# 1.ORACLE

## Types of Data

We have two types of data.

1) Unstructured Data

2) Structured Data

## 1) Unstructured Data

* Data which is not in readable format is called unstructured data.
* In general, meaning less data is called unstructured data.
* File contains unstructured data.

ex:

302 Lakemba SYD NSW AUS

## 2) Structured Data

* Data which is in readable format is called structured data.
* In general, meaningfull data is called structured data.
* Databases contains structured data.

ex:

**Unit Locality City State Country**

302 Lakemba SYD NSW AUS

**Oracle**

* Oracle is one of the database which is used to store the structured data.
* It is a product of Oracle Corporation.
* It is classified into two types.

Oracle

|

|-------------------------------------------------|

SQL PL/SQL

(Structured Query Language) (Procedural/Structured Query Language)

**Client-Server Architecture**

In this architecture we will see how our data will store from frontend to backend.

**Diagram: oracle1.1**



**FrontEnd**

The one which is visible to the enduser to perform some operations is called frontend.

ex:

JAVA, .Net, Python, Perl and etc.

## Communication Channel(CC)

It acts like a bridge between frontend and backend.

ex:

JDBC - Java Database Connectivity

ODBC - Open Database Connectivity

PDBC - Python Database Connectivity

**BackEnd**

The one which is not visible to the enduser but it performs operations based on the instructions given by frontend is called backend.

## Management System

Management system is a software which is used to manage the database.

Using management system we can perform following acitivities very easily.

1) Adding the new data

2) Modifying the existing data

3) Selecting the required data

4) Deleting the unnecessary data

**DBMS**

A database along with software which is used to manage the database is called database management system.

**RDBMS**

If a database is designed based on relational theories is called relational database management system.

ex:

Oracle, MySQL , SQL Server, PostgreSQL, Sybase, TeraData and etc.

**SQL**

* SQL stands for Structured Query Language which is pronounce as SEQUEL.
* This language is used to interact with Oracle Database.
* It is a case insensitive language.
* It is a command based language.
* Every command must starts with verb.

ex:

select,update,delete,insert,merge and etc.

* Every command must ends with semicolon(;).
* It is developed by Mr.Codd in 1972 (By IBM).

# 2.Sub Languages of SQL

We have five sub languages of SQL.

1) DDL (Data Definition Language)

2) DML (Data Manipulation Language)

3) DRL/DQL (Data Retrieve/Query Language)

4) TCL (Transaction Control Language)

5) DCL (Data Control Language)

## 1) DDL (Data Definition Language)

This language is used to maintain the objects in database.

It is a combination of five commands.

ex:

create,alter,drop,truncate and rename.

## 2) DML (Data Manipulation Language)

This language is used to manipulate the data present in database.

It is a collection of four commands.

ex:

insert,update,delete and merge.

## 3) DRL/DQL (Data Retrieve/Query Language)

This language is used to retrieve the data present in database.

It is a collection of one command

ex:

select

## 4) TCL (Transaction Control Language)

This language is used to maintain the transaction in a database.

It is a collection of three commands.

ex:

commit, rollback and savepoint.

## 5) DCL (Data Control Language)

This language is used to control the access of data to the user.

It is a collection of two commands.

ex:

grant and revoke

# 3.Table

* Table is an object which is used to represent the data.
* A table is a collection of rows and columns.
* Oracle is a case insensitive language but the data present in table is a case sensitive.

ex:

--------------------------------

NO | NAME | ADD

--------------------------------

101| Alan | Texas

--------------------------------

102| Jose | Vegas

--------------------------------

103| Linda | Florida

--------------------------------

Here we have 3 rows and 3 columns.

**Oracle**

Version : 10g or 11g

Vendor : Oracle Corporation

Category : Express Edition Software

Port No : 1521

Username : system (default)

Password : admin

Website : www.oracle.com/in/database

Download link :

<https://drive.google.com/file/d/0B9rC21sL6v0td1NDZXpkUy1oMm8/view?usp=drive_link&resourcekey=0-aKooR3NmAh_eLo_qGw_inA>

**Oracle 21c Download link**

https://drive.google.com/file/d/1tBx4vH14\_cF9XClgnOF6L\_1kdODbHSM5/view?usp=drive\_link

To perform any operation on database we need to establish the connection.

Once the work with database is completed we need to disconnect with database.

**Various ways to establish the connection with database**

1)

SQL>connect

username : system

password : admin

SQL>disconnect

2)

SQL>conn

username : system

password : admin

SQL>disc

3)

SQL> conn system/admin

SQL> disc

## create command

A create command is used to create the tables in a database.

**syntax:**

create table <table\_name>(col1 datatype(size), col2 datatype(size),........,

colN datatype(size));

**ex:**

create table student(sno number(3),sname varchar2(10),sadd varchar2(12));

create table dept(deptno number(3),dname varchar2(10),dloc varchar2(10));

create table emp(eid number(3), ename varchar2(10), esal number(10,2),

deptno number(3), job varchar2(10), comm number(6));

## Describe command

A describe command is used to see the structure of the table.

syntax:

desc <table\_name>;

ex:

desc student;

desc emp;

desc dept;

## Insert command

A insert command is used to insert the record/row in a table.

syntax:

insert into <table\_name> values(value1,value2,....,valueN);

**approach1**

ex:

insert into student values(101,'raja','hyd');

insert into student values('ravi',102,'delhi'); //invalid

insert into student values(102,'ravi'); //invalid

**null**

A null is a keyword which is used to represent undefined or unavailable.

ex:

insert into student values(102,'ravi',null);

**approach2**

insert into student(sno,sname,sadd) values(103,'ramana','vizag');

insert into student(sno,sname) values(104,'ramulu');

**approach3**

Using '&' symbol we can insert inputs at runtime.

ex:

insert into student values(&sno,'&sname','&sadd');

## commit command

It is used to make the changes permanent to database.

syntax:

commit;

**dept table**

create table dept(deptno number(3),dname varchar2(10),dloc varchar2(10));

insert into dept values(10,'CSE','HYD');

insert into dept values(20,'EEE','DELHI');

insert into dept values(30,'ECE','VIZAG');

insert into dept values(40,'MEC','PUNE');

commit;

**emp table**

create table emp(eid number(3), ename varchar2(10), esal number(10,2),

deptno number(3), job varchar2(10), comm number(6));

insert into emp values(201,'Alan',9000,10,'Clerk',null);

insert into emp values(202,'Jose',19000,10,'Clerk',200);

insert into emp values(203,'Lisa',49000,20,'HR',500);

insert into emp values(204,'Kelvin',29000,20,'HR',900);

insert into emp values(205,'Lara',27000,30,'Manager',500);

insert into emp values(206,'James',42000,30,'Manager',600);

commit;

## select command

A select command is used to select the records from database table.

syntax:

select \* from <table\_name>;

Here '\*' means all rows and all columns.

ex:

select \* from student;

select \* from dept;

select \* from emp;

## Projection

Selecting specific columns from database table is called projection.

ex:

select sno,sname,sadd from student;

select sno from student;

In select command we can perform arithmetic operations.

ex:

select sno+100,sname,sadd from student;

select sno-100,sname,sadd from student;

**column alias**

* A userdefined heading given to a column is called column alias.
* Column alias is temperory.
* Once the query is executed we will loss the column alias.

ex:

select sno,sname,sadd from student;

select sno as ROLL\_NO ,sname as NAME,sadd as CITY from student;

select sno-100 as SNO ,sname,sadd from student;

