1-10-2024

----------------------------ORACLE---------------------------

Topic 1 DBMS

Topic 2 : ORACLE

Topic 3 : SQL

- – Introduction To SQL
- – Sub- Language Of SQL
- – Data-Types In Oracle SQL
- – Operators In Oracle SQL
- – Functions In Oracle SQL
- – Clauses In Oracle SQL
- – Joins
- – Constraints
- – Sub-Quires
- – Views
- – Sequence
- – Index--------------(Interview Level)

Topic 4 : NORMALIZATION

- – What Is Normalization
- – Where We Want To Use Normalization
- – Why We Need Normalization
- – Types Of Normalization
  - First Normal Form
  - Second Normal Form
  - Third Normal Form
  - Bcnf Normal Form
  - Fourth Normal Form
  - Fifth Normal Form

Topic 5 : PL/SQL

- – Introduction To Pl/SQL
- – Difference Between SQL And Pl/SQL
- – Conditional &Looping Statements
- – Cursors
- – Exceptions And Handling
- – Stored Procedures
- – Stored Functions
- – Triggers

About full-stack Java developer:

In it field a user is interacting the following 2 types of applications 1) front-end applications

2) back-end applications

Front-end Applications:

Front-end application is an applications where the end-users are interacting to an applicationsdirectly

ex:Registration form,login form,view profile ,homepage ...etc.

design& develop:

– UI technologies (html,css and JavaScript,angular js,react,etc...

Back-end application:

– back-end application is an application where the end-users data information is storedex:Database

design & develop

– DB technologies (oracle,SQLserver,mySQL,postgreSQL,db2...etc

server-side Technologies:

– this technologies are use to establish a connection in between front-end applicationandback-endapplication

ex:Java,Python,.Net,Python...etc.

2-10-2024

Topic-1 DBMS

❖ **What is data?**

• It is a raw-fact(i.e character,number,special characters and symbols) • Data is never give meaningful statements to users

1001 is data	Smith is data
1002 is data	Allen is data
1003 is data	Miller is data

❖ **What is information?**

• Processing data is called as “Information”;

Ex:

Customer Id	**Customer name**
1001	Smith is data
1002	Allen is data

1003	Miller is data

❖ What is database?

- It is memory which is used to store the collection of inter-related data or informationof particular business organization.

  Ex.

  SBI_BANK_DB

1) Group of branches
    i) Group of departments
        a) Group of employees

What is inter-related data information ?

- Depending on each other is called as inter-related.

Ex.

No employee = no departments

no departments = no employee

Ex.

No products = no customers

no customers = no products

❖ Types of databases?

There are two types of databases in real world .

1.OLTP(online transaction processing)

2.OLAP(online analysis processing)

OLTP:

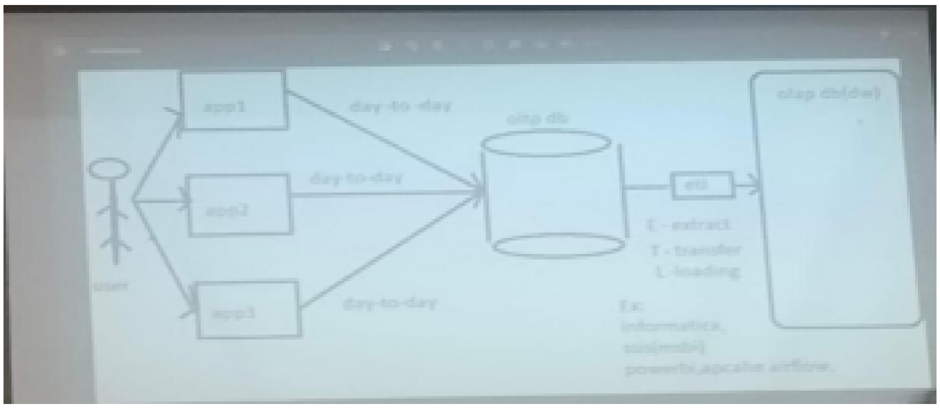
- These databases for storing “day to day” transactional information

  Ex.oracle,SQLserver,mySQL,postgreSQL,db2.....etc.

OLAP:

These databases are used for storing “historical data”(i.e. Big-data).

Ex. Data warehouse

Fig: 

- **What is DBMS?**

  It is a software which is used manage and maintain data or information with in the database. By using DBMS software we will following operations are,

  Creating databases

  Creating tables

  Inserting data

  Updating data

  Selecting data

  Deleting data

  Here DBMS s/w will act as an interface between user and database.

  USER DBMS s/w DATABASES

- **Models of DBMS?**

There are three types of DBMS models

1. Hierarchical databases management system (**HDBMS**)

   Ex.IMS(information management system)

2.network database management system(**NDBMS**)

   Ex.IDBMS(integrated database management system.

Note:

HDBMS,NDBMS model are outdated in real time.

3.relational database management(**RDBMS**)

Relational database management(RDBMS):

There are two modules in RDBMS.

1. Object relational DBMS(ORDBMS)
2. Object oriented DBMS(OODBMS)

3-10-2024

❖ **Object relational DBMS:**

• These databases are storing data in the form of **table** format .

A table = collection of rows & columns

A row = group of columns

• A row can be called as record/ tuple
• A column can be called as attribute/field.
• Object relational databases are depends on "SQL" . So that these databases are calledas"SQLdatabases".

Ex.Oracle,SQLserver,MySQL,PostgreSQL,DBs...etc

❖ **Object oriented DBMS:**
• These databases are storing data in the form of "Object".
• These databases are depends on **OOPs** concept but not SQL so these are called as **NoSQL**databases

Ex. MongoDB,Cassandra,...etc.

Topic-2 :ORACLE

Introduction to Oracle:

• Oracle is an RDBMS product which was introduced by **Oracle Corporation** in 1979. • Oracle is used to store data permanently(i.e Hard Disk) along with security manner. • When we want to deploy Oracle software in the system then we need a platform.

What is platform?

• It is combination of operating system and micro-processor.

There are two types of platforms.

1. **Platform dependent:** it support only one operating systemwith the combinationofany micro-processor. Ex.cobal,pascal,c
2. **Platform Independent:** it support any operating system with the combinationof anymicro-processor. Ex.Oracle,java,.net,Python...etc.

• Oracle software can be installed in any operating system such as windows,Linux,Mac,Salaries..etc• Oracle is a platform independent and RDBMS product.

Versions Of Oracle:

The first version s/w is “oracle 1.0”.

- Oracle 1.0
- Oracle 2.0
- Oracle 3.0
- Oracle 4.0
- Oracle 5.0
- Oracle 6.0
- Oracle 7.0
- Oracle 8.0
- Oracle 8i(internet)
- Oracle 9i
- Oracle 10g(grid technology)
- Oracle 11g
- Oracle 12c(cloud technology)
- Oracle 18c
- Oracle19c(Latest version)
- Oracle 21c(very latest version)
- Oracle 23c(Beta version).

Working with Oracle:

- When we are installing internally there are two components installed in the systemautomatically.**1.** Client component
  2. Server component

Client component:

- By using client tool we will perform the following the three operations 1. User can connect to

  oracle server.

  Enter Username : system(default username)

  Enter Password : tiger(created at oracle s/w installation) connected. 2. User

  send request to oracle server:

  Request:SQL query /SQL command

  3.user will get response from oracle server:

  Response: result / output

Ex.SQLplus, SQL Developer,Toad are client tools of oracle.

4-10-24

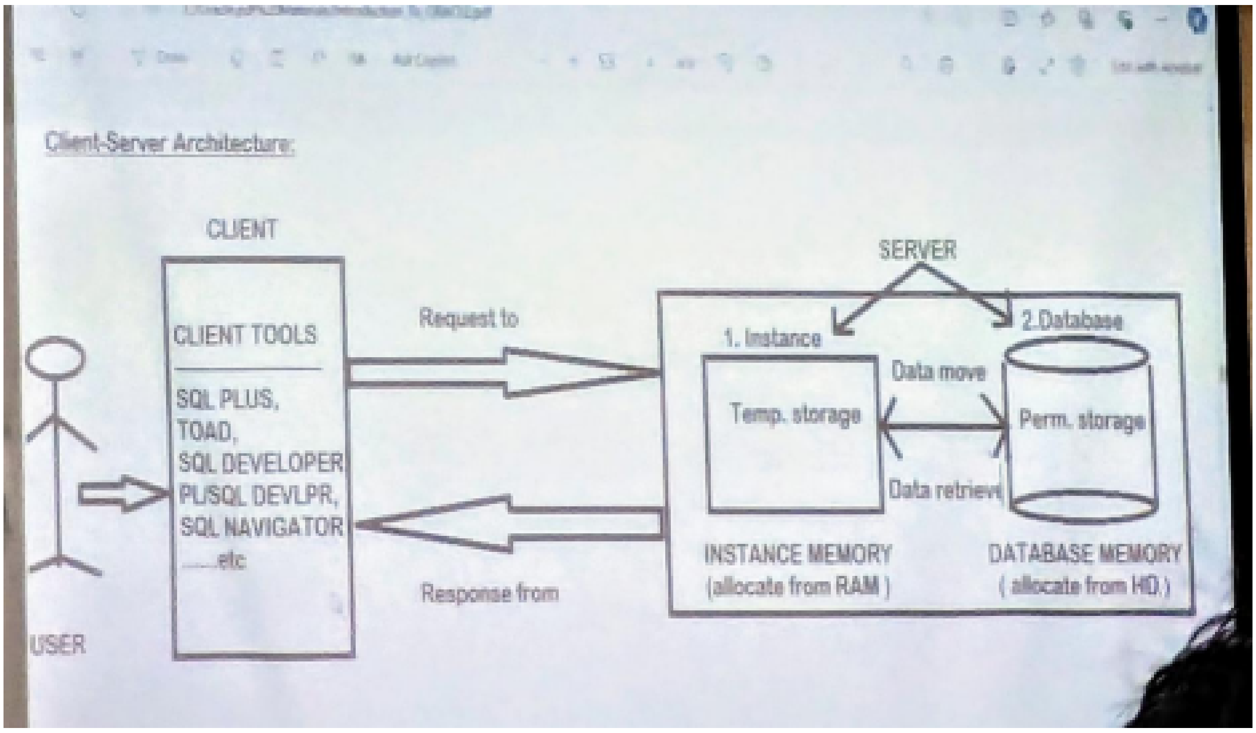
2.**Server component**:

Server is having two more sub-components internally

i. **Instance:** it is a temporary memory which will allocate from ram. Here data can be store

temporarily.

ii. **Database:** It is a permanent memory which is allocate in hard disk. Here data can be store permanently.



Types of oracle s/w editions:

There are two editions in oracle s/w

- **Oracle Express edition:**

  Supporting partial features of oracle databases.

  Ex.recycle Bin,flashback,purge,partition table...etc not allowed

- **Oracle enterprise edition:**

  Supporting all features of oracle database.

  Ex.Recycle bin,flashback,purge,partition table etc..all are allowed

5-10-24

Note: When want to work on database then we follow the following two steps procedure.

i. **Connect**: when user want to connect to SQL server then user required a databaseclient tool is“SQLplus”
ii. **Communicate**:When we want to communicate with database then we need a databaselanguageis“SQL”

SQLPlus VS SQL

SQLPLUS	SQL
1.It is a database client tool it is introduced**Oracle Corporation**.	1.It is a database language so wh by introduced by **IBM**.
2.It is used to connect to oracle server	is used for communicate with datab

3.It will act as an editor for writing & executingSQL PL/SQL programs.	3.It is again five sub languagesare(DDL,DML,DQL/DRL,TCL,DCL) usedtoperform some operations over database.

How to connect oracle server?

- Go to all programs
- After go to oracle-oradbhome-1 folder
- Click on SQLPLUS icon

Enter Username:system(default user)

Enter Password: tiger (created at installation)

Connected.

Note:Here username is not a case-sensitive **but** password is a case-sensitive. **How to**

create a username and password in oracle server?

Syntax:

Create user <username> identify by <password>;

Ex:

Go to open SQLPLUS

Enter the username:system/tiger;

Connected;

SQL>CREATE USER MYDB9AM IDENTIFIED BY 123; User created.

7-10-2024

SQL> CREATE USER MYDB9AM identified by 123

User created.

Enter user-name:MYDB9AM/123

ERROR:

ORA-0145 : User MYDB9AM lacks CREATE SESSION privilege; login denied.

Granting “DBA” permission to the user(MYDB9AM):

Syntax:

GRANT <PRIVILEGE NAME> TO <USERNAME>

Ex:

Enter the user-name:SYSTEM/TIGER

Connected;

SQL>GRANT DBA TO MYDB9AM;

Grant succeeded.

SQL> CONN

Enter user-name:MYDB9AM/123

Connected.

- **How to change password for user(MYDB9AM):**

Syntax:

SQL>PASSWORD;

Changing password for MYDB9AM

Old password:123

New password:ABC

Retype New password:ABC

Password changed

SQL> CONN

Enter user-name:MYDB9AM/ABC

Connected.

- **How to re-create a new password for user,if we forget it:** <u>Syntax:</u>

ALTER USER <USERNAME> IDENTIFIED BY <NEW PASSWROD> SQL>ALTER USER

MYDB9AM IDENTIFIED BY MYDB9AM User altered.

SQL>CONN

Enter the user-name :MYDB9AM/MYDB9AM Connected.

- **How to re-create a new password for SYSTE admin if we forget if: SQL>**CONN

Enter user-name: \SYS AS **SYSDBA(default user-name) (main boss of DB)** Enter

password:**SYS (default password)**

Connected.

SQL>ALTER USER SYSTEM IDENTIFIED BY LION User altered.

SQL>CONN

Enter user-name:SYSTEM/LION;

Connected.

- **How to view user-name in oracle if we forgot if:**

<u>Syntax:</u>

SELECT USERNAME FROM ALL USERS;(All users in a pre-defined table).

Ex:

SQL>SELECT USERNAME FROM ALL_USERS;

- **How to drop a user from oracle:**

Syntax:

DROP USER <USERNAME> CASCADE ;(cascade is a pre-defined keyword) Ex:

SQL>CONN

Enter user-name:SYSTEM/LION

Connected.

