINT <CONSTRAINT-NAME>] UNIQUE;

Example: -

```
CREATE TABLE EMP

(E_NO NUMBER (3) CONSTRAINT E_UNQ UNIQUE,

E_NAME VARCHAR2 (15));
```

## Constraints

**Unique constraints at table level: -**

Syntax: -

UNIQUE (COL-NAME-1, COL-NAME-2)

Example: -

```
CREATE TABLE EMP
(E_NO NUMBER (3),
E-NAME VARCHAR2 (15),
UNIQUE (E_NO, E_NAME));
```

## Constraints

**Unique constraints at table level: -**

Syntax: -

UNIQUE (COL-NAME-1, COL-NAME-2)

Example: -

```
CREATE TABLE EMP
(E_NO NUMBER (3),
E-NAME VARCHAR2 (15),
UNIQUE (E_NO, E_NAME));
```

**To ADD a constraint when the table is already there: -**

Example: -

```
ALTER TABLE EMP ADD CONSTRIANT E_UNQ (E_NO) UNIQUE;
```

## Constraints

**BUSINESS CONSTRAINTS: -**

For the integrity of data some of the rules must be implemented that is known as Business rules or Business constraints. **Example:** your employee no must start with a letter ‘E’.

Constraints can be connected to one column or more than one column of a table.

A business rule constraint can be implemented through CHECK keyword.

The CHECK constraint evaluate the condition and gives true or false result.

It takes longer time to execute compare to NOT NULL, PRIMARY KEY, FOREIGN KEY and UNIQUE constraint.

## Constraints

**CHECK: -**

**Check constraints at column level: -**

Syntax: -

COLUMN-NAME DATA-TYPE (SIZE) CHECK (LOGICAL EXPRESSION)

Example: -

```
CREATE TABLE EMP

(E_NO NUMBER (3) CHECK (E_NO LIKE 'E%'),

E_NAME VARCHAR2 (15));
```

## Constraints

**Check constraints at table level: -**

Syntax: -

CHECK (LOGICAL EXPRESSION)

Example: -

```
CREATE TABLE EMP
(E_NO NUMBER (3),
E-NAME VARCHAR2 (15),
CHECK (E_NO LIKE 'E%'));
```

## Constraints

**DEFAULT: -**

When we are creating any table at that time we can assign a default value to a column.

When user is inserting value in the columns and by mistake leaves the column empty to which we have assign “Default” keyword, then the oracle engine will automatically insert the default value in that column.

Syntax: -COLUMN-NAME DATA-TYPE (SIZE) DEFAULT (VALUE);

Example: -

```
CREATE TABLE EMP

(E_NO NUMBER (3) DEFAULT 100 PRIMARY KEY,

E_NAME VARCHAR2 (15));
```

## Group By and Having

The GROUP BY clause is another section of the select statement. This optional clause tells Oracle to group rows based on distinct values that exist for specified columns. The GROUP BY clause creates a data set, containing several sets of records grouped together based on a condition.

| P_NO | QTY_ORDER | QTY_DISP |
|---|---|---|
| P1 | 10 | 10 |
| P4 | 3 | 3 |
| P6 | 7 | 7 |
| P2 | 4 | 4 |
| P5 | 10 | 10 |
| P3 | 2 | 2 |
| P1 | 6 | 6 |
| P6 | 4 | 4 |
| P4 | 1 | 1 |
| P6 | 8 | 8 |

Group By and Having

SELECT P-NO, SUM (QTY_ORDERD) “TOTAL QTY” FROM SALES_DETAIL GROUP BY P_NO;

**OUTPUT: -**

| P_NO | TOTAL QTY |
|---|---|
| P1 | 16 |
| P2 | 4 |
| P3 | 2 |
| P4 | 4 |
| P5 | 10 |
| P6 | 19 |

Group By and Having

**HAVING CLAUSE:**

The HAVING clause can be used in combination with the GROUP BY clause. HAVING imposes a condition on the GROUP BY clause, which further filters the groups created by GROUP BY clause.