### Interview Queries

**Q) Write a query to display all employees information from employee table?**

select \* from emp;

**Q) Write a query to display employee id, employee name and employee salary from employee table?**

select eid,ename,esal from emp;

**Q) Write a query to display list of tables present in database?**

select \* from tab;

**Q) Write a query to see the logical database name?**

select \* from global\_name; //XE

**Q) Write a query to display employee id, employee name, employee salary and annual salary from employee table?**

select eid,ename,esal,esal\*12 from emp;

**Q) Write a query to display employee id, employee name, employee salary and annual salary as ANNUAL\_SAL from employee table?**

select eid,ename,esal,esal\*12 as ANNUAL\_SAL from emp;

## Where clause

It is used to read specific rows from database table.

syntax:

select \* from <table\_name> where condition;

ex:

select \* from student where sno=101;

select \* from student where sname='ramulu';

select \* from student where sadd='HYD'; // no rows selected

select \* from student where sadd='hyd';

### Interview Queries

**Q) Write a query to display employee information those who are working in 10 department?**

select \* from emp where deptno=10;

**Q) Write a query to display employee information those who are working as a Manager?**

select \* from emp where job='Manager';

**Q) Write a query to display all employees information those who have salary greater then 25000?**

select \* from emp where esal>25000;

**Q) Write a query to display student information who is living in hyderabad?**

select \* from student where sadd='hyd';

### Interview Query

Q) Write a query to display employee information whose comm is null?

select \* from emp where comm is null;

## Update command

It is used to update the data which is present in database.

**syntax:**

update <table\_name> set <col\_name>=value where condition;

ex:

update student set sname='rani' where sno=101;

update student set sadd='delhi' where sadd is null;

update student set sname='gogo',sadd='pune' where sno=104;

**Note:**

If we won't declare where clause then all rows will be updated.

ex:

update student set sname='raja';

## delete command

It is used to delete the records from database table.

**syntax:**

delete from <table\_name> where condition;

ex:

delete from student where sno=104;

delete from student where sname='raja';

delete from student where sadd='vizag';

**Note:**

If we won't declare where clause then all rows will be deleted.

ex:

delete from student;

delete from emp;

delete from dept;

### Interview Queries

**Q) Write a query to promote all employees from Clerk to Salesman?**

update emp set job='Salesman' where job='Clerk';

**Q) Write a query to increment salary by 1000 whose employee id is 201?**

update emp set esal=esal+1000 where eid=201;

**Q) Write a query to terminate all the employees whose salary is greater then 35000?**

delete from emp where esal>35000;

# 4.Logical Operators

If we want to declare more then one condition in where clause then we need to use logical operators.

We have three types of logical operators.

1) AND

2) OR

3) NOT

## 1) AND

It will return the records if both conditions are true.

Here conditions must be from same row only.

ex:

**Q) Write a query to display employee information whose employee id is 201 and employee name is Alan?**

select \* from emp where eid=201 AND ename='Alan';

**Q) Write a query to display employee information whose employee id is 201 and employee name is Kelvin?**

select \* from emp where eid=201 AND ename='Kelvin'; //no rows selected

**Q) Write a query to display employees information whose salary is greater then 30000 and less then 45000?**

select \* from emp where esal>30000 AND esal<45000;

## 2) OR

It will return the records if any condition is true

Here conditions can be from any row.

ex:

**Q)Write a query to display employees information whose employee id is 201,202,203?**

select \* from emp where eid=201 OR eid=202 OR eid=203;

**Q)Write a query to delete employees information whose employee id is 201,202,203?**

delete from emp where eid=201 OR eid=202 OR eid=203;

**Q) Write a query to display employees information those who are working as a**

**Clerk, HR and Manager?**

select \* from emp where job='Clerk' OR job='HR' OR job='Manager';

## 3) NOT

It will return the records except the condition.

Here '<>' symbol denoted as NOT operator.

ex:

select \* from emp where NOT eid=201;

select \* from emp where eid<>201;

select \* from emp where NOT ename='Kelvin';

select \* from emp where ename<>'Kelvin';

## IN operator

IN operator is a replacement of OR operator.

It will return the records those who are matching in the list.

ex:

select \* from emp where eid IN (201,202,203);

select \* from emp where ename IN ('Alan','Kelvin','Maria');

# 5.Between operator

* Between operator will return the records those who are in the range of values.
* Between operator takes the support of AND operator.
* In between operator first we need to declare lower limit then higher limit.

ex:

select \* from emp where eid between 201 and 206;

select \* from emp where esal between 10000 AND 30000;

select \* from emp where deptno between 10 AND 30;

## Pattern Matching operators

* Pattern matching operators are used to select the letters from database table.
* Pattern matching operators will take the support of like keyword.
* We have two types of pattern matching operators.

1) Percentage(%)

2) Underscore(\_)

## 1) Percentage(%)

**Q) Write a query to display employees information whose employee name starts with 'L' letter?**

select \* from emp where ename like 'L%';

**Q) Write a query to display employees information whose employee name ends with 'n' letter?**

select \* from emp where ename like '%n';

**Q) Write a query to display employees information whose employee name having middle letter as 'l'?**

select \* from emp where ename like '%l%';

## 2) Underscore(\_)

**Q) Write a query to display employees information whose employee name having second letter as 'a' letter?**

select \* from emp where ename like '\_a%';

**Q) Write a query to display employees information whose employee name having second last letter as 'a' letter?**

select \* from emp where ename like '%a\_';

**Q) Write a query to display employees information whose employee name having third letter as 'l' letter?**

select \* from emp where ename like '\_\_l%';

**Duplicate table or Copy of a table**

Using create and select command we can create duplicate or copy of a table.

ex:

create table stud as select \* from emp;

create table stud as select \* from emp where deptno=10;

create table stud as select eid,ename,esal from emp;

create table stud as select \* from where eid=201 AND ename='Alan';

create table stud as select \* from emp where eid IN (201,202,203);

create table stud as select \* from emp where deptno <> 10;

create table stud as select \* from emp where comm is null;

create table stud as select \* from emp where eid between 201 AND 206;

create table stud as select \* from emp where ename like 'A%';

**cl scr**

It is used to clear the output screen of SQL command prompt.

ex:

cl scr

# 6.DDL commands

1) create - (tables)

2) alter - (columns)

3) drop - (tables)

4) truncate- (rows/records)

5) rename - (tables)

## alter command

Using alter command we can perform following acitivities very easily.

i) Adding new columns.

ii) Modifying existing columns.

iii) Dropping the columns.

iv) Renaming the columns.

i) Adding new columns

Using alter command we can add new columns.

**syntax:**

alter table <table\_name> ADD (col\_name datatype(size));

**ex:**

alter table student ADD (state varchar2(10));

alter table student ADD (pincode number(8));

alter table student ADD (state varchar2(10),pincode number(8));

update student set state='Telangana' where sno=101;

### ii) Modifying existing columns

Using alter command we can modify the column.

We can increase or decrease the size of a column only when existing values are fit into new size.

**syntax:**

alter table <table\_name> modify (col datatype(size));

ex:

desc student;

alter table student modify(state varchar2(15));

desc student;

We can change the datatype of a column only if that column is empty.

ex:

desc student;

alter table student modify(pincode varchar2(8));

desc student;

### iii) Dropping the columns

Using alter command we can drop the columns.

**syntax:**

alter table <table\_name> DROP (col\_name);

ex:

alter table student DROP (pincode,state);

### iv) Renaming the columns

Using alter command we can rename the column.