SQL>DROP USER MYDB9AM CASCADE;

User dropped.

To clear the screen:

Syntax:

CL SCR;

To exit from oracle(i.e disconnect):

syntax:

EXIT;

Introduction to SQL:

- SQL stands for “structure query language” introduced by IBM • SQL is used to communicated with DATABASE.
- The initial name is “**SEQUEL”** and the later named as “**SQL**”. • SQL is not a case-sensitive language i.e we will write SQL queries either upper/lower/combinationof lower & upper case characters.

Ex.**SQL>** SELECT * FROM EMP**; ————————executed.**

SQL>SELECT * FROM EMP**;—————————-executed.**

SQL>SELECT * FROM EMP;——————————**-executed.**

- Every SQL queries ends with “ ; ” semicolon.

Sub-language of SQL:

There are five sub-language of SQL

1) Data Definition Language(DDL):

- CREATE
- ALTER

  ALTER – MODIFY

  ALTER-ADD

  ALTER-RENAME

  ALTER-DROP

- RENAME
- TRUNCATE
- DROP

New features /new commands:

- Recycle bin
- FLASHBACK
- PURGE

2)Data Manipulation Language(DML): (WRITE ONLY CMDS)

- INSERT
- UPDATE
- DELETE

3) DATA QUERY/ RETRIEVAL LANGUAGE(DQL/DRL): (READ ONLY) • SELECT

4) TRANSACTION CONTROL LANGUAGE (TCL):

- COMMIT
- ROLLBACK
- SAVEPOINT

5) DATA CONTROL LANGUAGE(DCL):

- GRANT
- REVOKE

1) Data Definition Language(DDL):

CREATE:

- To create a new database object such as

  Table,View,Sequence,Procedure,Functions,Triggers,...etc.**HOW TO CREATE A NEW TABLE IN**

  ORACLE DATABASE:

Syntax:

```
CREATE TABLE <TABLE NAME> (<COLUMN
NAME1><DATATYPE>[SIZE],<COLUMNNAME2><DATATYPE>[SIZE]...........);
```

8/10/2024

Data types in oracle:

- It is an attribute which will specify "what type of data" is storing into a column in the tuple. • Oracle supports the following data types are
  1. Number data types
  2. character/String data types
  3. Long data types
  4. Date data types
  5. Raw & long raw data types
  6. Lob data types

Number data types:

- storing integers & float values.
- It contains two arguments those are precision and scale.

Numbers(P,S):

- **Numbers**(**P**):store integer format values only.
- **Numbers(P,S):**store float values only.

Precision:

- Counting all digits in the given expression
- The maximum size is 38 digits

  Ex. 1 10 precision 2

2. 86745 precision 5

Ex. S.No number(1)

0

1

2

.

9

10(not possible to store)

Scale(s):

· Counting the right side digits of a decimal point from the given expression. · There is no maximum size of scale because it is a part of precision value.

Ex.56.23

Precision -4;

Scale - 2

Ex 2:
7584.20

Precision - 6.

Scale - 2.

Ex: price number(4,2)

0.0

56.23

95.23

99.99

100 (it will store 100.00 so error)

Character/String data type:

· Storing string format data type only
· In database string can be represent with '<string>'

Ex: EmpName char(10)

Smith —- Not allowed

'smith' —-allowed

1021 ——not allowed

'1021'——allowed

56.12—-not allowed

‘56.12’——-allowed

String format data

Characters only st	Alphanumeric string
[A-Z],[a-z]	[A-Z],[a-z],(0-9),(@#$%^&*..)
Ex.’SMITH’,’smith’...etc	Ex.’smith123@gmail.com’,PASSWORD

Types of character / String data types:

There are 2 types of data types.

1. **Non-Unicode data types:**
   - Supporting to store “**localized data** (i.e English language only). 1. char(size)
     2. varchar2(size)
2. **Unicode data types:**
   - Supporting to store “**Globalized data**” (i.e ALL National languages). 1. Nchar(size)
     2. Nvarchar2(size)

Here N stands for National language. [A-Z],[a-z]

9-10-2024

Char(size):

- It is fixed length datatype(i.e static datatype).
- It will store non-Unicode character in the form of 1 char = 1 byte • The maximum size is 2000 bytes.

Disadvantages:

Memory wasted.

Varchar2(size):

- it is variable length datatype (i.e dynamic datatype)
- it will store non-Unicode characters In the form of 1 char = 1 byte • the maximum size is 4000bytes.

Advantage:

Memory saved.

Nchar:

- It is fixed length datatype(i.e static datatype).
- It will store Unicode character in the form of 1 char = 1 byte • The maximum size is 2000 bytes.

Disadvantages:

Memory wasted.

NVarchar2(size):

- it is variable length datatype (i.e dynamic datatype)
- it will store Unicode characters In the form of 1 char = 1 byte • the maximum size is 4000bytes.

Advantage:

Memory saved

Long datatype:

it is a variable length datatype (i.e dynamic datatype)

it will store non-Unicode & Unicode character in the form of 1 char = 1 byte• a table ia having

only one long datatype column

the maximum size is 2gb
DATE DATATYPES:

to store date & information of a particular day /transaction

the range from '01-JAN-4712 BC' to '31-DEC-9999 AD'.

1. DATE

2. TIMESTAMP

DATE:

- storing date & information but time is optional
- if user not insert time information then oracle server will take '00:00:00 AM' by default • the default time format is oracle database is 'DD-MON-YY/YYYY HH:MI:SS'.

Ex:

'DD-MON-YY/YYYY HH:MI:SS'

'09-OCT-24/2024 10:27:XX'

1 1 2 1 1 1 ----------- 7 BYTES FIXED MEMORY . **TIMESTAMP:**

- Storing date & information including milliseconds.
- The default timestamp format in oracle database is 'DD-MON-YY/YYYY HH:MI:SS:MS**EX:**

'DD-MON-YY/YYYY HH:MI:SS:MS'

'09-OCT-24/2024 10:27:XX:-4gb'

1 1 2 1 1 1 4 ----------- 11 BYTES FIXED MEMORY. **Raw & Long Raw Datatypes:**

Storing image /audio/video file in the form of 010010101 binary format.

RAW :

static datatype: 2000bytes

LONGRAW :

dynamic datatype :2gb

LOB data types:

Lob stands for large objects.

1. **CLOB**
2. **NCLOB**
3. **BLOB**

CLOB:

- It stands for “character large object: datatype.
- It is dynamic datatype.
- It will store non-Unicode character in the form of 1char = 1 byte • The maximum size is 4gb.

NCLOB:

- It stands for “national character large object: datatype.
- It is dynamic datatype.
- It will store Unicode character in the form of 1char = 1 byte • The maximum size is 4gb.

BLOB:

- It stands for “binary large object” datatype.
- It is a dynamic datatype.
- It will sore image / audio/video file in the form of 011010101010 binary format • The maximum size is 4 GB.

Non-Unicode character:

Char(size)-2000

Varchar2(size) -4000

Long-2gb

CLob-4gb

Unicode character:

Nchar(size) -2000

NVarchar2(size)-4000

Long-2gb

NCLOb-4gb

Binary format data:

Raw-2000

LongRaw-2gb

Blob-4gb

17-10-2024

How to create a new table in oracle:

SYNTAX:

```
    CREATE TABEL <TABLE NAME>(<COLUMN NAME1>
    <DATATYPE>[SIZE],<COLUMNNAME2> <DATATYPE>[SIZE],........);
```

EX:

SQL>CONN

Connected

SQL>CREATE TABLE STUDENT (STID NUMBER(4),SNAME CHAR(10),SFEE NUMBER(6,2)); **To view the structure of a table:**

Syntax:

```
    DESC <table name>;
```

Ex:

SQL>DESC STUDENT;

To view the the list of tables in oracle database:

Syntax:

```
    SELECT * FROM TAB;
```

EX:

SQL>SELECT * FROM TAB;

ALTER command:

– To change or modified the structure of a table.
– It again four sub-commands those are,

- **ALTER –MODIFY:**

– To change datatype and also the size of datatype of a specific column in the table.

Syntax:

```
    ALTER TABLE <TABLENAME> MODIFY <COLUMN NAME> <new datatype>[newsize];
```

Ex:

SQL>ALTER TABLE STUDENT MODIFY SNAME VARCHAR2(20);

- **ALTER – ADD:**

To add new column to an existing table.

Syntax:

ALTER TABLE <TABLE NAME> ADD < NEW COLUMN NAME><DATATYPE>[SIZE];

EX:

SQL>ALTER TABLE STUDENT ADD SADDRESS LONG;

- **ALTER – RENAME**

To change a column name in the table.

Syntax:

ALTER TABLE <TABLE NAME> RENAME COLUMN <OLD COLUMN NAME> TO<NEWCOLUMN NAME>;

EX:

SQL>ALTER TABLE STUDENT RENAME COLUMN SADDRESS TO SADD;

- **ALTER – DROP:**

TO DELETE A COLUMN FROM A TABLE PERMENENTLY.

SYNTAX:

ALTER TABLE <TABLE NAME> DROP COLUMN <COLUMN NAME>; Ex:

SQL>ALTER TABLE STUDENT DROP COLUMN SFEE;

18-10-2024

RENAME CMD:

To change the table name.

Syntax:

RENAME <OLD TEBLE NAME> TO <NEW TABLE NAME>

SQL> RENAME STUDENT TO SDETAILS;

SQL>RENAME SDETAILS TO STUDENT;

TRUNCATE CMD:

Deleting all rows but not columns from a table.

We cannot delete a specific row because truncate cmd is NOT allowed “WHERE” clausecondition.

Syntax:

TRUNCATE TABLE <TABLE NAME>;

Ex:

SQL>TRUNCATE TABLE STUDENT WHERE STID=1022;

————————-NOTALLOWED**SQL>**TRUNCATE TABLE STUDENT**;**

—————-ALLOWED

DROP CMD:

To delete a table (i.e. collection of rows and columns) from database. Syntax:

DROP TABLE <TABLE NAME>

Ex:

SQL> DROP TABLE STUDENT;

Note: before oracle 10g version enterprise edition once we drop a table it is permanentlydroppedwhereas oracle 10 enterprise edition once we drop a table it is temperarly dropped

Oracle 10g enterprise edition features:

- **Recycle Bin:**

It is a predefined table in oracle

It will store the information about deleted table from database. It is similar to windows

recycle bin in computer.

Syntax:

SELECT OBJECT_NAME,ORIGINAL_NAME FROM RECYCLEBIN;

OBJECT_NAME	
BIN$ncrymuKR LSKLJFklskl	

- **FLASHBACK:**

It Is A DDL cmd which is used to restore a deleted table from recycle bin to database.

Syntax:

FLASHBACK TABLE <TABLE NAME> TO BEFORE DROP; EX:

SQL>FLASHBACK TABLE STUDENT TO BEFORE DROP;

- **PURGE:**

This statement is used to delete a table from database permanently. Syntax:

DROP TABLE <TABLE NAME> PURGE;

Ex:

SQL> DROP TABLE STUDENT PURGE;

NOTE: The above features are working under USER(MYDB9AM) account but not inDBA(SYSTEM) account.

Data Manipulation Language (DML):

- INSERT CMD:

It insert a new row data into a table.

1. **CASE-1: Inserting all columns values:**

Syntax:

INSERT INTO <TABLE NAME> VALUES (VALUES 1,VALUES2)...; EX:

SQL>INSERT INTO STUDENT VLAUES (1021,'ALEN',2500);

19-10-2024

2. **CASE-2: inserting specific columns values:**

Syntax:

INSERT INTO <TABLE NAME> (required columns names)

VALUES(values1,values2,...);EX:

SQL>INSERT INTO STUDENT (STID) VALUES(1021);

SQL>INSERT INTO STUDENT(SNAME,SFEE)VALUES('SMITH',2500);

SQL>INSERT INTO STUDENT(STID,SNAME,SFEE)VALUES(1023,'SMITH',4500);

SQL>INSERT INTO STUDENT(SNAME,SFEE,STID)VALUES('JONES',3900,1024);

How to insert values into a table dynamically:(multiple rows):

When we insert values into a table dynamically then we use a special operator is "&" Syntax:

INSERT INTO <TABLE NAME> VALUES(<&columns name1><&column name2>...); Ex:

SQL>INSERT INTO STUDENT VALUES(&SID,'&SNAME',&SFEE);

Enter value for sid:1025

Enter value for sname:adams

Enter value for sfee:5400

SQL> /(To re0execute the lastly execute SQL query in SQLplus editor) Enter value

for sid:1026

Enter value for sname:james

Enter value for sfee:3900

Insert into specific column values:

Syntax:

INSERT INTO <TABLE NAME>(required column names) VALUES <&column name)>...; Ex:

SQL>INSERT INTO STUDENT (STID) VALUES (&STID);

Enter value for sid:1027

SQL> /;

Enter value for sid: 1028;

UPDATE:

To update all rows data in a table at a time

Or

To update a specific row data in a table by using “WHERE” clause condition. Syntax:

UPDATE <TABEL NAME> SET <column name1>=<value1>,<column name2>=<value2>...[WHERE <condition>;

Ex:

SQL>UPDATE STUDENT SET SFEE = 6000 WHERE SNAME=’ADAMD’; **SQL>**UPDATE STUDENT SET

SNAME = ‘WARD’,SFEE=3000 WHERE STID=1027; **SQL>**UPDATE STUDENT SET SNAME = NULL,

WHERE SNAME=’JONES’; **SQL>**UPDATE STUDENT SET STID=NULL,SNAME=NULL,SFEE=NULL

WHERESTID=1029;

SQL>UPDATE STUDENT SET STID=1026,SNAME = ‘WARD’,SFEE=3000 WHERESTIDIS1026;

SQL>UPDATE STUDENT SET SFEE=NULL;

SQL>UPDATE STUDENT SET SFEE=5000;

DELETE:

TO DELETE ALL ROWS FROM A TABLE AT A TIME.