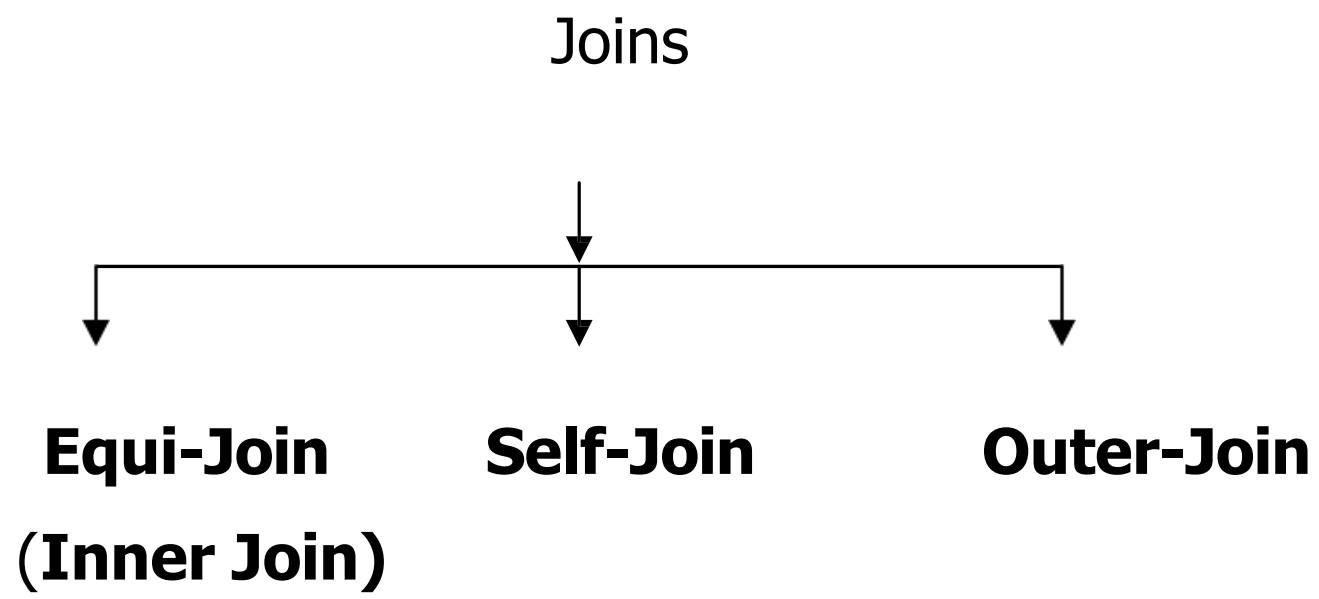
SELECT P-NO, SUM (QTY_ORDERD) “TOTAL QTY” FROM SALES_DETAIL GROUP BY P_NO HAVING P_NO=’P1’ OR P_NO=’P4’;

**OUTPUT: -**

| P_NO | TOTAL QTY |
|---|---|
| P1 | 16 |
| P4 | 4 |

## Types of Joins

- Join means to access rows from two or more tables. A Join operation involves two table and table must be JOIN with a WHERE clause in which the common key field must be specify.
- The following are different types of joins.



## Types of Joins

**Equi-Join (Inner Join, Joining Multiple Table): -**

When matching record are displayed from both the table is called as “Equi – Join”.

**Example: -**

**SALES_ORDER**

| O_NO | CLIENT_NO | ORDER_DATE |
|---|---|---|
| O001 | C006 | 12-APR-04 |
| O002 | C002 | 25-DEC-04 |
| O003 | C001 | 03-OCT-04 |
| O004 | C005 | 18-JUN-04 |
| O005 | C002 | 20-AUG-04 |
| O006 | C001 | 12-JAN-04 |

Types of Joins

SELECT O_NO, NAME, ORDER_DATE “ORDER-DATE” FROM SALES_ORDER, CLIENT_MASTER

WHERE CLIENT_MASTER.CLIENT_NO = SALES_ORDER.CLIENT_NO ORDER BY ORDER_DATE;

**CLIENT_MASTER**

| CLIENT_NO | NAME | BAL_DUE |
|---|---|---|
| C001 | ASHOK | 500 |
| C002 | VISHAL | 1000 |
| C003 | AJAY | 0 |
| C004 | ROHIT | 0 |
| C005 | NALINI | 0 |
| C006 | PARESH | 0 |
| C007 | RAHUL | 0 |

## Types of Joins

### Self-Join (Joining a table to itself): -

In some situation you may find it is necessary to join a table to itself. As if you are joining two separate tables. In sort we are opening a single table with a different name and join them as they are different tables present in distinct memory location.

**Example:**

EMP_INFO

| EMP_NO | NAME | MNGR_NO |
|---|---|---|
| E001 | RAVI | E002 |
| E002 | ARPIT | E005 |
| E003 | RAHUL | E004 |
| E004 | LAXMAN | - |
| E005 | SUMIT | - |

## Types of Joins

```
SELECT EMP.NAME, MNGR.NAME "MANAGER" FROM EMP_INFO EMP, EMP_INFO MNGR
WHERE EMP.EMP_NO = MNGR.MNGR_NO;
```

**Output: -**

| NAME | MANAGER |
|---|---|
| ARPIT | RAVI |
| LAXMAN | RAHUL |
| SUMIT | ARPIT |

## Types of Joins

### Outer Join (+): -

In previous two join operations, if the matching records are not present in the second table, certain records from the first table will be not displayed.