**syntax:**

alter table <table\_name> RENAME column <old\_name> to <new\_name>;

ex:

alter table student rename column sadd to city;

alter table student rename column sno to rollno;

alter table emp rename column esal to dailywages;

alter table emp rename column job to designation;

## drop command

It is used to drop the tables from database.

syntax:

drop table student;

drop table emp;

drop table dept;

## truncate command

Truncate command is used to delete the records permanently from database.

syntax:

truncate table <table\_name>;

ex:

truncate table student;

truncate table dept;

truncate table emp;

**Q) What is the difference between delete and truncate command?**

**delete** **truncate**

It is a DML command. It is a DDL command.

It deletes the data temperory. It deletes the data permanently.

We can rollback the data. We can't rollback the data.

Where clause can be used. Where clause can't be used.

## rename command

It is used to rename the table name.

syntax:

rename <old\_name> to <new\_name>;

ex:

rename emp to employees;

rename dept to department;

rename student to students;

# 7.Functions

Functions are used to manipulate data items and give the result.

We have two types of functions.

1) Group Functions / Multiple Rows Functions

2) Scalar Functions / Single Row Functions

## 1) Group Functions

Group functions are applicable for multiple rows.

We have following list of group of functions.

ex:

sum(), avg(), max(), min(), count(\*) and count(expression).

**Q) Write a query to display sum of salary of each employee?**

select sum(esal) from emp;

**Q) Write a query to display average salary of each employee?**

select avg(esal) from emp;

**Q) Write a query to display highest salary from emp table?**

select max(esal) from emp;

**Q) Write a query to display least salary from emp table?**

select min(esal) from emp;

**Q) Write a query to display number of records present in database table?**

select count(\*) from emp;

**Q) What is the difference between count(\*) and count(Expression) ?**

count(\*)

It will count number of records present in a database table.

It will add null records.

ex:

select count(\*) from student;

**count(Expression)**

It will return number of values present in a database table column.

It will not add null values.

ex:

select count(eid) from emp;

select count(comm) from emp;

**userlist table**

drop table userlist;

create table userlist(uname varchar2(10), pwd varchar2(10));

insert into userlist values('raja','rani');

insert into userlist values('king','kingdom');

commit;

Q) Write a query to check given credentials are valid or not?

select count(\*) from userlist where uname='raja' and pwd='rani'; // 1

or

select count(\*) from userlist where uname='raja' and pwd='rani2'; //0

### Dual table

Dual table is a dummy table which contains one row and one column.

It is used to perform arithmetic operations and to see the current system date.

ex:

select 10+20 from dual;

select 10\*20 from dual;

select sysdate from dual;

select current\_date from dual;

## 2) Scalar Function

Scalar functions are applicable for single row.

We have following list of sclar functions.

i) Character functions

ii) Number functions

iii) Date functions

iv) Conversion functions

### i) Character functions

**upper()**

It will convert lowercase to uppercase.

ex:

select upper('oracle training') from dual;

**lower()**

It will convert uppercase to lowercase.

ex:

select lower('ORACLE TRAINING') from dual;

**initcap()**

It will display initial letter capital.

ex:

select initcap('oracle training') from dual;

**lpad()**

It will add the character towards left side.

ex:

select lpad('oracle',10,'z') from dual; // zzzzoracle

**rpad()**

It will add the character towards right side.

ex:

select rpad('oracle',10,'z') from dual;// oraclezzzz

**ltrim()**

It will trim the characters towards left side.

ex:

select ltrim('zzoraclezz','z') from dual; // oraclezz

**rtrim()**

It will trim the characters towards right side.

ex:

select ltrim('zzoraclezz','z') from dual; // zzoracle

**trim()**

It wil trim the characters from both the sides.

ex:

select trim('z' from 'zzoraclezz') from dual;

**concat()**

IT will concatinate two strings.

ex:

select concat('ihub','talent') from dual;

**Q) Write a java program to merge two arrays and display them in sorting order?**

input:

arr1 = 4 1 3 2 5

arr2 = 6 9 10 7 8

output:

1 2 3 4 5 6 7 8 9 10

**ex:**

import java.util.Arrays;

class Test

{

public static void main(String[] args)

{

int[] arr1 ={4,1,3,2,5};

int[] arr2 ={6,9,10,7,8};

int size1=arr1.length;

int size2=arr2.length;

arr1=Arrays.copyOf(arr1,size1+size2);

int j=0;

for(int i=size1;i<arr1.length;i++)

{

arr1[i]=arr2[j++];

}

Arrays.sort(arr1);

//display elements

for(int i:arr1)

{

System.out.print(i+" ");

}

}

}

### ii) Number functions

**abs()**

It will return absoluate value.

ex:

select abs(-20) from dual; // 20

select abs(45) from dual; // 45

**sqrt()**

It will return square root value.

ex:

select sqrt(25) from dual; // 5

select sqrt(26) from dual; // 5.09

**power(A,B)**

It will return power value.

ex:

select power(2,5) from dual; // 2\*2\*2\*2\*2 =32

select power(5,3) from dual; //125

**ceil()**

It will return ceil value.

ex:

select ceil(10.5) from dual; // 11

select ceil(10.2) from dual; // 11

**floor()**

It will return floor value.

ex:

select floor(40.6) from dual; // 40

select floor(34.2) from dual; // 34

**round()**

It will return nearest value.

ex:

select round(10.6) from dual; // 11

select round(10.5) from dual; // 11

select round(10.4) from dual; // 10

**trunc()**

It will remove decimals.

ex:

select trunc(10.56) from dual; //10

select trunc(-19.67) from dual; // -19

**greatest()**

It will return highest value.

ex:

select greatest(101,102,103) from dual; //103

**least()**

It will return lowest value.

ex:

select least(101,102,103) from dual; //101

### Working with Date values

Every database software supports different date patterns.

ex:

oracle --> dd-MMM-yy

MySQL --> yyyy-MM-dd

and etc.

**emp1 table**

drop table emp1;

create table emp1(eid number(3),ename varchar2(10),edoj date);

insert into emp1 values(501,'Alan','01-JAN-24');

insert into emp1 values(502,'Nelson',sysdate);

insert into emp1 values(503,'Kelvin',current\_date);

commit;

### iii) Date functions

**ADD\_MONTHS()**

It is used to add the months in a given date.

ex:

select ADD\_MONTHS('01-JAN-24',5) from dual;

**MONTHS\_BETWEEN()**

It will return number of months in a given two dates.

ex:

select MONTHS\_BETWEEN('01-JAN-24','01-MAY-24') from dual;

select ABS(MONTHS\_BETWEEN('01-JAN-24','01-MAY-24')) from dual;

select ABS(MONTHS\_BETWEEN('01-JAN-24','15-MAY-24')) from dual;

**NEXT\_DAY()**

It will return next date of a given day in a week.

ex:

select NEXT\_DAY('23-MAY-24','sunday') from dual;

select NEXT\_DAY('23-MAY-24','thursday') from dual;

select NEXT\_DAY('01-JAN-24','monday') from dual;

**LAST\_DAY()**

It will return last date of a month.

ex:

select LAST\_DAY('21-JAN-24') from dual;

select LAST\_DAY('19-FEB-24') from dual;

### iv) Conversion functions

It is used to convert from one type to another type.

ex:

TO\_CHAR()

We have two pseudo of to\_char() function.