OR

TO DELETE A SPECIFIC ROW FROM A TABLE BY USING “WHERE”

CLAUSECONDITIONSYNTAX:

DELETE FROM <TABLE NAME> [WHERE <CONDITION>];

EX:

SQL>DELETE FROM STUDENT WHERE STID=1029;

SQL>DELETE FROM STUDENT WHERE SNAME IS NULL;

SQL>DELETE FROM STUDENT;

DELETE
1. It is a DML operation
2. Deleting a specific rov
3. Supporting "WHERE" CLAUSE

4. Deleting data temporarily	
5. We can restore deleted datausing "ROLLBACK"	
6. Deleting rows in one-by-on	
7. Executing speed is slo	

21-10-2024

Data query/retrieve language:

- To retrieve all rows from a table at a time .

  OR

- To retrieve a specific rows from a table by using "WHERE" clause condition. Syntax:

    SELECT * FROM <TABLE NAME> [WHERE <condition>];

    Here "*" is representing all columns in a table.

Ex:

```
SQL>SELECT * FROM DEPT;
        OR
SQL>SELECT DEPTNO,DEPTNAME,LOC FROM DEPT; EX:
SQL>SELECT * FROM EMP WHERE JOB='MANAGER';
SQL>SELECT * FROM EMP WHERE EMPNO= 7698;
SQL>SELECT ENAME,DEPTNO,SAL FROM EMP WHERE DEPTNO=20;
```

ALIAS NAMES:

It is a temporary name / alternative name for columns/ table/expression.

We will create alias names at two levels.

- **column level alias:**

  Creating alias name for column name.

- **Table level alias:**

  Creating alias name for table name.

Syntax:

```
SELECT <COLUMN NAME> [AS] <COLUMN ALIAS NAME1>, <COLUMN
ALIASNAME2>[AS]<COLUMN ALIAS> ..................FROM <TABLE NAME><TABLE ALIAS NAME>;
```

EX:

SQL>SELECT DEPTNO X,DNAME Y,LOC Z FROM DEPT D; OR

SQL>SELECT DEPTNO AS X,DNAME AS Y,LOC AS Z FROM DEPT D;

DISTINCT keyword:

To eliminate duplicate values from specific columns.

Syntax:

SELECT DISTINCT <COLUMN NAME>;

EX:

SQL>SELECT DISTINCT DEPTNO FROM EMP;

SQL>SELECT DISTINCT JOB FROM EMP;

CONCATENATION OPERATOR(||):

- To combine two or more than two expressions.

Syntax:

<expression1> || <expression2> || <expression3>||........;

Ex:

SQL>SELECT 'THE EMPLOYEE' || ' ' || ENAME || ' ' || 'IS WORKING AS A ' || ' ' || JOBFROMEMP;

NOTE: To display a table data in proper systematical way in SQLplus editor.then we need toset thefollowing two properties are;

PAGESIZE N:

To show no of rows in a page.

Here 'n' is represent no.of rows in a page.

The maximum size of pagesize property is 50000.

Syntax:

SET PAGESIZE N;

Ex:

SQL>SET PAGESIZE 100;

LINES N;

TO display no.of characters in a single row /line.

Here "n" is represent no.of characters

The maximum size of line property is 32767

Syntax:

SET LINES N;

EX:

SQL>SELECT LINES 160;

22-10-2024

Operators in oracle SQL:

• To perform some operation on the given operand values Oracle SQL supports

the following operators are:

1. Assignment operator (=)
2. Arithmetic operator (+,-,/,*)
3. Relational operator(<,>,<=,>=,!=or<>) 4. Logical operator (AND, OR, NOT)
5. Set operator (UNION,UNION ALL,INTERSECT,MINUS) 6. Special operator (+ve) (-ve)

+ve	
in	
Between	
Is null	
like	

Assignment operator:

• To assign a value to variable /to attribute

Syntax:

<column name><assignment operator> <value>

Ex:

SQL>UPDATE EMP SET SAL=5000 WHERE EMPNO = 7780; **SQL>**UPDATE EMP

SET JOB=’HR’

Arithmetic operator:

• To perform addition,substraction,multiplication,division

Syntax:

<column name> <arithematic operator><value>

Ex:

Waq to display all employee salaries after adding 2000/-?

SQL>SELECT ENAME,SAL AS OLD_SALARY, SAL+200 AS NEW_SALARY FROMEMP; EX:

waq to display EMPNO,ENAME,BASIC SALARY AND ANNUAL SALARY OF THE EMPLOYEESWho are working as a manager?

SQL>SELECT EMPNO,ENAME,SAL AS BASIC_SALARY,SAL*12 AS

ANNUAL_SALARYFROMEMP WHERE JOB='MANAGER';

Ex:

Waq to display all employees salaries after increment of 10%?

SQL>SELECT ENAME,SAL AS BEFORE_INCREMENT,SAL+SAL*10/100 AS AFTER_INCREMENT FROM EMP;

EX:

WAQ to display ENAME,DEPTNO,BASIC_SALARY,SALARY_INCREMENT OF 5%AMOUNTandTOTAL_SALARY of the employees who are working under deptno is 20?

SQL>SELECT ENAME,DEPTNO,SAL AS BASIC_SALARY,SAL*0.05 AS SALARY_INCREMENT ,SAL + SAL*0.05 AS AFTER_INCREMENT FROMEMPWHEREDEPT=20;

EX.

Waq display EMPID,ENAME,JOB,HIREDATE,BASE_SALARY,10% HRA,20%of DA,5%PFGROSS_SALAND ALSO NET_SALARY OF EMPLOYEES

SQL>SELECT EMPID,ENAME,JOB,HIREDATE,SAL AS BASE_SALARY,SAL*0.1 ASHRA,SAL*0.2 AS DA,SAL * 0.05 AS PF , SAL+ SAL*0.1 + SAL*0.2 +SAL*0.05 ASGROSS_SALARY, SAL+ SAL*0.1 + SAL*0.2 -SAL*0.05 AS NET_SALARY FROMEMP;

23-10-2024

Waq to display all employee salaries after decrement of 10%?

SQL>SELECT ENAME,SAL AS BEFORE_DECREMENT ,SAL-SAL*10/100 ASAFTER_DECREMENT FROM EMP;

RELATIONAL OPERATORs:

– Comparing a specific column values with user defined condition in the query. Syntax:

Where <column name><relational operator><value>;

Ex:

Waq to display list of employee who are joined after 1981? **SQL>**SELECT * FROM EMP WHERE HIREDATE > '31-DEC-1981';

Waq to display list of employee who are joined BEFORE 1981? **SQL>**SELECT * FROM EMP WHERE HIREDATE < '1-JAN-1981';

LOGICAL OPERATORs:

– To check more then one condition in query AND OR NOT

AND OPERATOR:

– **It** returns a value if both conditions are true in the query.

Condi 1	Condi
true	true
true	false
false	true
false	false

Ex:

Waq to display employee who are working SALESMAN who name is TURNER **SQL>**SELECT * FROM EMP WHERE JOB='SALESMAN' AND ENAME='TURNER';

OR OPERATOR:

– It returns if any one condition is TRUE in the query.

Condi 1	Condi
true	true
true	false
false	true
false	false

EX:

waq to display employees whose EMPNO is 7369,7566,7788? **SQL>** SELECT * FROM EMP WHERE EMPNO=7369 OR EMPNO=7566 OREMPNO=7788; EX:

Waq to display employees who are working as "PRESEDENT" or whose salary is more thenor isequalto 3000?

SQL> SELECT * FROM EMP WHERE JOB = 'PRESEDENT' OR SAL >=3000;

NOT OPERAATOR:

– It return all values except the given conditional values in the query. Syntax:

Where NOT <condition1> AND NOT<condition2>;

Ex:

Waq to display employees who are not working as a "MANAGER" and as a "ANALYST" **SQL>** SELECT * FROM EMP WHERE NOT JOB='MANAGE' AND NOT JOB='ANALYST';

SET OPERATOR:

– Set operators are to combined the results of two select statements.

Syntax:

<select query1> <set operator> <select query2>;

Ex;

A = {10,20,30} B = {30,40,50}

UNION:

– It combined the value of two sets without duplicates

A U B = {10,20,30,40,50}

UNION ALL

– It combine the value of two sets with duplicates

A UL B = {10,20,30,30,40,50}

INTERSECT:

– It return the common values from both sets.

A | B = {30}

MINUS:

– It return uncommon values from the left side set but not the right side set A-B ={10,20}

B-A={40,50}

DEMO TABLES:

SQL> SELECT * FROM EMP_HYD;

1023	MILLER	6

SQL>SELECT * FROM EMP_MUMBAI;

EID	ENAME	SA
1021	SMITH	850
1024	WARD	380

EID	
1021	SMI
1022	ALL

24-10-24

Waq to fetch employees who are working in hyd but not in MUMBAI branch? **SQL>** SELECT * FROM EMP_HYD minus SELECT * FROM EMP_MUMBAI

Waq to fetch employees who are working in both branches? **SQL>** SELECT * FROM EMP_HYD

intersect SELECT * FROMEMP_MUMBAI Waq to fetch all employees details who are working in

the organization?

SQL> SELECT * FROM EMP_HYD union all SELECT * FROMEMP_MUMBAI(withduplicate)

Waq to fetch all employees details who are working without duplicate?

SQL> SELECT * FROM EMP_HYD union SELECT * FROM EMP_MUMBAI(without duplicate)

SPECIAL OPERATORS:

IN operator:

Comparing the list of values with single condition.

Syntax:

Where <column name> IN <value1,value2....);

Where <column name> NOT IN <value1,value2....);

Ex:

Waq to display employees whose EMPNO is 7566,7788,7900,

SQL> SELECT * FROM EMP WHERE EMPNO IN(7566,7788,7900); Waq to display employees who are not as a “CLERK” ,”SALESMAN”,”MANAGER”. **SQL>** SELECT * FROM EMP WHERE JOB NOT IN (‘CLERK’,’SALESMAN’,’MANAGER’);

BETWEEN operator:

– Comparing a particular range values

Syntax:

Where <column name> BETWEEN<low value> AND <high value>; Where <column name>NOT BETWEEN <low value> AND <high value>; Ex:

Waq to list of employee who are joined in the year of 1981

SQL>SELECT * FROM EMP WHERE HIREDATE BETWEEN ‘01-JAN-1981’ AND‘31-DEC-2024;

NOTE :BETWEEN operator return all values including source value and also destinationvalue**IS NULL operator:**

– Comparing NULLS in a table.

Syntax:

Where <column name> IS NULL;

Where <column name > IS NOT NULL;

Ex:

Waq to display employees whose commission is NULL/UNDEFINED/EMPTY/UNKNOWN?**SQL>**SELECT * FROM EMP WHERE COMM IS

NULL;

Waq to display employees whose commission is NOT NULL/NOT UNDEFINED/NOTEMPTY/NOT UNKNOWN?

SQL>SELECT * FROM EMP WHERE COMM IS NOT NULL; **WORKING WITH**

NULL:

It is an empty / a undefined value/unknown value in database.

NULL != 0 , NULL!= space.

Ex:

Waq to display EMPNO, ENAME,JOB,SAL,COMM AND SAL+ COMMfromEMPtablewhoseemployee name is “SMITH”

SQL>SELECT EMPNO,ENAME,JOB,SAL,COMM, SAL + COMMAS TOTAL_SALARYFROM EMP WHERE ENAME = ‘SMITH’

OUTPUT:

EMPNO ENAME JOB SAL COMM TOTAL_SALARY7369 SMITH CLERK 800

In the above example the employee SMITH salary is 800 and there is no commissionsothatSAL+ COMM is 800 only BUT it returns NULL.

To overcome the above problem oracle will provide a pre-defined function is NVL(). **What is**

NVL(expl1,expl2):

to replace a user defined value in-place of NULL In the expression. This function is

having two arguments are expression1 and expression2.

• If EXP1 is NULL then it return EXP2 value (user defined value) • If EXP1 is NOT NULL then it return EXP1 value only.

Ex:

SQL>SELECT NVL(NULL,0) FROM DUAL; ——————0;

SQL>SELECT NVL(NULL,100) FROM DUAL; ——————100;

SQL>SELECT NVL(0,1000 FROM DUAL;—————0;

SQL>SELECT NVL(500,200) FROM DUAL ————-500;

Solution:

SQL>SELECT EMPNO,ENAME,JOB,SAL,COMM,SAL + NVL(COMM,0) ASTOTAL_SALARY FROM EMP WHERE ENAME=’SMITH’;

OUTPUT:

EMPNO ENAME JOB SAL COMM TOTAL_SALARY7369 SMITH 800 800

25-10-2024

NVL2():

It is an extension of NVL function it contains 3 arguments (expr1,expre2,expre3). If exp1 in NULL the it return EXP3 value(user defined value)

If exp1 is NOT NULL then it return EXP2 value(user defined value).

Syntax:

Nvl2(exp1,exp2,exp3);

SQL>SELECT NVL2(NULL,100,200) FROM DUAL; ————-200 **SQL>**SELECT NVL2(500,100,200) FROM DUAL;—————100 Ex:

Waq to update all employees COMM in the table based on the following condition. 1) if the employee COMM is NULL then update those employee commissions as 8002)if the employee COMM is NOT Null then update those employee commission as COMM+300**SQL>**UPDATE EMP SET COMM=NVL2(NULL,COMM+300,800);

****LIKE OPERATOR:**

Comparing a specific string character pattern wise.

When we use "LIKE" operator we must use the following two wild-cards operators are "%" it represent the running group of characters after selected character fromexpression. "_" counting a single character from the expression.

Syntax:

Where <column name> LIKE '[<wild-card operator>] <character pattern> [wild-card operator]'

26-10-2024

Ex:

waq to fetch employees whose name start with "s" character? **SQL>**SELECT * FROM EMP WHERE ENMAE LIKE 'S%';

EX:

Waq to fetch employee whose name ends with "R" character? **SQL>**SELECT * FROM EMP WHERE ENAME LIKE '%R';

EX

Waq to fetch employee whose name is having "I" character? **SQL>** SELECT * FROM EMP WHERE ENAME LIKE '%I%'; Ex:

Waq To fetch employee whose name starts with and ends with M and N? **SQL>** SELECT * FROM EMP WHERE ENAME LIKE 'M%N' Ex:

Waq to fetch employee whose name is having the second position character is 'O'? **SQL>**SELECT *

FROM EMP WHERE ENMAE LIKE '_O%';

EX:

Waq to fetch employee whose name is having 4 characters? **SQL>** SELECT * FROM EMP WHERE ENMAE LIKE '____'; (4 UNDERSCORESwithout space)Ex:

Waq to fetch employes whose EMPNO start with 7 ends with 8? **SQL>** SELECT * FROM EMP WHERE EMPNO '7%8'; EX:

Waq To fetch employees who are join in 1981?