To retrieve this record also we have to perform the outer join operation using the outer join operator (+). The data, which is not available in the second table, will be presented as the NULL values.

**Example:**

**PLAYER**

| ROLL_NO | NAME |
|---|---|
| 10 | VIJAY |
| 20 | PEAS |
| 30 | ANAND |
| 40 | MAHESH |

**MATCH**

| MATCH_NO | ROLL_NO | MATCH-DATE | OPPONENT |
|---|---|---|---|
| 1 | 20 | 10-JUL-96 | WASINGTON |
| 2 | 30 | 12-JAN-96 | SAHPRAS |
| 3 | 20 | 12-AUG-96 | VIJAY |
| 4 | 30 | 20-MAR-96 | BORG |

Types of Joins

```
SELECT PLAYER.ROLL_NO, NAME, MATCH_DATE, OPPONENT

FROM PLAYER, MATCH WHERE PLAYER.ROLL_NO = MATCH.ROLL_NO (+);
```

- **Output: -**

| ROLL_NO | NAME | MATCH_DATE | OPPONENT |
|---|---|---|---|
| 10 | VIJAY | - | - |
| 20 | PAES | 10-JUL-96 | WASHINGTON |
| 20 | PAES | 12-AUG-96 | VIJAY |
| 30 | ANAND | 12-JAN-96 | SAHPRAS |
| 30 | ANAND | 20-MAR-96 | BORG |
| 40 | MAHESH | - | - |

## Built In Functions

An oracle function serves the purpose of manipulating the data items and returning a function. Function are also capable of accepting user supplied variables or constants and operates on them. Such variables and constants are called as arguments. Any number of arguments can be passing to a function in the following format.

FUNCTION-NAME (Arg-1, Arg-2, ……., Arg-N)

Oracle function can be group to gather depending upon whether they operate a single row or group of rows retrieve from a table.

## Built In Functions

Group Function / Aggregate Function: -

Function that act on a set of values are called as Group Function.

For Example: -

SUM is a function, which calculates the total set of values provided as a argument in a function. A Group Function returns a single result row for a group of query rows.

Categories of Group Function: -

## Built In Functions

- **AVG Function: -**

Syntax: - AVG ([Distinct / All] N)

Purpose: - It returns the average value of N and it ignores the NULL value.

Example: - SELECT AVG (SALARY) “Avg - Salary” FROM EMP;

Output: - Avg - Salary

……
4142.85714

## Built In Functions

- **MIN Function: -**

Syntax: - MIN ([Distinct / All] Expression)

Purpose: - It returns the minimum value of Expression.

Example: - SELECT MIN (SALARY) “Minimum - Salary” FROM EMP;

Output: - Minimum - Salary

……

2500

## Built In Functions

- **MAX Function: -**

Syntax: - MAX ([Distinct / All] Expression)

Purpose: - It returns the maximum value of Expression.

Example: - SELECT MAX (SALARY) “Maximum - Salary” FROM EMP;

Output: - Maximum - Salary

……

6500

## Built In Functions

- **SUM Function: -**

Syntax: - SUM ([Distinct / All] N)

Purpose: - It returns the total of value N.

Example: - SELECT SUM (SALARY) “Total - Salary” FROM EMP;

Output: - Total - Salary

……

29000

## Built In Functions

- **COUNT Function: -**

Syntax: - COUNT (* [Distinct / All] Expression)

Purpose: - It counts the number of expression in a given column.

The * option returns number of rows including duplicate values and NULL values. It gives number of row count where expression is not Null.

Example: - SELECT COUNT (*) “Total No of Rows” FROM EMP;

Output: - Total No Of Rows

……

## Built In Functions

Example: - SELECT COUNT (SALARY) “Total No Of Salary” FROM EMP;

Output: - Total No Of Salary

……

7

Built In Functions

**Example of all Group Function: -**

SELECT AVG (SALARY), MIN (SALARY), MAX (SALARY), SUM (SALARY),
)UNT (SALARY) FROM EMP;

Output: -

| AVG (SALARY) | MIN (SALARY) | MAX (SALARY) | SUM (SALARY) | COUNT (SALARY) |
|---|---|---|---|---|
| 4142.85714 | 2500 | 6500 | 29000 | 7 |

# THANK YOU