**1) number to\_char()**

It will accept '9' in digits , Euros and dollar symbol.

ex:

select eid,ename,esal from emp;

select eid,ename,TO\_CHAR(esal,'9,999') from emp;

select eid,ename,TO\_CHAR(esal,'99,999') from emp;

select eid,ename,TO\_CHAR(esal,'$99,999') from emp;

select eid,ename,TO\_CHAR(esal,'$99,999') as ESAL from emp;

select eid,upper(ename) as ENAME, TO\_CHAR(esal,'$99,999') as ESAL from emp;

**2) date to\_char()**

ex:

select TO\_CHAR(sysdate,'dd-MM-yyyy') from dual;

select TO\_CHAR(sysdate,'yyyy-MM-dd') from dual;

select TO\_CHAR(sysdate,'day') from dual;

select TO\_CHAR(sysdate,'dy') from dual;

select TO\_CHAR(sysdate,'month') from dual;

select TO\_CHAR(sysdate,'mon') from dual;

select TO\_CHAR(sysdate,'year') from dual;

select TO\_CHAR(sysdate,'dd') from dual;

select TO\_CHAR(sysdate,'mm') from dual;

select TO\_CHAR(sysdate,'yyyy') from dual;

select TO\_CHAR(sysdate,'HH:MI:SS') from dual;

select TO\_CHAR(sysdate,'dd-MM-yyyy HH:MI:SS') from dual;

# 8.Group by clause

Group by clause is used to divide the rows into groups so that we can apply group functions.

A column which we used in group by clause , same column name we need to use in select clause.

ex:

**Q) Write a query to display sum of salary of each department?**

select sum(esal),deptno from emp group by deptno;

**Q) Write a query to dispaly highest salary of each job?**

select max(esal),job from emp group by job;

**Q) Write a query to display average salary of each department?**

select avg(esal),deptno from emp group by deptno;

## Having clause

Having clause is used to filter the rows from group by clause.

Having clause must placed after group by clause.

**ex:**

**Q) Write a query to display sum of salary of each department where sum of salary is grater then 30000?**

select sum(esal),deptno from emp group by deptno having sum(esal)>30000;

**Q) Write a query to display minimum salary of each job where minimum salary if greater then 15000?**

select min(esal),job from emp group by job having min(esal)>15000;

## Order by clause

Order by clause is used to arrange the rows in a table.

By default it will arrange all the rows in ascending order.

**ex:**

select \* from emp order by eid;

select \* from emp order by eid desc;

select \* from emp order by ename;

**Note:**

group by clause, having clause and order by clause.

**ex:**

**Q) Write a query to display sum of salary of each department where sum of salary is grater then 30000?**

select sum(esal),deptno from emp group by deptno having sum(esal)>30000;

or

select sum(esal),deptno from emp group by deptno having sum(esal)>30000 order by deptno desc;

# 9.Integrity Constraints

Constraints are the rules which are applied on the tables.

Using contraints we can achieve accuracy and quality of data.

We have five types of constraints.

1) NOT NULL

2) UNIQUE

3) PRIMARY KEY

4) FOREIGN KEY

5) CHECK

Constraints can be created at two levels.

i) Column level

ii) Table level

## 1) NOT NULL

NOT NULL constraint does not accept null values.

NOT NULL constraint can accept duplicate values.

NOT NULL constraint can be created only at column level.

**column level**

drop table student;

create table student(sno number(3) NOT NULL,sname varchar2(10),sadd varchar2(12));

insert into student values(101,'raja','hyd');

insert into student values(101,'ravi','delhi');

insert into student values(null,'ramana','vizag'); //invalid

commit;

**Note:**

NOT NULL constraint can be created for multiple columns also.

ex:

drop table student;

create table student(sno number(3) NOT NULL,

sname varchar2(10) NOT NULL,

sadd varchar2(12) NOT NULL);

insert into student values(101,'jose','usa'); //valid

insert into student values(null,'raja','hyd'); //invalid

insert into student values(102,null,'delhi'); //invalid

insert into student values(103,'ramana',null); //invalid

commit;

## 2) UNIQUE

UNIQUE constraint does not accept duplicate values.

UNIQUE constraint can accept null values.

UNQIUE constraint can be created at column level and table level.

**column level**

drop table student;

create table student(sno number(3) UNIQUE,sname varchar2(10),sadd varchar2(12));

insert into student values(101,'raja','hyd');

insert into student values(null,'ravi','delhi');

insert into student values(101,'ramana','vizag'); //invalid

commit;

**table level**

drop table student;

create table student(sno number(3),sname varchar2(10),sadd varchar2(12),UNIQUE(sno));

insert into student values(101,'raja','hyd');

insert into student values(null,'ravi','delhi');

insert into student values(101,'ramana','vizag'); //invalid

commit;

Note:

UNIQUE constraint can be created for multiple columns.

ex:

drop table student;

create table student(sno number(3),sname varchar2(10),sadd varchar2(12),UNIQUE(sno,sname,sadd));

insert into student values(101,'raja','hyd');

## 3) PRIMARY KEY

Primary key is a combination of NOT NULL and UNIQUE constraint.

Primary key does not allow null values and duplicate values.

A table can have only one primary key.

Primary key can be created at column level and table level.

**column level**

drop table student;

create table student(sno number(3) PRIMARY KEY,sname varchar2(10),sadd varchar2(12));

insert into student values(101,'raja','hyd');

insert into student values(101,'ravi','delhi'); //invalid

insert into student values(null,'ramana','vizag'); //invalid

commit;

**table level**

drop table student;

create table student(sno number(3),sname varchar2(10),sadd varchar2(12),PRIMARY KEY(sno));

insert into student values(101,'raja','hyd');

insert into student values(101,'ravi','delhi'); //invalid

insert into student values(null,'ramana','vizag'); //invalid

commit;

## 4) Foreign Key

Foreign key is used to establish the relationship between two tables.

To establish the relationship a parent table must have primary key or unique key and child table must have foreign key.

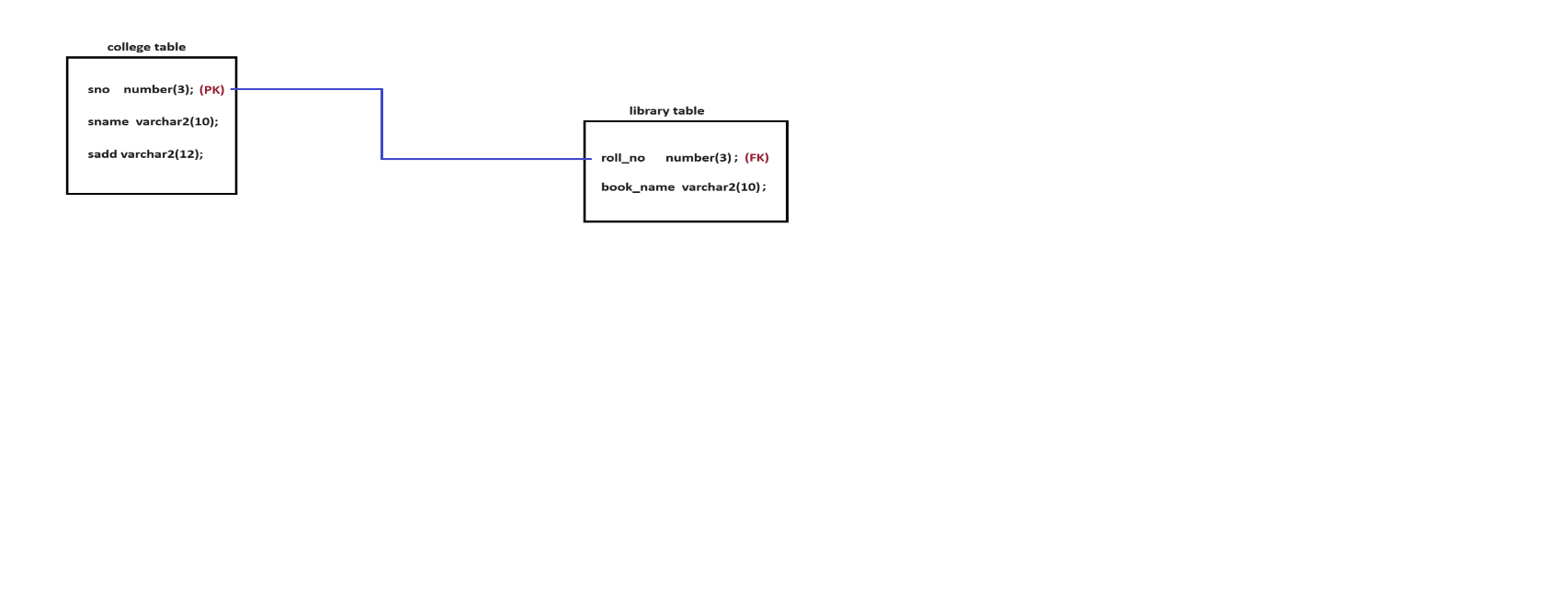
This relationship is also known as master and detailed relationship.