SQL> SELECT * FROM EMP WHERE HIREDATE LIKE '%81'; (not 1981) Waq to fetch employee who are joined in "DECEMBER"?

SQL>SELECT * FROM EMP WHERE HIREDATE LIKE '%DEC%'; Waq to fetch employee who are joined in "DECEMBER" in "1982"? **SQL>**SELECT * FROM EMP WHERE HIREDATE LIKE '%DEC%82'; OR

SQL>SELECT * FORM EMP WHERE HIREDATE LIKE '%DEC_82'; OR

SQL>SELECT * FROM EMP WHERE HIREDATE LIKE '%DEC-82'; OR

SQL>SELECT * FROM EMP WHERE HIREDATE LIKE' %DEC%' AND HIREDATELIKE'%82';Ex:

To fetch employee who are joined in the month of 'JUNE' and 'DECEMBER'?

SQL>SELECT * FROM EMP WHERE HIREDATE LIKE '%JUN%' OR HIREDATELIKE'%DEC%';

28-10-2024

LIKE operator on special operator:

DEMO TABLE:

SQL>SELECT * FROM TEST;

```
CID CNAME
1 BHUVIN_KUMAR
2 WARN@RER
3 #YUVIN
4 _ADAMS
5 JON%ES
```

Ex:
waq to fetch customer details name is having "@" symbol?

SQL>SELECT * FROM TEST WHERE CNAME LIKE '@';

Ex:
waq to fetch customer details name is having '#' symbol?

SQL>SELECT * FROM TEST WHERE CNAME LIKE '#'

Ex:

Waq to fetch customer details name is having '_' symbol?

SQL>SELECT * FROM TEST WHERE CNAME LIKE '%_%'; ——-NOT WORKINGEx:

Waq to fetch customer details name is having '%' symbol?

SQL>SELECT * FROM TEST WHERE CNAME LIKE '%%%'; ————-NOT WORKING

– When we fetch data from a table based on "_" and "%" symbols oracle server will treat aswild-card operators but not special characters.
– To overcome the above problem we must use a keyword is "ESCAPE'\'". **SOLUTION:**

SQL>SELECT * FROM TEST CNAME LIKE '%_%'ESCAPE'\';
SQL>SELECT * FROM TEST CNAME LIKE '%\%%'ESCAPE'\';

Ex:

Waq to fetch employees details whose name not starts with "S" character? **SQL>**SELECT *

FROM TEST CNAME NOT LIKE 'S%';

FUNCTIONS IN ORACLE:

– To perform some tasks as per the given input values and it must be return a value – Oracle supports the following two types of functions those are 1. Pre-defined function
– Use in SQL & PL/SQL
2. User-defined function
– Use in PL/SQL only

Pre-defined functions:

These functions are also called as "built-in-functions"

It again two types:

1. Single row function
2. Multiple row function

Single row function:

These functions are always return a single value.

There are few types of single row functions.

– Numeric functions
– Character functions
– Date functions
– Null function(NVLf(),NVL2())
– Conversion function
– Analytical function

How to call a function in Oracle:

Syntax:

SELECT <FNAME>(value(s)) FROM DUAL;

What is DUAL?

– It is pre-defined table in oracle.
– It is used to test function functionality (i.e work flow).
– It contains a single row & a single column.
– It is also called as "Dummy Table" in oracle.

How to view the structure of DUAL table?

Syntax:

DESC <table name>

Ex

SQL>DESC DUAL;

How to view data of DUAL table:

Syntax:

SELECT * FROM <table name>;

Ex:

SQL> SELECT * FROM DUAL;

Numeric functions:

ABS():

To convert (-ve) sing values into (+ve) sign values.

Syntax:

Abs(n);

Ex:

SQL>SELECT ABS(-12) AS RESULT FROM DUAL;

<u>RESULT</u>

12

EX:

Waq to display ENAME,SALARY,COMMISSION, and COMMISS-SAL from EMP table? **SQL>**SELECT ENAME,SAL,COMM,ABS(COMM-SAL) AS RESULT FROMEMP; **CEIL():**

– It returns a value which is grater than to (or) is equals to given expression. Syntax:

Ceil(n);

Ex:

SQL>SELECT CEIL(9.0) AS RESULT FROM DUAL;

RESULT

9

EX:

SQL>SELECT CEIL(9.1) AS RESULT FROM DUAL; RESULT

10

FLOOR():

– It returns a value which is less than to (or) is equals to the given expression. Syntax:

Floor(n)

Ex:

SQL>SELECT FLOOR(9.0) AS RESULT FROM DUAL; RESULT

9

SQL>SELECT FLOOR(9.1) AS RESULT FROM DUAL; RESULT

9

SQL>SELECT FLOOR(9.8) AS RESULT FROM DUAL; RESULT

9

29-10-2024

Ex:

SQL>SELECT ENAME ,CEIL(SAL*0.05) AS RESULT FROM EMP; **SQL>**SELECT

ENAME,FLOOR(SAL*0.05) AS RESULT FROM EMP; **POWER():**

– It returns the power of the given expression.

Syntax:

Power(m,n)

Ex:

SQL>SELECT POWER(2,3) FROM DUAL; ————————-> 8 **SQL>**SELECT

POWER(SAL,2) FROM EMP;

MOD():

It returns the reminder value of the expression

Syntax:

mod (m,n)

Ex:

SQL>SELECT MOD(10,2) FROM DUAL;———-0

SQL>SELECT * FROM EMP WHERE MOD(EMPNO,2)=0;(FOR EVEN EMPNO) **SQL>**SELECT *

FROM EMP WHERE MOD(EMPNO,2)=1;(F0R ODD EMPNO) **ROUND():**

– It returns nearest value of the given expression based on 0.5 value. ▪ If the expression is having less then 0.5 then it return ——-> 0 ▪ If the expression is having less grater then or is equal to 0.5 then it return——-1

Syntax:

Round(expression[,decimal places])

EX:

SQL>SELECT ROUND(56.2) AS RESULT FROM DUAL; 56.2=====>

0.2<0.5=====>0

<u>+ 0</u>

56

SQL>SLELECT ROUND(56.5) AS RESULT FROM DUAL; 56.5=====>

0.5<=0.5=====>1

<u>+ 1</u>

57

SQL>SELECT ROUND(56.8) AS RESULT FROM DUAL; 56.8=====>

0.8<=0.5=====>1

<u>+ 1</u>

57

SQL>SELECT ROUND(56.870,2) AS RESULT FROM DUAL; 56.87====>0.0 < 0.5

====> 0

<u>+0</u>

56.87

SQL>SELECT ROUND(56.875,2) AS RESULT FROM DUAL; 56.87====>0.5 <= 0.5

====> 1

<u>+1</u>

56.88

SQL>SELECT ROUND(56.877,2) AS RESULT FROM DUAL; 56.87====>0.7 < 0.5

====> 1

<u>+1</u>

56.88

TRUNC():

– It returns an exact value from the given expression.
– It doesn’t depends on 0.5 value.

Syntax:

trunc(expression[,decimal place]);

Ex:

SQL>SELECT TRUNC(56.2) AS RESULT FROM DUAL;———-56 **SQL>**SELECT

TRUNC(56.5) AS RESULT FROM DUAL;————56 **SQL>**SELECT TRUNC(56.8) AS

RESULT FROM DUAL;————56

EX:

SQL>SELECT TRUNC(56.82,1) AS RESULT FROM DUAL;——56.8 **SQL>**SELECT

TRUNC(56.85,1) AS RESULT FROM DUAL;——56.8 **SQL>**SELECT TRUNC(56.89,1) AS RESULT

FROM DUAL;——56.8 **Character / string function:**

LENGTH():

– It return the length of the given string.

Syntax:

length(string)

Ex:

SQL>SLECT LENGTH (‘HELLO’) FROM DUAL;—————->5 **SQL>**SELECT

LENGTH(‘WEL COME’) FROM DUAL;—————->8

Ex:

SQL>SELECT ENAME, LENGTH (ENAME) AS RESULT FROM EMP; **SQL>**SELECT * FROM

EMP WHERE LENGTH (ENAME)<5;

SQL>SELECT * FROM EMP WHERE LENGTH (ENAME)=5;

SQL>SELECT * FROM EMP WHERE LENGTH (ENAME)>5;

LOWER():

– It returns upper case characters into lower case characters.

Syntax:

lower(string)

Ex:

SQL>SELECT LOWER(‘HELLO’) FROM DUAL;———->hello

ex:

SQL>SELECT ENAME,LOWER(ENAME) FROM EMP; **SQL>** UPDATE EMP SET ENAME = LOWER(ENAME) WHERE JOB=’MANAGER’; **SQL>** UPDATE EMP SET ENAME = LOWER(ENAME);

UPPER():

– It converts lower case characters to upper case characters syntax:

Upper(string)

Ex:

SQL>SELECT UPPER(‘hello) FROM DUAL;———->HELLO EX:

SQL> UPDATE EMP SET ENAME = LOWER(ENAME);

INITCAP():

– It makes an initial character is capital from the given string. Syntax:

Initcap(string)

Ex:

SQL>SELECT INITCAP(‘HELLO’) FROM DUAL;——->Hello **SQL>**SELECT INITCAP(‘bhuvin kumar’) FROM DUAL;——->Hello

Ex:

SQL>SELECT ENAME,INITCAP(ENAME) FORM EMP; **SQL>**UPDATE EMP SET ENMAE=INITCAP(ENAME) ;

CONCAT():

– To add two strings expressions.

Syntax:

Concat(string1,string2)

Ex:

SQL>SELECT CONCAT (‘HI’,’BYE’) FROM DUAL;———>HAIBYE EX:

SQL> SELECT ENAME,CONCAT(‘Mr./Mis.’,ENAME) FROM EMP; **SQL>**UPDATE EMP SET ENAME = CONCAT(‘Mr.’,ENAME);

LTRIM():

– It remove unwanted characters from the given string on the left Syntax:

Ltrim(string,'<trimming character>')

Ex

SQL>SELECT LTRIM(' HELLO') FROM DUAL;——->HELLO **SQL>**SELECT LTRIM('XXXXXHELLO','X') FROM DUAL;———>HELLO **SQL>**SELECT LTRIM('XYZXYZXYZHELLO','XYZ') FROM DUAL;——>HELLO**RTRIM():**

– It remove unwanted character from the given string on the right side. Syntax:

Rtrim(string,<trimming character>')

Ex:

SQL>SELECT RTRIM('HELLO ') FROM DUAL;———>HELLO **SQL>**SELECT RTRIM('HELLOXXX','X') FROM DUAL;———>HELLO

30-10-2024

TRIM():

It removes unwanted characters from both sides of the given string.

Syntax:

Trim('trimming character' from STRING)

Ex:

SQL>SELECT TRIM('x', FROM 'XXXSMITHXXX') FROM DUAL;————->SMITHEx:

SQL>SELECT TRIM('XY', FROM 'XYXYSMITHXYXY') FROM DUAL; ERROR at line 1:

ORA-3000: trim set should have only character

REPLACE():

It replace string to string / string to character / character to string.

Syntax:

Replace(string,<old character>,'<new character>')

Ex:

SQL>SELECT REPLACE ('JACK AND JUE','J','BL') FROM DUAL;——-> BLACKANDBLUE**SQL>** SELECT REPLACE ('HELLO','ELL','D') FROM DUAL;———->HDO**SQL>** SELECT REPLACE ('HELLO','ELLO','XYZ') FROM DUAL;———->HXYZ **TRANSLATE():**

To translate each character by character

Syntax:

Translate(string,'<old string>','<new string>');

Ex:

SQL>SELECT TRANSLATE('HELLO','ELO','XYZ') FROM DUAL;——->HXYZ Here, E=X,L=Y,O=Z

SQL>SELECT TRANSLATE('HELLO','ELO','AB') FROM DUAL;——->HABB Here E=A,L=B

LPAD():

– Filling a specified character on the left side of the given string if string lengthislessthentouser defined length.

Syntax:

Lpad(string,<user defined length>,'<specified character>');

Ex:

SQL>SELECT LPAD('HELLO',1) FROM DUAL;————-> H **SQL>**SELECT LPAD('HELLO',10)

FROM DUAL;————> HELLO(5 SPACES)

SQL>SELECT LPAD('HELLO',10,'A') FROM DUAL;—————>AAAAAHELLO(5 AUSERDEFINED)

RPAD():

– Filling a specified character on the right side of the given string if string lengthislessthento user defined length.

Syntax:

rpad(string,<user defined length>,'<specified character>');

Ex:

SQL>SELECT LPAD('HELLO',10,'A') FROM DUAL;—————>HELLOAAAAA(5 AUSERDEFINED)

SUBSTR():

It return the required sub string from the given string expression.

Syntax:

Substr(string,<string position of character>,<no.of character>)

Expression:

-7	-	-	-	-	-	-
W			(		O	ME
1						

Ex:

SQL>SELECT SUBSTR('WELCOME',1,1)FROM DUAL;——->W **SQL>**SELECT SUBSTR('WELCOME',4,2)FROM DUAL;———>CO **SQL>**SELECT SUBSTR('WELCOME',6,3)FROM DUAL;———->ME **SQL>**SELECT SUBSTR('WELCOME',-5,1)FROM DUAL;———->L **SQL>**SELECT SUBSTR('WELCOME',-7,5)FROM DUAL;————>WELCO**SQL>**SELECT SUBSTR('WELCOME',-4,2)FROM DUAL;————> Here no.of characters should not be (-ve) sign.