Foreign key will accept only those values which are present in primary key.

Foreign key will accept duplicate and null values.

Foreign key name may or may not match with primary key name but datatype must match.

**Diagram: oracle6.1**

****

**parent table**

drop table college;

create table college(sno number(3) PRIMARY KEY,sname varchar2(10),sadd varchar2(12));

insert into college values(101,'raja','hyd');

insert into college values(102,'ravi','delhi');

insert into college values(103,'ramana','vizag');

commit;

**child table**

drop table library;

create table library(roll\_no number(3) REFERENCES college(sno),book\_name varchar2(10));

insert into library values(101,'java');

insert into library values(102,'oracle');

insert into library values(103,'react');

insert into library values(103,'framework');

insert into library values(null,'angular');

insert into library values(104,'html'); //invalid

commit;

We can't drop parent table directly.First we need to drop child table then parent table.

ex:

drop table library;

drop table college;

## 5) CHECK constraint

It is used to check the domain of a column.

Here domain means what type of a values a column must accept.

Check constraint can be created at column level and table level.

ex:

**column level**

drop table student;

create table student(sno number(3),sname varchar2(10),smarks number(3) check(smarks<=100));

insert into student values(101,'raja',67);

insert into student values(102,'ravi',100);

insert into student values(103,'ramana',150); //invalid

commit;

**column level**

drop table student;

create table student(sno number(3),sname varchar2(10) check(sname=lower(sname)),smarks number(3));

insert into student values(101,'raja',67);

insert into student values(102,'RAVI',100); //invalid

insert into student values(103,'RaMaNa',150); //invalid

commit;

**column level**

drop table student;

create table student(sno number(3),sname varchar2(10) check(sname=upper(sname)),smarks number(3));

insert into student values(101,'raja',67); //invalid

insert into student values(102,'RAVI',100);

insert into student values(103,'RaMaNa',150); //invalid

commit;

**table level**

drop table student;

create table student(sno number(3),sname varchar2(10) ,smarks number(3),

check(sname=upper(sname)));

insert into student values(101,'raja',67); //invalid

insert into student values(102,'RAVI',100);

insert into student values(103,'RaMaNa',150); //invalid

commit;

### Interview Queries

**Q) Write a query to add constraint in a existing table?**

alter table emp ADD primary key(eid);

**Q) Write a query to drop the contraint from table?**

alter table emp DROP primary key;

# 10.TCL Commands

1) commit

2) rollback

3) savepoint

## 1) commit

It is used to make the changes permanent to database.

ex:

drop table student;

create table student(sno number(3),sname varchar2(10),sadd varchar2(12));

insert into student values(101,'raja','hyd');

insert into student values(102,'ravi','delhi');

commit;

select \* from student; // 2 records

## 2) rollback

It is used to undo the changes which are not-permanent.

ex:

drop table student;

create table student(sno number(3),sname varchar2(10),sadd varchar2(12));

insert into student values(101,'raja','hyd');

insert into student values(102,'ravi','delhi');

commit;

insert into student values(103,'ramulu','pune');

insert into student values(104,'rakesh','vizag');

select \* from student; // 4 records

rollback;

select \* from student; // 2 records

## 3) savepoint

It is used to make a logical transaction in a database.

Instead of complete rollback ,we can rollback upto savepoint.

syntax:

savepoint <save\_point\_name>;

ex:

drop table student;

create table student(sno number(3),sname varchar2(10),sadd varchar2(12));

insert into student values(101,'raja','hyd');

insert into student values(102,'ravi','delhi');

savepoint sp1;

insert into student values(103,'ramulu','pune');

insert into student values(104,'rakesh','vizag');

savepoint sp2;

insert into student values(105,'Alan','USA');

insert into student values(106,'Jose','UK');

select \* from student; // 6 records

rollback to sp2;

select \* from student; // 4 records

rollback to sp1;

select \* from student; // 2 records

# 11.Pseudo Columns

Pseudo column means a column which is not real.

We have two pseudo columns.

1) ROWNUM

2) ROWID

## 1) ROWNUM

ROWNUM always starts with 1 and increment by 1.

ROWNUM values are temperory.

Once the query is executed we will loss the ROWNUM values.

ex:

select eid,ename,esal from emp;

select ROWNUM,eid,ename,esal from emp;

## 2)ROWID

ROWID is a memory location where our records will store in a database table.

ROWID is permanent.

ex:

select eid,ename,esal from emp;

select ROWID,eid,ename,esal from emp;

select ROWID,ROWNUM,eid,ename,esal from emp;

### Interview Queries

**Q) Write a query to read first three records from emp table?**

select \* from emp where rownum<=3;

**Q) Write a query to display 4th record from emp table?**

select \* from emp where rownum<=4

minus

select \* from emp where rownum<=3;

# 12.DCL commands

1) Grant

2) Revoke

## Schema

It is a memory location which is used to run SQL commands.

**Privileges**

Permissions given to a user is called privileges.

In general, rights given to a user is called privileges.

We have two types of privileges.

**1) System privilege**

Permission given by DBA to user.

**2) Object privilege**

Permission give by one user to another user.

## 1) Grant

It is used to grant the permissions to the user.

syntax:

grant <privilege1>,<privilege2> to user;

## 2) Revoke

It is used to take back the permissions from the user.

**syntax:**

revoke <privilege1>,<privilege2> from user;

DBA> alter session set "\_ORACLE\_SCRIPT"=true;

DBA> create user swami identified by swami;

DBA> create user renuka identified by renuka;

SWAMI> conn swami/swami (logon denied)

RENUKA> conn renuka/renuka (logon denied)

DBA> grant connect,resource to swami,renuka;

DBA> ALTER USER swami quota 100M on users;

SWAMI> conn swami/swami

RENUKA> conn renuka/renuka

SWAMI>

create table employees(eid number(3),ename varchar2(10),esal number(10));

insert into employees values(501,'Alan',10000);

insert into employees values(502,'Jose',20000);

insert into employees values(503,'Kelvin',30000);

commit;

select \* from employees; //3 records

RENUKA> select \* from employees ; //table or view does not exit

SWAMI> grant select on employees to renuka;

RENUKA> select \* from swami.employees;

RENUKA> delete from swami.employees; (insufficient privileges)

SWAMI> grant update,delete on employees to renuka;

RENUKA> delete from swami.employees;

RENUKA> commit;

SWAMI> select \* from employees;

SWAMI> revoke select,update,delete on employees from renuka;

RENUKA> disc

SWAMI > disc

DBA> revoke connect,resource from swami,renuka;

# 13.Sequence

Sequence is an object which is used to generate the numbers.

syntax:

create sequence <sequence\_name> start with value increment by value.

ex:

create sequence sq1 start with 1 increment by 1;

create sequence sq2 start with 10 increment by 10;

create sequence sq3 start with 201 increment by 1;

We have two types of pseudo's in sequence.

1) NEXTVAL

2) CURRVAL

## 1) NEXTVAL

It is used to generate next number in a sequence.

ex:

create sequence sq1 start with 1 increment by 1;

drop table student;

create table student(sno number(3),sname varchar2(10),sadd varchar2(12));

insert into student values(sq1.NEXTVAL,'raja','hyd');

insert into student values(sq1.NEXTVAL,'ravi','delhi');

insert into student values(sq1.NEXTVAL,'ramana','vizag');

commit;

select \* from student;

## 2) CURRVAL

It will return the last number which is generated by sequence.

ex:

select sq1.CURRVAL from dual;

**Q) Write a query to display list of sequences present in database?**

select sequence\_name from user\_sequences;

**Q) Write a query to drop the sequence?**

drop sequence sq1;

# 14.Synonyms

* Alternate name given to a table is called synonym.
* We can use synonym name instead of table name for all commands.
* Using synonym name length of a table will reduce.

syntax:

create synonym <synonym\_name> for <object\_name>;

ex:

create synonym syn1 for student;

select \* from syn1;

delete from syn1; // 3 records

select \* from student; // no rows selected

Q) Write a query to see the list of synonyms present in database?

select synonym\_name from user\_synonyms;

Q) Write a query to drop the synonym ?

drop synonym syn1;

# 15.Indexes

* Index is an object which is used to improve the performance of select command.
* Index in a database is similar to index in a book.
* We can create index only to those columns which are widely used in where clause.
* When we create index, two columns will be created.One is ROWID and another is indexed column.
* All the records will store in the form of ascending order in the indexed column.

Indexed table

------------------------------

ROWID indexed column

------------------------------

| 9000

| 15000

| 29000

| 37000

| 45000

------------------------------

We have two types of index.

## 1) Simple index

If a index is created only for one column is called simple index.

**syntax:**

create index <index\_name> on <table\_name>(col\_name);

ex:

create index idx1 on emp(esal);

select \* from emp where esal=45000;

Here index is used when we use that column in a where clause.

## 2) Complex index

If index is created for multiple columns is called complex index.

**syntax:**

create index <index\_name> on <table\_name>(col1,col2,..,colN);

ex:

create index idx2 on emp(eid,deptno);

select \* from emp where eid=206 and deptno=30;

Here index is used when we use two columns in a where clause.

**Q) Write a query to display list of indexes present in database?**

select index\_name from user\_indexes;

**Q) Write a query to drop the index?**

drop index idx1;

drop index idx2;

# 16.Joins

select \* from emp; // 6 records

select \* from dept; // 4 records

select \* from emp,dept; // 6\*4= 24 records

select eid,ename,esal,deptno,dname,dloc from emp,dept; // column ambiguously defined

To overcome above limitation we need to use table\_name.column\_name.

ex:

select emp.eid,emp.ename,emp.esal,dept.deptno,dept.dname,dept.dloc from emp,dept;

### table alias

A userdefined heading given to a table is called table alias.

Using table alias length of the query will reduce mean while performance is maintained.

ex:

select e.eid,e.ename,e.esal,d.deptno,d.dname,d.dloc from emp e,dept d; // 6\*4=24 records

**Definition**

Joins are used to retrieve the data from one or more then one table.

We have different types of joins.

1) Equi Join

2) Non-Equi Join

3) Self Join

4) Cartisian product

5) Inner Join

6) Outer Join

## 1) Equi Join

* When two tables are joined based on common column is called equi join.

ex:

select e.eid,e.ename,e.esal,d.dname,d.dloc from emp e,dept d

where(e.deptno=d.deptno);// 6 records

## 2) Non-Equi Join

* When two tables are joined without using equi-join condition is called non-equi join.

ex:

select e.eid,e.ename,e.esal,d.dname,d.dloc from emp e,dept d

where e.esal>30000; // 3 \* 4 = 12 records

## 3) Self Join

* When table joined to itself is called self join.
* In self join we will create two table alias for same table.

**ex:**

select e1.eid,e1.ename,e1.esal,e2.job,e2.comm from emp e1,emp e2

where(e1.deptno=e2.deptno); //6 + 6 = 12 records

## 4) Cartisian product

* It will return all possible combination.
* In cartisian product we will not declare join condition.

ex:

select e.eid,e.ename,e.esal,d.dname,d.dloc from emp e,dept d; // 6 \* 4 = 24 records

## 5) Inner Join

* It is similar to equi join.
* It is given by ANSII people.
* American National Standard for Information Interchange.

ex:

select e.eid,e.ename,e.esal,d.dname,d.dloc from emp e INNER JOIN dept d

ON(e.deptno=d.deptno);// 6 records

## 6) Outer Join

* It is extension of equi join.
* It will return matching as well as not matching records.
* A '+' symbol deonted as outer join operator.

We have following list of outer joins.

### 1) Left outer join

**SQL**

select e.eid,e.ename,e.esal,e.deptno,d.deptno,d.dname,d.dloc from emp e,dept d

where(e.deptno=d.deptno(+));

**ANSII**

select e.eid,e.ename,e.esal,e.deptno,d.deptno,d.dname,d.dloc

from emp e LEFT OUTER JOIN dept d

ON(e.deptno=d.deptno);

### 2) Right outer join

**SQL**

select e.eid,e.ename,e.esal,e.deptno,d.deptno,d.dname,d.dloc from emp e,dept d

where(e.deptno(+)=d.deptno);

**ANSII**

select e.eid,e.ename,e.esal,e.deptno,d.deptno,d.dname,d.dloc

from emp e RIGHT OUTER JOIN dept d

ON(e.deptno=d.deptno);

### 3) Full outer join

**ANSII**

select e.eid,e.ename,e.esal,e.deptno,d.deptno,d.dname,d.dloc

from emp e FULL OUTER JOIN dept d

ON(e.deptno=d.deptno);

# 17.View

* View is a virtual representation of a data from one or more then one table.
* View does not consumes the memory.
* View does not have any data.
* View will get the data when we execute select command.
* A table which is used to create a view is called base table or above table.

syntax:

create view <view\_name> as select stmt;

We have different types of views.