INSTR():

To find out occurrence position of the specified character

Syntax:

Instr(string,'<specified character>','<starting position of character>','<occurrence positionof specified character>')

Ex:

-13												-
W									E H			l
1												12

NOTE: we can search or required character from left to right (or) right to left in the string expressionbutthe position of characters are fixed.

1-11-2024

SQL>SELECT INSTR('WELCOME HELLO','L') FROM DUAL;———--> 3 **SQL>**SELECT INSTR ('WELCOME HELLO','L',1,1) FROM DUAL; ——>3 **SQL>**SELECT INSTR ('WELOCME HELLO','L',1,2) FROM DUAL ——->11 **SQL>**SELECT INSTR ('WELOCME HELLO','L',8,3) FROM DUAL ——->0 **SQL>**SELECT INSTR ('WELOCME HELLO','E',2,3) FROM DUAL ——->10 **SQL>**SELECT INSTR ('WELOCME HELLO','E',-8,3) FROM DUAL ——->0 **SQL>**SELECT INSTR ('WELOCME HELLO',E',-8,1) FROM DUAL ——->2

DATE FUNCTION:

SYSDATE():

It return the current date information of the system.

Syntax:

Sysdate()

Ex:

SQL>SELECT SYSDATE FROM DUAL;——->01-NOV-24

SQL>SELECT SYSDATE+10 FROM DUAL;——->11-NOV-24 **SQL>**SELECT SYSDATE-10 FROM DUAL;——->22-OCT-24

ADD_MONTHS():

It add/subtract no.of months from / to the given date expression.

Syntax:

Add_months(date,<no.of months>);

Ex:

SQL>SELECT ADD_MONTHS(SYSDATE,6)FROM DUAL;——->01-MAY-25 **SQL>**SELECT

ADD_MONTHS(SYSDATE,-6)FROM DUAL;——->01-MAY-24 **LAST_DAY():**

– It return the last day of the month in the given date expression.

Syntax:

Last_day(date)

Ex:

SQL>SELECT LAST_DAY(SYSDATE)FROM DUAL;——->30-NOV-24 **SQL>**SELECT

LAST_DAY('04-SEP-22')FROM DUAL;——->30-SEP-22 **MONTHS_BETWEEN():**

– It return no.of months in between in the given two dates.

Syntax:

Months_between(date1,date2);

Note:

– Date1 is must be grater than to date2 expression otherwise it return (-ve) signvaluesEx:

SQL>SELECT MONTHS_BETWEEN ('05-MAY-22','05-MAY-23)

FORMDUAL;—->-12**SQL>**SELECT MONTHS_BETWEEN ('05-MAY-23','05-MAY-22)

FORMDUAL;—->12**ANALYTICAL FUNCTIONS:**

– To assign rank numbers to each row wise (or) to each group or rows wise. – Oracle supports the following two types of analytical functions are, ▪ Rank()
▪ Dense_rank()
– The analytical functions are also called as "ranking functions".

Syntax;

Analytical function name() over ([partition by <column name>] order by <column name ><asc/desc>)

Here,
Partition by —- optional clause

Order by —- mandatory clause

4-11-24

Ex:

ENAME SALARY RANK() DENSE_RANK()

A 85000 1 1

B 72000 2 2

C 72000 2 2

D 68000 4 3

E 55000 5 4

F 55000 5 4

G 48000 7 5

H 32000 8 6

NOTE:rank() it is skip numbers but dense_rank() not skip number

EX ON WITHOUT PARTITION BY CLAUSE:

SQL>SELECT ENAME,SAL,RANK() OVER (ORDER BY SAL DESC) AS RANKFROMEMP;

SQL>SELECT ENAME,SAL,DENSE_RANK() OVER (ORDER BY SAL DESC) ASRANKFROMEMP;

EX ON WITH PARTITION BY CLAUSE:

SQL>SELECT ENAME,DEPTNO,SAL,RANK() OVER(PARTITION BY DEPTNOORDERBYSALDESC) AS RANKS FROM EMP;

SQL>SELECT ENAME,DEPTNO,SAL,DESC_RANK() OVER(PARTITION BYDEPTNOORDERBY SAL DESC) AS RANKS FROM EMP;

CONVERSION FUNCTIONS:

1. TO_CHAR()
2. TO_DATE()

TO_CHAR():

– To convert data type to character(string) type and also display date in different formats. Syntax:

To_char(sysdate,’<intervals>’)

YEAR FORMATS:

YYYY - Year In Four Digits Format

YY - Last Two Digits From Year YEAR - Twenty Twenty-Four

CC - Century 21

AD/BC - AD YEAR / BC YEAR **Ex:**

SQL> SELECT TO_CHAR (SYSDATE,’YYYY YY YEAR CC BC’) FROMDUAL; **OUTPUT**:

2024 24 TWENTY TWENTY-FOUR 21 AD

MONTH FORMAT:

MM - Month In Number Format

Mon - First Three Char’S From Month Spelling Month - Full Name Of Month

Ex:

SQL>SELECT TO_CHAR(SYSDATE,’MM MON MONTH’) FORM DUAL; OUTPUT:

11 NOV NOVEMBER

DAY FORMAT:

DDD - DAY OF THE YEAR

DD - DAY OF THE MONTH D - DAY OF THE WEEK

SUN - 1

MON -2

TUE -3

WED -4

THU -5

FRI -6

SAT -7

DAY - FULL NAME OF THE DAY DY - FIRST THREE CHAR’S OF DAY SPELLING

EX:

SQL>SELECT TO_CHAR(SYSDATE,’DDD DD D DY DAY) FROM DUAL; OUTPUT:

309 04 2 MON MONDAY

QUATER FORMAT:

Q – one digit quater of the year

1-JAN-MAR

2-APR-JUN

3-JUL-SEP

4-OCT-DEC

Ex:

SQL>SELECT TO_CHAR(SYSDATE,’Q’) FROM DUAL; **OUTPUT:**

4

WEEK FORMAT:

WW - WEEK OF THE YEAR W - WEEK OF THE MONTH EX:

SQL>SELECT TO_CHAR (SYSDATE,’WW W’) FROM DUAL; OUTPUT:

45 1

TIME FORMAT:

HH - HOUR PART IN 12HR FORMAT HH24 - HOUR PART IN 24HRS FORMAT

MI - MINUTE PART

SS - SECONDS PART

AM/PM - AM TIME OR PM TIME

EX:

SQL>SELECT TO_CHAR(SYSDATE,’HH HH24 MI SS PM’)FROM DUAL; OUTPUT:

10 10 22 58 AM

EX:

WAQ to display list of employee who are joined in 1981 by using to_char()? **SQL>**SELECT *

FROM EMP WHERE TO_CHAR(HIREDATE,’YYYY’)=’1981’; OR

SQL>SELECT * FROM EMP WHERE TO_CHAR(HIREDATE,’YY’)=’81’; OR

SQL>SELECT * FROM EMP WHERE TO_CHAR(HIREDATE,’YEAR’)=’NINTEENEIGHTY-ONE’;

EX:

WAQ to display list of employee who are joined in 1980,1982,1983 By using to_char()?

SQL>SELECT * FROM EMP WHERE TO_CHAR(HIREDATE,’YYYY’)= ’1980’
ORTO_CHAR(HIREDATE,’YYYY’)= ’1982’ OR
TO_CHAR(HIREDATE,’YYYY’)=’1983’;

OR

SQL>SELECT * FROM EMP WHERE TO_CHAR(HIREDATE,’YYYY’) IN(‘1980’,’1982’,’1983’);1

5-11-24

waq to display list of employee who are joined in the month of DECEMBER? **SQL>**SELECT * FROM

EMP WHERE TO_CHAR(HEIREDATE,’MM’) = ‘12’;

waq to display list of employee who are joined in the month of DECEMBER IN 1982?

SQL>SELECT * FROM EMP WHERE TO_CHAR(HIREDATE,’MM’)=’12’ AND

TO_CHAR(HIREDATE,’YYYY’)=’1982’;

OR

SQL>SELECT * FROM EMP WHERE TO_CHAR(HIREDATE,’MMYYYY’)=’121982’;
Waq to display list of employee who are joined on monday?

SQL>SELECT * FROM EMP WHERE TO_CHAR(HIREDATE,’D’)=’2’;

OR

SQL>SELECT * FROM EMP WHERE TO_CHAR(HIREDATE,’DY’)=’MON’; OR

SQL>SELECT * FROM EMP WHERE TO_CHAR(HIREDATE,’DAY’)=’MONDAY’; Waq to

display list of employee who are joined on weekends?

SQL>SELECT * FROM EMP WHERE TO_CHAR(HIREDATE,’D’) = ‘7’ OR
TO_CHAR(HIREDATE,’D’)=’1’;

OR

SQL>SELECT * FROM EMP WHERE TO_CHAR(HIREDATE,’DY’) = ‘SAT’
ORTO_CHAR(HIREDATE,’DY’)=’SUN’;

OR

SQL>SELECT * FROM EMP WHERE TO_CHAR(HIREDATE,’DAY’) = ‘SATURDAY’
ORTO_CHAR(HIREDATE,’DAY’)=’SUNDAY’;

OR

SQL>SELECT * FROM EMP WHERE TO_CHAR(HIREDATE,’DY’) IN (‘SAT’,’SUM’);

Waq to display list of employees who are joined in 2nd week of JUN ?

SQL>SELECT * FROM EMP WHERE TO_CHAR(HREDATE,’MM’)=’6’ AND
TO_CHAR(HIREDATE,’W’)=’2’;

OR

SQL>SELECT * FROM EMP WHERE TO_CHAR(HIREDATE,’WMM’)=’206’;

Waq to display the list of employee who are joined 4th quater of 1981 **SQL>**SELECT * FROM EMP

WHERE TO_CHAR(HIREDATE,’QYYYY’)=’41981’;

ii) TO_DATE

To convert string type to oracle default date type.

Syntax:

To_date(string)

Ex:

SQL>SELECT TO_DATE('23/AUGUST/2022') FROM DUAL;——>23-AUG-22 **SQL>**SELECT

TO_DATE('23/AUGUST/2022) + 5 FROM DUAL;—->28-AUG-22 **SQL>**SELECT

TO_DATE('23/AUGUST/2022) -5 FROM DUAL;—->18-AUG-22

MULTIPLE ROW FUNCTIONS:

These functions as also called as "grouping functions " / "aggregative function These functions are working on group of values of a column

Syntax:

<aggregative function name>(<column name>)

Sum():

– It return total value.

Ex:

SQL>SELECT SUM(SAL) FROM EMP;

SQL>SELECT SUM(SAL) FROM EMP WHERE DEPTNO = 20; **AVG():**

– It return the average value.

EX:

SQL>SELECT AVG(SAL) FROM EMP;

SQL>SELECT AVG(SAL) FROM EMP WHERE DEPTNO=20;

MIN():

– It returns the minimum value.

EX:

SQL>SELECT MIN(SAL) FROM EMP;

SQL>SELECT MIN(HIREDATE) FROM EMP;

SQL>SELECT MIN(SAL) FROM EMP WHERE DEPTNO=30;

MAX():

It returns the maximum value.

Ex:

SQL>SELECT MAX(SAL) FROM EMP;

COUNT():

It again three formats

I). Count(*):

Counting all rows including duplicate and nulls in a table.

Ex;

SQL>SELECT COUNT(*) FROM EMP;—->14

II). COUNT(COLUMN NAME):

Counting all values including duplicate values but not nulls.

Ex:

SQL>SELECT COUNT(MGR) FROM EMP;——>13

III) COUNT (DISTINCT <COLUMN NAME>):

Counting unique values only.(no duplicates & no nulls)

Ex:

SQL> SELECT COUNT(DISTINCT MGR) FROM EMP;——>6

Assignments:

1. Waq to fetch employee who are working under DEPTNO is 20? Condition - without where

   clause

   SQL>SELECT * FROM EMP JOIN(SELECT 20 AS DEPTNO FROM DUAL) ASUMEONEMP.DEPTNO = ASUME.DEPTNO;

2. Waq to arrange the employee salaries in descending order? Condition – without

   ORDER BY clause

3. Waq to display no. of employee working under each JOB? Condition – without

   GROUP BY clause

4. Waq to display sum of salary of the job is more then 5000? Condition – without

   HAVING clause

6-11-24

CLAUSES:

It is a statement which is used to add to SQL query for providing some additional

facilitieslike“filtering rows, sorting values and grouping similar data” based on columns automatically.

Oracle supports the following clause are

- WHERE
- ORDER BY

• GROUP BY
• HAVING

Syntax:

<select query> + <clause statement>;

WHERE:

– Filtering rows before grouping the data in a table.
– It can be used in “SELECT,UPDATE,DELETE” commands only Syntax:

Where <filtering condition>

Ex:

SQL>SELECT * FROM EMP WHERE DEPTNO = 20;

SQL>UPDATE EMP SET SAL = 25000 WHERE JOB = ‘MANAGER’;

SQL>DELETE FROM EMP WHERE SAL = 3000;

ORDER BY:

– To arrange a specific column/(s) values in either in ascending or descendingorder. – By default order by clause will arrange the values in ascending order. If we want toarrangethe values in descending order then we must use “DESC” keyword.
– It can be used in “SELECT” command only

Syntax:

Select */<list of column> from <table name> order by <column name1> <asc/desc>,<columnname2> <asc/desc>,..........;

Ex:

SQL>SELECT * FROM EMP ORDER BY SAL ;(default order by arrange ascending

soneedasc)**SQL>**SELECT * FROM EMP ORDER BY SAL DESC;

SQL>SELECT * FROM EMP ORDER BY NAME;

SQL>SELECT * FROM EMP ORDER BY NAME DESC;

SQL>SELECT * FROM EMP ORDER BY HIREDATE;

SQL>SELECT * FROM EMP ORDER BY HIREDATE DESC;

Waq to display employees who are working under DEPNO is 20 and arrange those employeessalariesindescending order?

SQL>SELECT * FROM EMP WHERE DEPTNO = 20 ORDER BY SAL DESC;

WAq to arrange employees DEPTNO in ascending order and their salaries in descending order fromeachDEPTNO wise?

SQL>SELECT * FROM EMP ORDER BY DEPTNO , SAL DESC;

NOTE:ORDER BY clause not only column names even though we can apply on the positionof columninthe SELECT query.