1) Simple View

2) Complex View

3) With read only view

4) With check option view

5) Materialized view

## 1) Simple View

If a view is created by using one base table is called simple view.

ex:

create view v1 as select \* from emp;

create view v1 as select eid,ename,esal from emp;

create view v1 as select \* from emp where deptno=10;

create view v1 as select \* from emp where ename like 'A%';

create view v1 as select \* from emp where eid IN(201,202,203);

create view v1 as select \* from emp where comm is null;

create view v1 as select \* from emp where deptno<>10;

select \* from v1;

Note:

DML operations are allowed in simple view.

ex:

delete from v1 where eid=207;

select \* from v1;

select \* from emp;

## 2) Complex View

If a view is created by using more then one base table is called complex view.

ex:

create view v2 as select e.eid,e.ename,e.esal,d.dname,d.dloc from emp e, dept d

where(e.deptno=d.deptno);

select \* from v2;

**Note:**

DML operations are not allowed in complex view.

ex:

delete from v2; //cannot delete from view

## 3) With read only view

If we want to create view by using one base table and DML operations not required then we need to use with read only view.

ex:

create view v3 as select \* from emp with read only;

select \* from v3;

**Note:**

DML operations are not allowed.

ex:

delete from v3; //cannot perform a DML operation

## 4) With check option view

If we want to create view by using one base table and DML operations are required only if our condition is true then we need to use with check option view.

ex:

create view v4 as select \* from emp where deptno=30 with check option;

select \* from v4;

insert into v4 values(208,'Alan',45000,50,'Salesman',500); // view WITH CHECK OPTION

insert into v4 values(208,'Lisa',45000,30,'Salesman',500);

## 5) Materialized view

To create a materialized view a table must have primary key.

ex:

alter table emp ADD primary key(eid);

create materialized view v5 as select \* from emp;

select \* from v5;

delete from emp where eid=206;

commit;

select \* from emp; // 5 records

select \* from v5; // 6 records

In order to see the changes in a view from base table we need to referesh the materialized view.

ex:

exec DBMS\_SNAPSHOT.REFRESH('v5');

select \* from emp; // 5 records

select \* from v5; // 5 records

**Q) Write a query to see the list of views present in database?**

select view\_name from user\_views;

**Q) Write a query to drop the view?**

drop view v1;

drop view v2;

drop view v3;

drop view v4;

**Q) Write a query to drop the materialized view?**

drop materialized view v5;

# 18.Merge Command

Merge command is a combination of insert and update command.

**student10 table**

drop table student10;

create table student10(sno number(3),sname varchar2(10),sadd varchar2(12));

insert into student10 values(101,'raja','hyd');

insert into student10 values(102,'ravi','delhi');

insert into student10 values(103,'ramana','vizag');

commit;

**student20 table**

drop table student20;

create table student20(sno number(3),sname varchar2(10),sadd varchar2(12));

insert into student20 values(103,'Jose','USA');

insert into student20 values(104,'Mark','UK');

commit;

**ex:**

merge into student10 s1

using student20 s2

on(s1.sno=s2.sno)

when matched

then update set sname=s2.sname,sadd=s2.sadd

when not matched

then insert(sno,sname,sadd) values (s2.sno,s2.sname,s2.sadd);

select \* from student10;

select \* from student20;

# 19.Sub Queries

* If we declare a query inside another query is called sub query.
* Sub queries are used to select the records based on unknown values.
* In sub query, first inner query will execute then outer query.
* A query can be nested upto 32 levels.

We have following list of sub queries.

1) Single Row Subquery

2) Multiple Row Subquery

3) Multiple Column Subquery

## 1) Single Row Subquery

If a sub query returns only one row is called single row subquery.

ex:

**SQL Query**

select \* from emp where eid=203;

**Sub Query**

select \* from emp where eid=(select eid from emp where ename='Nelson');

**ex:**

**SQL Query**

select \* from emp where eid=201 and esal=9000;

**Sub Query**

select \* from emp where

eid=(select eid from emp where ename='Alan') and

esal=(select esal from emp where eid=201);

**Q) Write a query to display second highest salary from emp table?**

select max(esal) from emp where esal<(select max(esal) from emp);

**Q) Write a query to display all employees records whose salary is greater then Lisa salary?**

select \* from emp where esal>(select esal from emp where ename='Lisa');

## 2) Multiple Row Subquery

If a subquery returns more then one row then it is called multiple row sub query.

In multiple row sub query , we need to use multiple row operators.

We have three types of multiple row operators.

1) ANY

2) ALL

3) IN

### 1) ANY

select \* from emp where esal > ANY(select esal from emp where deptno=10);

select \* from emp where esal < ANY(select esal from emp where deptno=10);

### 2) ALL

select \* from emp where esal > ALL(select esal from emp where deptno=10);

### 3) IN

select \* from emp where esal IN (select esal from emp where deptno=10);

## 3) Multiple Column subquery

If subquery returns more then one column then we need to use multiple column subquery.

In multiple column subquery we will use IN operator.

ex:

select \* from emp where(eid,ename,esal) IN (select eid,ename,esal from emp);

select eid,ename,esal from emp where(eid,ename,esal) IN (select eid,ename,esal from emp);

**Assignment**

Q) Write a query to display last three records from emp table?

Q) Write a query to delete duplicate records from emp table?

# 20.PL/SQL

PL/SQL stands for Procedural Language / Structured Query Language.

It is a extension of SQL and it contains following features.

1) We can achieve programming features like control statements ,loops and etc.

2) It reduces network traffic.

3) We can display custom exception/error messages by using the concept of exception handling.

4) We can perform related operations by using the concept of triggers.

5) It compiles and stores PL/SQL program permenent to database for repeated execution.

**PL/SQL BLOCK**

A PL/SQL program is also known as PL/SQL block.

ex:

DECLARE

-

- -- Declaration Section

-

BEGIN

-

- -- Executable Section

-

EXCEPTION

-

- -- Exception Section

-

END;

/

Here '/' is used to submit the PL/SQL program into database.

**Declaration Section**

In declaration section, we will declare variables, exceptions , cursors and etc.

It is optional section.

**Executable Section**

It contains lines of code which is used to complete a table.

Executable section contains actual logic.

It is mandatory section.

**Exception Section**

It contains lines of code whic is executed when exception is raised.

It is optional section.

To see the output in PL/SQL we need to set server output environment.

ex:

SQL> set serveroutput on

**Q) Write a PL/SQL program to display Hello World?**

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Hello World');

END;

/

Here DBMS\_OUTPUT is a package name.

Here PUT\_LINE is a procedure name.

**Q) Write a PL/SQL program to perform sum of two numbers?**

DECLARE

A number;

B number;

C number;

BEGIN

A:=10;

B:=20;

C:=A+B;

DBMS\_OUTPUT.PUT\_LINE(C);

END;

/

**Declaration and Initialization using single line**

Declare

A number:=10;

B number:=20;

C number:=A+B;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('sum of two numbers is ='||C);

END;

/

Using '&' symbol we can read dynamic inputs.

ex:

DECLARE

A number;

B number;

C number;

BEGIN

A:=&a;

B:=&b;

C:=A+B;

DBMS\_OUTPUT.PUT\_LINE('sum of two numbers is ='||C);

END;

/

DML operations are allowed in PL/SQL.

**Q) Write a PL/SQL program to insert a record into student table?**

DECLARE

A number(3);

B varchar2(10);

C varchar2(12);

BEGIN

A:=&sno;

B:='&sname';

C:='&sadd';

insert into student values(A,B,C);

DBMS\_OUTPUT.PUT\_LINE('Record Inserted');

END;

/

**Q) Write a PL/SQL program to update student name based on student number?**

DECLARE

A number(3);

BEGIN

A:=&no;

update student set sname='rani' where sno=A;

DBMS\_OUTPUT.PUT\_LINE('Record Updated');

END;

/

**Q) Write a PL/SQL program to delete a record from student table based on student number?**

DECLARE

A number(3);

BEGIN

A:=&sno;

delete from student where sno=A;

DBMS\_OUTPUT.PUT\_LINE('Record Deleted');

END;

/

DRL operations are allowed in PL/SQL.