Ex:

SQL>SELECT EMPNO,ENAME,SAL ORDER BY 3;

SQL>SELECT EMPNO,ENAME,SAL ORDER BY 2 DESC;

SQL>SELECT EMPNO,ENAME,SAL ORDER BY 1;

ORDER by NULL clauses:

There are two types of NULL clause:

1. NULLS FIRST
2. NULLS LAST

NULLS FIRST:

By default order by clause will arrange the nulls in ASCENDING order is: First:Values

Last : NULLs

Ex:

SQL>SELECT * FROM EMP ORDER BY COMM;

To overcome the above problem in ASCENDING order then we must use “NULLSFIRST”

clauseSolution:

SQL>SELECT * FROM EMP ORDER BY COMM NULLS FRIST; **NULLs LAST:**

By default order by clause will arrange the nulls in DESCENDING order is: First:NULLS

Last : values

Ex:

SQL>SELECT * FROM EMP ORDER BY COMM DESC; To overcome the above problem in DESCENDING order then we must use “NULLSLAST” clauseSolution:

SQL>SELECT * FROM EMP ORDER BY COMM NULLS LAST;

7-11-24

GROUP BY:

It is used to make groups based on column / columns.

When we use “BROUP BY ”clause we must “grouping /aggregative” functions to get theresult It can be used in “SELECT” query only.

Syntax:

Select <column name1>,<column name2>,.........,<aggregative function name1>,....from<tablename> GROUP BY <column name1>,<column name2>,.....;

Ex:

Waq to find out no.of employee working in a organization?

SQL>SELECT COUNT(*) AS TOTAL_NO_OF_EMPLOYEES FROM EMP; OUTPUT:

```
TOTAL_NO_OF_EMPLOYEE
            14
```

EX:

Waq to find out no.of employee under the job is SALESMAN?

SQL>SELECT COUNT(*) AS TOTAL_NO_OF_EMPLOYEES FROM EMP WHEREJOB=’SALESMAN’

OUTPUT:

```
TOTAL_NO_OF_EMPLOYEEi

        4
```

EX:

WAQ TO find out no.of employee are working under each job? **SQL>**SELECT JOB,COUNT(*) AS NO_OF_EMPLOYEES FROM GROUP BY JOB; OUTPUT:

```
JOB NO_OF_EMPLOYEES

CLERK 4

SALESMAN 4

ANALYST 2

MANAGER 3

PRESIDENT 1
```

EX:

WAQ to display no.of employees working under each job along with their DEPTNOwise?

SQL>SELECE DEPTNO,JOB,COUNT(*) AS NO_OF_EMPLOYEES FROMGROUPBYDEPTNO,JOB;

SQL>SELECE DEPTNO,JOB,COUNT(*) AS NO_OF_EMPLOYEES FROMGROUPBYDEPTNO,JOB ORDER BY DEPTNO:

EX:

WAQ to display sum of salaries of each DEPTNO wise

SQL>SELECT DEPTNO,SUM(SAL) AS NO_OF_SALARY FROM EMP GROUPBYDEPTNO;

SQL>SELECT DEPTNO,SUM(SAL) AS NO_OF_SALARY FROM EMP GROUPBYDEPTNOORDER BY DEPTNO;

EX:

Waq to display average and total salary of each DEPTNO wise?

SQL>SELECT DEPTNO,SUM(SAL) AS TOTAL_SAL,AVG(SAL) AS AVERAGE_SALFROMEMP GROUP BY DEPTNO ORDER BY DEPTNO;

GROUP BY clause with all aggregative functions:

SQL>SELECT DEPTNO,COUNT(*) AS NO_OF_EMPLOYEES,SUM(SAL) AS TOTAL_SALARY,AVG(SAL) AS AVG_SALARY,MAX(SAL) AS MAX_SALARY,MIN(SAL) ASMIN_SALARY FROM EMP GROUPT BY DEPTNO ORDER BY DEPTNO;

HAVING:

– Filtering rows after grouping the data in a table.
– HAVING clause should use after “GROUP BY” clause only.(WHERE clause should usebefore“GROUP BY”)

Syntax:

SELECT <column name1>,<column name2>,.........,<aggregative function name1>,....from<tablename> GROUP BY <column name1>,<column name2>,.....HAVING<filteringcondition>

Ex:

Waq to display DEPNO's from EMP table in which DEPTNO the no.of employees are workingmorethen3?

SQL>SELECT DEPTNO,COUNT(*) FROM EMP GROUP BY DEPTNO HAVINGCOUNT(*)>3ORDER BY DEPTNO;

EX:

Waq to display jobs from emp table if sum of salary of the job is less then 5000? **SQL>**SELECT

JOB,SUM(SAL) FROM EMP GROUP BY JAB HAVING SUM(SAL)<5000;

WHERE vs **HAVING**

WHERE	**HAVING**
1. Filtering rows before grouping datatable.	1. Filtering rows after grouping d in the table
2. WHERE condition will work on eachwise.	2. HAVING condition will work on e row of rows wise.
3. It does not supports "aggregativefunctions"	3. It supports "aggregative functions"
4. It will use before "GROUP B	4. It will use after "GROUP BY
5. Without "GROUP BY" clause WHEREclause be worked	5. Without "GROUP BY" clause HAVINGclause not worked.

8-11-24

USING ALL CALUSE IN SINGLE "SELECT" STATEMENT:

Syntax:

SELECT <column name1>,<column name2>,.........,<aggregative function name1>,....from<table name> [WHERE <filtering condition>GROUP BY <columnname1>,<column name2>,.....HAVING <filtering condition>ORDERBY <column name1><asc/desc>];

Ex:

SQL>SELECT DEPTNO,COUNT(*) FROM EMP WHERE SAL>1000 GROUP BYDEPTNOHAVING COUNT(*)>3 ORDER BY DEPTNO;

```
DEPTNO COUNT(*)

20 4

30 5
```

JOINS:

– In RDBMS data can be restore in multiple table. From those multiple table if wewant toretrieve the required data / information then we should use a technique is knownas“JOINS”.– JOINS are used to retrieving data / information from multiple table at a time. – Oracle supports the following types of joins are:

1. Inner join
   - Equi join
   - Non-equi join
   - Self join
2. Outer joins
   - Left outer join
   - Right outer join
   - Full outer join
3. Cross join / cartisean join
4. Natural join

HOW TO JOIN TWO TABLES:

Syntax:

Select */<list of columns> from <table name1><join key> <table name2> on <joiningcondition>;

EQUI JOIN:

– When we retrieve data from multiple tables based on equal”(=)” operator JOINconditioniscalled as “EQUI JOIN”

– When we use EQUI join we should maintain at least one common column in bothtablesandthose common columns data types must be match / same.

– Whenever you want to join multiple tables there is no need to maintain relationshipbetweentables. Here relationship is optional

– By using EQUI JOIN we always retrieve matching rows (data) frommultiple tables. Syntax:

On <table name1>.<common column>=<table name2>.<common column>; Ex:

DEMO tables:

SQL> SELECT * FROM COURSE;

<u>CID CNAME CFEE</u>

1 ORACLE 2500

2 JAVA 5000

3 PYTHON 8000

SQL>SELECT * FROM STUDENT;

<u>STID SNAME CID</u>

1021 SMITH 1

1022 ALLEN 1

1023 JONES 2

1024 ADAMS

Ex:

Waq to retrieve STUDENTS and the corresponding COURSE details? **SQL>**SELECT *

FROM STUDENT JOIN COURSE ON CID=CID; **ERROR** at line 1:

ORA-00918: COLUMN AMBIGUOUSLY DEFINED.

– To overcome the above problem then we should use a table name as an identitytoacolumnlike below

Solution:

SQL>SELECT * FROM STUDENT JOIN COURSE ON STUDENT.CID=COURSE.CID; OR

SQL>SELECT STID,SNAME,CNAME,CFEE FROM STUDENT JOIN COURSEONSTUDENT.CID=COURSE.CID;

OR

SQL>SELECT STID,SNAME,CNAME,CFEE FROM STUDENT S INNER JOINCOURSECONS.CID=C.CID;

9-11-24

RULES FOR JOINS:

• A row in the table is comparing with all rows of the second table. Ex:

Waq to display student and course details for the tables who are join in oracle course?

SQL>SELECT SNAME,CNAME FROM STUDENT S INNER JOIN COURSE CONS.CID=C.CID WHERE CNAME = ‘ORACLE’;

OR

SQL>SELECT SNAME,CNAME FROM STUDENT S INNER JOIN COURSE CONS.CID=C.CID AND CNAME = ‘ORACLE’;

S.CID=C.CID CNAME = ‘ORACLE’

JOIN CONDITION FILTER CONDITION EX:

Waq to display employees who are working in the location is ‘CHICAGO”?

SQL>SELECT ENAME,LOC FROM EMP E INNER JOIN DEPT D ON E.DEPTNO=D.DEPTNOAND LOC = ‘CHICAGO’;

ASSIGNMENT:

Waq to display sum of salaries each dept names from EMP,DEPT tables?

SQL>SELECT DNAME, SUM(SAL) FROM EMP E INNER JOIN DEPT D ONE.DEPTNO= D.DEPTNO GROUP BY DNAME;

Waq to display DEPTNO and also sum of salaries of each department name wise fromEMP,DEPTtables?

SQL>SELECT DNAME, D.DEPTNO, SUM(SAL) FROM EMP E INNER JOINDEPTDONE.DEPTNO = D.DEPTNO GROUP BY DNAME, D.DEPTNO;

Waq to display no.of employees working in each department name from EMP,DEPT tables?

SQL>SELECT DNAME, COUNT(*) AS NUM_EMPLOYEES FROM EMP E INNERJOINDEPT D ON E.DEPTNO = D.DEPTNO GROUP BY DNAME;

Waq to display department names from EMP,DEPT tables in which department name the no.of employeesare less then to 3

SQL>SELECT DNAME FROM EMP E INNER JOIN DEPT D ON E.DEPTNO= D.DEPTNOGROUP BY DNAME HAVING COUNT(*) < 3;

11-11-24

OUTER JOIN:

There are three types:

LEFT OUTER JOIN:

Retrieving matching rows both tables and unmatching rows form left side of the table. Ex:

SQL>SELECT * FROM STUDENT S LEFT OUTER JOIN COURSE C ON S.CID=C.CID; **SQL>**SELECT * FROM COURSE C LEFT OUTER JOIN STUDENT S ON C.CID=S.CID; **RIGHT OUTER JOIN:**

Retrieving matching rows both tables and un matching rows form left side of the table. Ex:

SQL>SELECT * FROM STUDENT S RIGHT OUTER JOIN COURSE C ON S.CID=C.CID; **SQL>**SELECT * FROM COURSE C RIGHT OUTER JOIN STUDENT S ONC.CID=S.CID; **FULL OUTER JOIN:**

It is a combination of left outer and right outer join

Retrieving matching and also un-matching rows from both tables at a time. Ex:

SQL>SELECT * FROM STUDENT S FULL OUTER JOIN COURSE C ON S.CID=C.CID; **SQL>**SELECT * FROM COURSE C FULL OUTER JOIN STUDENT S ON C.CID=S.CID; NOTE:

Generally “equi join” is retrieving only matching rows from the multiple tables if we want retrievingmatching and unmatching rows from the multiple tables then we must use “OUTER JOINS” techniques.

12-11-24

NON-EQUI JOIN:

– When we are retrieving data from multiple tables based on any operator except and “=” operator– In this join we will use the following operators are <,>,<=,>=,AND,OR,BETWEEN... Etc.

Demo tables:

SQL>SELECT * FROM TEST11;

SNO NAME

1 SMITH

2 ALLEN

SQL>SELECT * FROM TEST12

SNO SAL

1 23000

2 34000

Ex:

SQL>SELECT * FROM TEST11 T1 JOIN TEST12 T2 ON T1.SNO>T2.SNO;

SQL>SELECT * FROM TEST11 T1 JOIN TEST12 T2 ON T1.SNO>=T2.SNO; **SQL>**SELECT *

FROM TEST11 T1 JOIN TEST12 T2 ON T1.SNO<T2.SNO; **SQL>**SELECT * FROM TEST11 T1

JOIN TEST12 T2 ON T1.SNO>=T2.SNO; **SQL>**SELECT * FROM TEST11 T1 JOIN TEST12 T2

ON T1.SNO!=T2.SNO; Ex:

Waq to display employees whose salary is between low salary and high salary

fromEMP,SALGRADEtable**SQL>**SELECT ENAME,SAL,LOSAL,HISAL FROM EMP JOIN SALGRADE

ONSALBETWEEN

LOSAL AND HISAL;

OR

SQL>SELECT ENAME,SAL,LOSAL,HISAL FROM EMP JOIN SALGRADE ON(SAL>=LOSAL)AND (SAL<= HISAL);

CROSS JOIN:

– Joining two or more then two tables without any condition.
– In cross join mechanism each row of table will join with each row of another table. – For example a table is having (m) no.of rows and another table is having(n) no.of rowsthentheresult in (mXn) rows:

Ex:

SQL>SELECT * FROM STUDENT CROSS JOIN COURSE;

DEMO TABLES:

SQL>SELECT * FROM ITEMS1;

SNO INAME PRICE

1 PIZZA 180

2 BURGER 85

SQL>SELECT * FROM ITEMS2;

SNO INAME PRICE

1 PEPSI 20

2 COCACOLA 25

SQL>SELECT I1.INAME,I1.PRICE,I2.INAME,I2.PRICE,I1.PRICE + I2.PRICE ASTOTAL_AMOUNT FROM ITEMS1 I1 FROM ITEMS1 I1 CROSS JOINITEMS2I2;

NATURAL JOIN :

– It is similar to equi join.
– Retrieving matching rows only
– Natural join condition is preparing by system by default based on an "=" operator – The advantage of natural join is avoiding duplicate columns from the result set.

Ex:

SQL>SELECT * FROM STUDENT S NATURAL JOIN COURSE S; **13-11-24**

SELF JOIN:

– Joining a table by itself is known as "self join".

Or

– Comparing a table data by itself is known as "self join".
– When we use self join we must create alias names otherwise self join cannot beimplemented
– Whenever we are creating alias name on table internally oracle server well preparedvirtualtables on each alias name.
– We can create any no.of alias names on a single table but each alias nameshouldbedifferent.
– Self join can be used at two levels :
  • Level-1: comparing a single column values by itself with in the table. • Level-2: comparing two different columns values to each other with inthetable.

Level-1: comparing a single column values by itself with in the table: Ex:

Waq to display employees who are working in the same location where the employeeSMITHisalso working?

DEMO TABLE

SQL>SELECT * FROM TEST;

ENAME LOC

SMITH HYD

ALLEN PUNE

WARD HYD

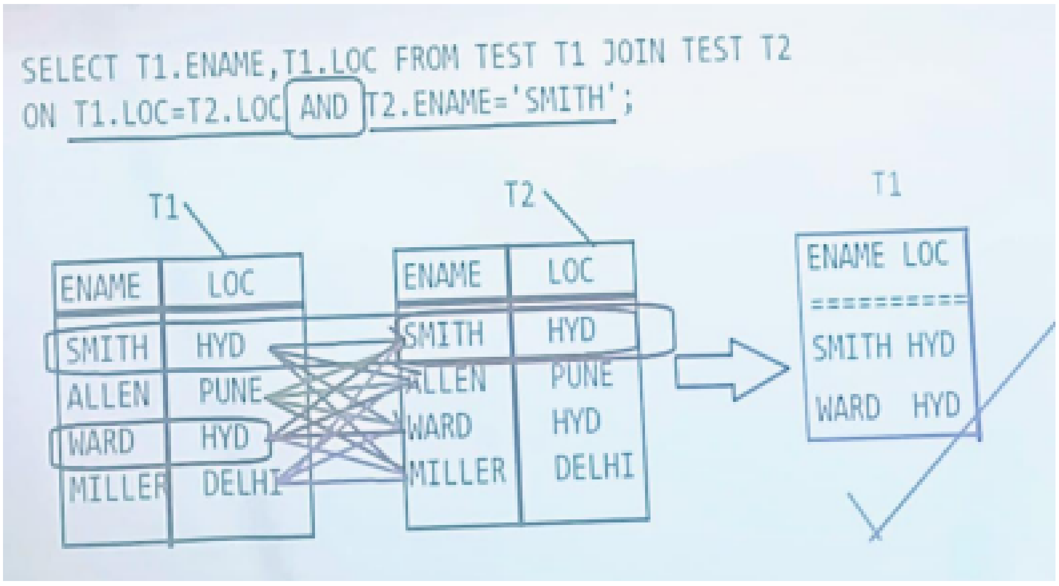
MILLER DELHI

SQL>SELECT T1.ENAME,T1.LOC FROM TEST T1 JOIN TEST T2 ON T1.LOC = T2.LOCANDT2.ENAME =

‘SMITH’;

OUTPUT:

```
ENAME LOC
SMITH HYD
WARD HYD
```



```
SELECT T1.ENAME,T1.LOC FROM TEST T1 JOIN TEST T2
ON T1.LOC=T2.LOC AND T2.ENAME='SMITH';
```

Waq to display employees whose salaries are same as the employee ‘SCOTT’ salary?

SQL> SELECT E.ENAME AS EMPLOYEE ,E.SAL AS EMP_SAL FROMEMP EJOINEMPE1ONE.SAL > E1.SAL WHERE E1.ENAME='SCOTT';

Waq to display managers under their employees from EMP table?

SQL>SELECT M.ENAME AS MANAGER,E.ENAME AS EMPLOYEE FROMEMPMJOINEMPEON E.MGR = M.EMPNO;

Waq to disply employees who are working under the manager “BLAKE”?

SQL>SELECT E.ENAME AS EMPLOYEE_NAME,M.ENAME AS MANAGERFROMEMPMJOINEMP E ON E.MGR = M.EMPNO WHERE M.ENAME = 'BLAKE';

Waq to display manager of the employee ‘BLAKE’?

SQL>SELECT E.ENAME AS EMPLOYEE, M.ENAME AS MANAGER FROMEMPEJOINEMPMON E.MGR = M.EMPNO WHERE E.ENAME = 'BLAKE';

Waq to display employees who are joined before their manager?

SQL> SELECT E.ENAME AS EMPLOYEE,M.ENAME AS MANAGER,E.HIREDATEASEMP_HIREDATE,M.HIREDATE AS MAN_HIREDATE FROM EMP E JOIN EMPMONE.MGR=M.EMPNO WHERE E.HIREDATE < M.HIREDATE;

Waq to display employee whose salary is more then to their manager salary?

SQL> SELECT E.ENAME AS EMPLOYEE,M.ENAME AS MANAGER,E.SAL ASEMP_SAL,M.SALAS MAN_SAL FROM EMP E JOIN EMP M ON E.MGR = M.EMPNO WHERE E.SAL>M.SAL;

HOW TO JOIN MORE THAN TWO TABLES:

Syntax:

SELECT */<LIST OF COLUMNS> FROM <TN1> <JOIN KEY><TN1> ON <JOININGCONDITION1> <JOIN KEY> <TN3> ON<JOIN CONDITION2> <JOIN KEY> <TN4> ON<JOINCONDITION3>;

Ex:

Demo_table

SQL>SELECT * FROM REGISTER;

```
REG.NO REGDATE CID

  1001 15-NOV-24 1

  1002 16-NOV-24 2
```

JOINING THREE TABLES:

SQL>SELECT * FROM STUDENT;

SQL>SELECT * FROM COURSE;

SQL>SELECT * FROM REGISTER;

Example of EQUI JOIN:

SQL>SELECT SID,SNAME,CNAME,REGDATE,CFEE FROM STUDENT S JOINCOURSECON S.CID=C.CID JOIN REGISTER R ON C.CID = R.CID

OUTPUT:
STID SNAME CNAME REGDATE CFEE 1021 SMITH ORACLE 15-NOV-24 2000

18-11-24

CONSTRAINTS:

▪ Constraints are used to enforce unwanted / invalid data from a table ▪ Oracle supports the following six types of constraints those are ▪ Unique
- Not null
- Check
- Primary key
- Foreign key
- Default

▪ Constraints are applied a table in two ways

- Column level
- Table level

Column level:

Constraints can be defined on each individual column wise.

Syntax:

Create table <table name>(<column name> <data type>[size]<constraint

type>,<columnname><data type>[size]<constraint type>………);

TABLE LEVEL:

Constraint can be after all columns definitions i.e the end of the table. Syntax:

Create table <table name>(<column name1><datatype>[size],<column name2><datatype>[size],<constraint type><column name1>,<column name2>,…..;

UNIQUE:

To restricted duplicated values but allowed nulls into a column.

Ex:

<u>Column level:</u>

SQL>SELECT TABLE TEST1(SNO NUMBER(2) UNIQUE,NAME VARCHAR2(20) UNIQUE; TESTING:

SQL>SELECT INTO TEST1 VALUES(1,'A');----ALLOWED
SQL>SELECT INTO TEST1 VALUES(1,'A');----NOT ALLOWED **SQL>**SELECT INTO TEST1

VALUES(NULL,NULL);----ALLOWED TABLE LEVEL:

SQL>CREATE TABLE TEST2(SNO NUMBER(2),NAME

VARCHAR2(10),UNIQUE(SNO,NAME);TESTING:

SQL>INSERT INTO TEST2 VALUES(1,'A');---ALLOWED **SQL>**INSERT INTO TEST2

VALUES(1,'A');-----NOT ALLOWED **SQL>**INSERT INTO TEST2 VALUES(1,'B');-----ALLOWED

SQL>INSERT INTO TEST2 VALUES(NULL,NULL);------ALLOWED **NOT NULL:**

- To restricted nulls but allowed duplicate values into a column.
- It can be defined at column level only

Ex:

<u>Column level:</u>

SQL>CREATE TABLE TEST3(SNO NUMBER(3) NOT NULL,NAME VARCHAR2(10)

NOTNULL;TESTING:

SQL>INSERT INTO TEST3 VALUES(1,'A');-----ALLOWED **SQL>**INSERT INTO TEST3

VALUES(1,'A');----ALLOWED **SQL>**INSERT INTO TEST3 VALUES(NULL,NULL);---NOT ALLOWED

CHECK:

To check the values with user defined condition on a column before accepting values Ex:

Column level:

SQL>CREATE TABLE TEST4 {

REGNO NUMBER(5)UNIQUE NOT NULL,CNAME VARCHAR(2) NOT NULL,ENTRY_FEENUMBER(6,2)NOT NULL CHECK(ENTRY_FEE=500),AGE NUMBER(2) NOT NULL CHECK(AGEBETWEEN 18 AND 30),LOC VARCHAR2(10) NOT NULL CHECK(LOC IN('HYD','MUMBAI','DELHI')

}

TESTING:

SQL>INSERT INTO TEST4

VALUES(10001,'SMITH',450,17,'HYDERABAD')---NOTALLOWED**SQL>**INSERT INTO TEST4

VALUES(10001,'SMITH',500,30,'HYD');-----ALLOWED

TABLE LEVEL:
SQL>SELECT TABLE TEST5(ENAME VARCHAR2(10),SAL NUMBER(8,2),CHCK(ENAME=LOWER(ENAME);

TESTING:

SQL>INSERT INTO TEST5 VALUES('ALLEN',23000);---NOT ALLOWED **SQL>**INSERT INTO TEST5

VALUES('allen',23000);---ALLOWED **19-11-24**

PRIMARY KEY:

- It is a combination of unique and not null constraint.
- By using primary key we can restricted duplicated and nulls from a column ▪ A table is having only one primary key.

Column level:

SQL>CREATE TABLE TEST6(STID NUMBER(4) PRIMARY KEY,SNAME VARCHAR2(10) NOTNULL,REGNO NUMBER(5)PRIMARY KEY;

ERROR at line 2:

ORA-02260 : table can have only one primary key

Solution:

SQL>CREATE TABLE TEST6(STID NUMBER(4) PRIMARY KEY,SNAME VARCHAR2(10) NOTNULL,REGNO NUMBER(5)UNIQUE NOT NULL;

TESTING:

SQL>INSERT INTO TEST6(1021,'SMITH',10001);---ALLOWED **SQL>**INSERT INTO

TEST6(1021,'JONES',10001);---NOT ALLOWED **SQL>**INSERT INTO TEST6(NULL,'JONES',NULL);---NOT

ALLOWED **SQL>**INSERT INTO TEST6(1022,'JONES',10002);---ALLOWED COMPOSITE PRIMARY

KEY:(table level)

- When we apply a primary key constraint on combination of multiple columns are calledas"composite primary key"
- In composite primary key individual column are accepting duplicate values but combinationcolumnsare not accepting duplicate values.

Ex:

SQL>CREATE TABLE TEST7(BCODE NUMBER(4),BNAME

VARCHAR2(10),LOCVARCHAR2(10),PRIMARY KEY(BCODE,BNAME);

TESTING:

SQL>INSERT INTO TEST7 VALUES (1021,'SBI','MADHAPUR');---ALLOWED
SQL>INSERT INTO TEST7 VALUES (1021,'SBI','SRINAGAR')----NOT ALLOWED**SQL>**INSERT

INTO TEST7 VALUES (NULL,NULL,'MADHAPUR')----NOT ALLOWED**SQL>**INSERT INTO TEST7

VALUES (1022,'SBI','SRINAGAR')----ALLOWED

20-11-24

FOREIGN KEY:

To make relationship between tables for taking an identity from one table to another table. **BASIC**

RULES:

1. To maintain at least one commonness in both tables.
2. Commonness columns datatype must be same match.
3. In a relationship one table should contain primary key and another table should containforeignkeyand these two constraints should apply on commonness column only 4. A primary key table is called as "parent table" and a foreign key table is called as "childtable" 5. Foreign key column is allowed the values which was found in primary key column. 6. By default a foreign key is allowed duplicates and nulls.

Syntax:

<common column of child table><datatype>[size] references <parent table>(commoncolumnof parent table)

Ex:

SQL>CREATE TABLE DEPT1(DNO NUMBER(2) PRIMARY KEY,DNAME VARCHAR2(10); --PARENT

SQL>INSERT INTO DEPT1 VALUES(1,'ORACLE');

SQL>INSERT INTO DEPT1 VALUES(2,'JAVA');

SQL>COMMIT;

SQL> CREATE TABLE EMP1 (EID NUMBER(4) PRIMARY KEY;ENAME VARCHAR2(10),DNONUMBER(2) REFERENCES DEPT(DNO);------CHILD

SQL>INSERT INTO EMP1 VALUES (1021,'SMITH',1);

SQL>INSERT INTO EMP1 VALUES (1022,'JONES',1);

SQL>INSERT INTO EMP1 VALUES(1023,'ADAMS',2);

NOTE:

One we established relationship between there are two rules are coming to picture**Rule:1** insertion:

We cannot insert values into a child table those values are not existing in primary key columnof parent of parent table.

Ex:

SQL>INSERT INTO EMP1 VALUES(1025,'SCOTT',3);

ERROR at line 1:

ORA-O2291:integrity constraint (MYDB9AM.SYS_C009399) violated parent key not found

Rule.2: we cannot delete a row from parent table that row is having the corresponding childrowsintablewithout addressing to child.

Ex:

SQL>SELECT FROM DEPT1 WHERE DNO=1;

ERROR:at line 1:

ORA-02292 : integrity constraint (MYDB9AM.SYS_C009399) Violated – child record found **How to address to child table:**

To address to child table there are 2 cascade rules in relationship.

- ON DELETE CASCADE
- ON DELETE SET NULL

<u>ON DELETE CASCADE:</u>

- When we delete a row from a parent table then the corresponding child rows aredeletedfrom child table automatically.

Ex:

SQL>CREATE TABLE DEPT2(DNO NUMBER(2) PRIMARY KEY,DNAME VARCHAR2(10);---PARENT

SQL>INSERT INTO DEPT2 VALUES(1,'ORACLE');

SQL>INSERT INTO DEPT2 VALUES(2,'JAVA');

SQL>COMMIT;

SQL>CREATE TABLE EMP2(EID NUMBER(4) PRIMARY KEY;ENAME VARCHA2(10),DNONUMBER(2) REFERENCE DEPT2(DNO) ON DELETE CASCADE);CHILD

SQL>INSERT INTO EMP2 VALUES(1021,'SMITH',1);

SQL>INSERT INTO EMP2 VALUES(1022,'JONES',2);

SQL>COMMIT;

<u>TESTING:</u>

SQL>DELETE FROM DEPT2 WHERE DNO=1;--DELETED **SQL>**SELECT * FROM DEPT2;

SQL>SELECT * FROM EMP2;

<u>ON DELETE SET NULL:</u>

- When we delete a row from parent table then the corresponding child rows of aforeignkeycolumn are converting into NULL in child table automatically.

Ex:

SQL>CREATE TABLE DEPT3(DNO NUMBER(2) PRIMARY KEY,DNAME VARCHAR2(10);---PARENT

SQL>INSERT INTO DEPT3 VALUES(1,'ORACLE');

SQL>INSERT INTO DEPT3 VALUES(2,'JAVA');

SQL>COMMIT;

SQL>CREATE TABLE EMP3(EID NUMBER(4) PRIMARY KEY;ENAME VARCHA2(10),DNONUMBER(2) REFERENCE DEPT3(DNO) ON DELETE NULL);CHILD

SQL>INSERT INTO EMP3 VALUES(1021,'SMITH',1);

SQL>INSERT INTO EMP3 VALUES(1022,'JONES',2);

SQL>COMMIT;

TESTING:

SQL>DELETE FROM DEPT3 WHERE DNO=1;--DELETED **SQL>**SELECT * FROM

DEPT3;

SQL>SELECT * FROM EMP3;

21-11-24

DATA DICTIONARY:

- When we are installing oracle s/w internally system is creating some pre-defined tablesfor storing the information about database objects such as tables,constraints,views,sequences,indexes,procedures,functions,triggers,…etc
- Data-dictionaries are not allowed DML operations but allowed "SELECT" operations only. Sothatdata dictionaries are also called as "READ ONLY TABLES" in oracle.
- If we want to view all data dictionaries in oracle database then we follow the followingsyntax:

SQL>SELECT * FROM DICT;(dictionary is a main table);

SYSTEM DEFINED CONSTRAINTS NAME:

When we applied constraints on column internally system is creating a unique identificationforeach constraint for identifying a constraint.

Ex:

SQL> CREATE TABLE TEST8(SNO NUMBER(2) PRIMARY KEY, NAME VARCHAR2(10) UNIQUE);
Note: if we want to view constraint name along with column name of a specific table in oracledatabasethen we should use a data-dictionary is "USER_CONS_COLUMNS"

EX:

SQL>DESC USER_CONS_COLUMNS;

```
SQL> SELECT COLUMN_NAME,CONSTRAINT_NAME FROM
USER_CONS_COLUMNSWHERE TABLE_NAME = 'TEST8'; ( here TEST8 is should be capital)
COLUMN_NAME CONSTRAINT_NAME
SNO
SYS_C007462
NAME
SYS_C007463
```

me should be capital.

MYDB9AM

USER_CONS_COLUMN(DATADICTIONARY

Column_name Constraint_name Table_name SNO SYS_C009415 TEST8

SNAME SYS_C0094 TEST8

USER DEFINED CONSTRAINT NAME:

- In place of system defined constraint name we also create user defined constraint name(ID) foridentifying a constraint.

Syntax:

<column name><datatype>[size] constraint<user defined constraint name><constraint type>

```
SQL> CREATE TABLE TEST9 (SNO NUMBER(2) CONSTRAINT SN0_PK PRIMARYKEY,
NAMEVARCHAR2(10) CONSTRAINT NAME_UQ UNIQUE);
SQL> SELECT COLUMN_NAME,CONSTRAINT_NAME FROM
USER_CONS_COLUMNSWHERE TABLE_NAME = 'TEST9';

 COLUMN_NAME CONSTRAINT_NAMESNO SN0_PKNAME NAME_UQ
```

22-11-24

How to add constraints to an existing table:

Syntax:

ALTER TABLE <TN> ADD CONSTRAINT <CONSTRAINT NAME>
<CONSTRAINTTYPE>(<COLUMN NAME

EX:

SQL> CREATE TABLE PARENT(STID NUMBER(4),SNAME VARCHAR(2),SFEENUMBER(6,2));

I) Adding a primary key:

SQL> ALTER TABLE PARENT ADD CONSTRAINT STID_PK PRIMARY KEY(STID); II) adding

Unique, Check constraint:

SQL> ALTER TABLE PARENT ADD CONSTRAINT SNAME_UQ UNIQUE(SNAME); **SQL>**ALTER

TABLE PARENT ADD CONSTARINT SFEE_CHK CHECK(SFEE>=5000);

If we want to view check constraint conditional value of a specific column in the specific tableinoraclethenwe use a data-dictionary IS “USER_CONSTRAINT”

SQL> DESC USER_CONSTRAINTS;

SQL>SELECT CONSTRAINT_NAME,SEARCH_CONDITION FROM USER_CONSTRAINTSWHERE TABLE_NAME='PARENT';

CONSTRAINT_NAME SEARCH_CONDITION SFEE_CHK SFEE>=5000 III) Adding a foreign key references:

Syntax:

ALTER TABLE <TN> ADD CONSTARINT <CONSTRAINT NAME> FOREIGNKEY(COMMON COLUMN OF CHILDN NAME) REFERENCES <PARENT TABLENAME>(COMMONCOLUMN OF PARENT TABLE) ON DELETE CASCADE/ ON DELETE SET NULL;

EX:

SQL> CREATE TABLE CHILD(CNAME VARCHAR2(10),STID NUMBER(4));

SQL> ALTER TABLE CHILD ADD CONSTRAINT STID_FK FOREIGN KEY(STID) REFERENCES PARENT(STID) ON DELETE CASCADE;

How to apply NOT NULL constraint:
Syntax:

ALTER TABLE <TN> MODIFY <COLUMN NAME> CONSTRAINT <CONSTRAINTNAME>NOTNULL;

EX:

SQL> ALTER TABLE PARENT MODIFY SNAME CONSTRAINT SNAME_NNNOTNULL; How to drop

constraint from an existing table:

Syntax:

ALTER TABLE <TN> DROP CONSTRAINT <CONSTRAINT NAME>; Dropping a

Primary key:

Method 1:with relationship:

SQL> ALTER TABLE PARENT DROP CONSTRAINT STID_PK CASCADE; Method 2:

without relationship:

SQL> ALTER TABLE PARENT DROP CONSTRAINT STID_PK;

Iii) Dropping unique,check,not null constraint:

SQL> ALTER TABLE PARENT DROP CONSTRAINT SFEE_CHK;

SQL> ALTER TABLE PARENT DROP CONSTRAINT SNAME_UQ;

SQL> ALTER TABLE PARENT DROP CONSTRAINT SFEE_NN;

How to change a constraint name:

Syntax:

ALTER TABLE <TN> RENAME CONSTRAINT <OLD CONSTRIANT NAME> TO<NEWCONSTRIANT NAME>;

Ex:

SQL> CREATE TABLE TEST10(REGNO NUMBER(5) PRIMARY KEY);

SQL> SELECT COLUMN_NAME,CONSTRAINT_NAME, FROM USER_CONS_COLUMNWHERETABLE_NAME='TEST10';

COLUMN_NAME CONSTRAINT_NAMEREGNO SYS_C007471 **SQL>** ALTER TABLE TEST10 RENAME

CONSTRAINT SYS_C007471 TOREGNO_PK;--->COLUMN_NAME CONSTRAINT_NAMEREGNO

REGNO_PK

DEFAULT constraint:

To assign a user defined default value to a column.

Ex:

SQL> CREATE TABLE TEST12(ENAME VARCHAR2(10), SAL NUMBER(8,2)

DEFAULT15000);**SQL>** INSERT INTO TEST12(ENAME,SAL) VALUES('A',25000);

SQL> INSERT INTO TEST11(ENAME) VALUES('A');

How to add a default value to an existing table column:

Syntax:

ALTER TABLE <TN>MODIFY <COLUMN NAME>DEFAULT<VALUE>; EX:

SQL> CREATE TABLE TEST14(ENAME VARCHAR2(10),LOC VARCHAR2(20)); **SQL>**

ALTER TABLE TEST14 MODIFY LOC DEFAULT 'HYD';

TESTING:

SQL> INSERT INTO TEST14 (ENAME) VALUES('SMITH');

SQL> SELECT * FROM TEST14;

Note:

If we want to view the default constraint value of a column in the table in oracle databasethenuseadatadictionary is “USER_TAB_COLUMNS”.

EX:

SQL>DESC USER_TAB_COLUMNS:

SQL> SELECT COLUMN_NAME, DATA_DEFAULT FROM USER_TAB_COLUMNSWHERETABLE_NAME='TEST14';

<u>COLUMN_NAME</u> <u>DATA_DEFAULT</u> LOC 'HYD' <u>How to remove a default constraint values from a column:</u>

Syntax:

ALTER TABLE <TN> MODIFY <COLUMN NAME> DEFAULT null; EX:

SQL> ALTER TABLE TEST14 MODIFY LOC DEFAULT NULL; <u>COLUMN_NAME</u> <u>DATA_DEFAULT</u>

LOC null

23-11-24

<u>TRANSACTION CONTROL LANGUAGE (TCL):</u>

What is transaction?

To perform some operation over database.

To control these transaction on database then we must use TCL. Commands: ■ Commit

■ Rollback

■ Savepoint

I)COMMIT:

■ To make a transaction is permanent

■ There are two types of commit transactions those are,

<u>Implicit commit:</u>

■ These transactions are committed by system by default. Ex: DDL Commands

<u>Explicit commit:</u>

■ These transactions are committed by user

Ex:DML Commands

Ex:

Syntax:

COMMIT;

Ex:

SQL>CREATE TABLE BRANCJ (BCODE NUMBER(4) BNAME VARCHAR2(10),LOCVARCHAR2(10);

SQL>INSERT INTO BRANCH VALUES(1021,’SBI’,’HYD’);

SQL>COMMIT;

SQL>UPDATE BRANCH SET LOC=’PUNE’ WHERE BCODE = 1021; **SQL>**COMMIT;

SQL>DELETE FROM BRANCH WHERE BCODE = 1021; **SQL>**COMMIT;

OR

SQL>INSERT INTO BRANCH VALUES(1021,’SBI’,’HYD’);
SQL>UPDATE BRANCH SET LOC=’PUNE’ WHERE BCODE = 1021; **SQL>**DELETE

FROM BRANCH WHERE BCODE = 1021;

SQL>COMMIT;

II) **ROLLBACK:**

TO cancel a transaction

Once a transaction is committed then we cannot rollback.

Syntax:

ROLLBACK

Ex:

SQL>CREATE TABLE BRANCH (BCODE NUMBER(4) BNAME
VARCHAR2(10),LOCVARCHAR2(10);

SQL>INSERT INTO BRANCH VALUES(1021,’SBI’,’HYD’);

SQL>ROLLBACK;

SQL>UPDATE BRANCH SET LOC=’PUNE’ WHERE BCODE = 1021; **SQL>**ROLLBACK;

SQL>DELETE FROM BRANCH WHERE BCODE = 1021;

SQL>ROLLBACK;

OR

SQL>INSERT INTO BRANCH VALUES(1021,’SBI’,’HYD’);

SQL>UPDATE BRANCH SET LOC=’PUNE’ WHERE BCODE = 1021; **SQL>**DELETE

FROM BRANCH WHERE BCODE = 1021;

SQL>ROLLBACK;

III. **SAVE POINT:**

When we create a save point internally system is allocating a special memory for storingrequiredrow/rows which we want to cancel (I.e rollback) In the future.

Syntax to create a savepoint:

SAVEPOINT <POINTER NAME>;

Syntax to rollback a savepoint:

ROLLBACK TO <POINTER NAME>;

Ex:

SQL>DELETE FROM BRANCH WHERE BCODE = 1021; **SQL>**DELETE

FROM BRANCH WHERE BCODE = 1025; **SQL>**SAVEPOINT P1;

SQL>DELETE FROM BRANCH WHERE BCODE = 1023; TESTING:

CASE:1:

SQL>ROLLBACK TO P1; (rollback :1023 row only); Case 2:

SQL>COMMIT/ ROLLBACK;

Ex:

SQL>DELETE FROM BRANCH WHERE BCODE = 1021; **SQL>**SAVEPOINT

P1;

SQL>DELETE FROM BRANCH WHERE BCODE = 1025;

TESTING:

CASE:1:

SQL>ROLLBACK TO P1; (rollback :1025 row only); Case 2:

SQL>COMMIT/ ROLLBACK;

25-11-24

SUBQUERY:

a query inside another query is called as subquery or nested query.

Syntax:

SELECT * FROM <TN> WHERE <CONDITION> (SELECT * FROM …..(SELECT* FROM…..));Outer

query / parent query/main query child query TYPES OF SUBQUERIES:

1. Non-corelated subquery
2. Co-related subquery

Non-corelated subquery:

I) Single row subquery

II) Multiple row subquery

III) Multiple column subquery

Single row subquery:

When a subquery return a single value.

In this we will use the following operators are “=,<,>,<=,>=,!=”

Ex:

Waq to display employees details who are getting the 1st highest salary?

Step1:

SQL>SELECT MAX(SAL) FROM EMP;

Step2:

SQL>SELECT * FROM EMP WHERE <inner query return value column name > = (inner query);Step3:subquery statement = (outer query + inner query):

SQL>SELECT * FROM EMP WHERE SAL = (SELECT MAX(SAL) FROMEMP); Waq to display the senior most employee details from emp table? Step1:

SQL>SELECT MIN(HIREDATE) FROM EMP;

Step2:

SQL> SELECT * FROM EMP WHERE HIREDATE = (SELECT MIN(HIREDATE) FROMEMP); Waq to display the second highest salary from emp table?

SQL> SELECT MAX(SAL) FROM EMP WHERE SAL<(SELECT MAX(SAL) FROMEMP);