To execute select command in PL/SQL we need to use into clause.

**Q) Write a PL/SQL program to display employee name whose employee id is 201?**

DECLARE

L\_Ename varchar2(10);

BEGIN

select ename into L\_Ename from emp where eid=201;

DBMS\_OUTPUT.PUT\_LINE(L\_Ename);

END;

/

**Q) Write a PL/SQL program to display employee name based on employee id?**

DECLARE

L\_Eid number(3);

L\_Ename varchar2(10);

BEGIN

L\_Eid:=&eid;

select ename into L\_Ename from emp where eid=L\_Eid;

DBMS\_OUTPUT.PUT\_LINE(L\_Ename);

END;

/

**Q) Write a PL/SQL block to display employee id , employee name, employee salary whose employee id is 205?**

DECLARE

L\_Eid number(3);

L\_Ename varchar2(10);

L\_Esal number(10,2);

BEGIN

select eid,ename,esal into L\_Eid,L\_Ename,L\_Esal from emp where eid=205;

DBMS\_OUTPUT.PUT\_LINE(L\_Eid||' '||L\_Ename||' '||L\_Esal);

END;

/

## Percentage(%) TYPE Attribute

It is used to declare a local variable with respect to column type.

syntax:

variable\_name table\_name.column\_name%TYPE;

**Q) Write a PL/SQL block to display employee id , employee name, employee salary whose employee id is 205?**

DECLARE

L\_Eid emp.eid%TYPE;

L\_Ename emp.ename%TYPE;

L\_Esal emp.esal%TYPE;

BEGIN

select eid,ename,esal into L\_Eid,L\_Ename,L\_Esal from emp where eid=205;

DBMS\_OUTPUT.PUT\_LINE(L\_Eid||' '||L\_Ename||' '||L\_Esal);

END;

/

## Percentage(%) ROWTYPE attribute

It is used to declare a local variable which holds complete row of a table.

syntax:

variable\_name table\_name%ROWTYPE;

We can't display rowtype variable directly.

We need to use table\_name.column\_name to dispaly the data from rowtype variables.

**Q) Write a PL/SQL program to display employees information whose employee id is 202?**

DECLARE

A emp%ROWTYPE;

BEGIN

select \* into A from emp where eid=202;

DBMS\_OUTPUT.PUT\_LINE(A.eid||' '||A.ename||' '||A.esal||' '||A.deptno||' '||A.job);

END;

/

**Q) Write a PL/SQL program to display employees information based on employee id?**

DECLARE

L\_Eid emp.eid%TYPE;

A emp%ROWTYPE;

BEGIN

L\_Eid:=&id;

select \* into A from emp where eid=L\_Eid;

DBMS\_OUTPUT.PUT\_LINE(A.eid||' '||A.ename||' '||A.esal||' '||A.deptno||' '||A.job);

END;

/

**Assignment**

Q) Write a oracle query to display last three records from a table?

SELECT \* FROM (

SELECT \*

FROM student

ORDER BY sno DESC

)

WHERE ROWNUM <= 3;

To see the output in PL/SQL we need to use below command.

ex:

SQL> set serveroutput on

## Control Statements

### 1) IF THEN

It will evaluate the code only if our condition is true.

ex:

DECLARE

A number:=5000;

BEGIN

IF A>2000 THEN

DBMS\_OUTPUT.PUT\_LINE('It is Greatest');

END IF;

END;

/

**ex:**

DECLARE

A number:=2000;

BEGIN

IF A>5000 THEN

DBMS\_OUTPUT.PUT\_LINE('It is Greatest');

END IF;

END;

/

### 2) IF THEN ELSE

It will evalute the code either our condition is true or false.

**ex:**

DECLARE

A number:=5000;

BEGIN

IF A>2000 THEN

DBMS\_OUTPUT.PUT\_LINE('It is greatest');

ELSE

DBMS\_OUTPUT.PUT\_LINE('It is least');

END IF;

END;

/

ex:

DECLARE

A number:=2000;

BEGIN

IF A>5000 THEN

DBMS\_OUTPUT.PUT\_LINE('It is greatest');

ELSE

DBMS\_OUTPUT.PUT\_LINE('It is least');

END IF;

END;

/

### 3) IF THE ELSIF THEN

It will evalute the code based on multiple conditions.

ex:

DECLARE

opt number;

BEGIN

opt:=&option;

IF opt=100 THEN

DBMS\_OUTPUT.PUT\_LINE('It is police number');

ELSIF opt=103 THEN

DBMS\_OUTPUT.PUT\_LINE('It is enquiry number');

ELSIF opt=108 THEN

DBMS\_OUTPUT.PUT\_LINE('It is emergency number');

ELSE

DBMS\_OUTPUT.PUT\_LINE('Invalid option');

END IF;

END;

/

## LOOPS

We have three types of LOOPS in PL/SQL.

1) Simple LOOP

2) While LOOP

3) For LOOP

### 1) Simple LOOP

It will evaluate the code untill our condition is true.

**ex:**

DECLARE

A number:=1;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Welcome');

LOOP

DBMS\_OUTPUT.PUT\_LINE('Hello');

EXIT WHEN A=4;

A:=A+1;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('Thank You');

END;

/

**2) While LOOP**

It will evaluate the code untill our condition is true.

ex:

DECLARE

A number:=1;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Welcome');

WHILE A<=4 LOOP

DBMS\_OUTPUT.PUT\_LINE('Hello');

A:=A+1;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('Thank You');

END;

/

### 3) For LOOP

It will evaluate the code untill our condition is true.

**Ex:** DECLARE

A number;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Welcome');

FOR A IN 1 .. 4 LOOP

DBMS\_OUTPUT.PUT\_LINE('Hello');

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('Thank You');

END;

/

**ex:**

DECLARE

A number;

BEGIN

FOR A IN 1 .. 10 LOOP

DBMS\_OUTPUT.PUT\_LINE(A);

END LOOP;

END;

/

**ex:**

DECLARE

A number;

N number:=5;

BEGIN

FOR A IN 1 .. 10 LOOP

DBMS\_OUTPUT.PUT\_LINE(N||' \* '||A||' = '||N\*A);

END LOOP;

END;

/

# Exceptions

Runtime errors are called exceptions.

We have two types of exceptions in PL/SQL.

1) Predefined exceptions

2) Userdefined exceptions

## 1) Predefined exceptions

Built-In exceptions are called predefined exceptions.

We have following list of predefined exceptions.

1) NO\_DATA\_FOUND exception

2) TOO\_MANY\_ROWS exception

3) ZERO\_DIVIDE exception

4) VALUE \_ERROR exception

5) DUP\_VAL\_ON\_INDEX exception

6) OTHERS

### 1) NO\_DATA\_FOUND exception

This exception will occur when select statement does not return any row.

ex:

DECLARE

L\_Ename emp.ename%TYPE;

BEGIN

select ename into L\_Ename from emp where eid=209;

DBMS\_OUTPUT.PUT\_LINE(L\_Ename);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Please check employee id ');

END;

/

### 2) TOO\_MANY\_ROWS exception

This exception will occur when select statement returns more then one row.

ex:

DECLARE

L\_Ename emp.ename%TYPE;

BEGIN

select ename into L\_Ename from emp where deptno=10;

DBMS\_OUTPUT.PUT\_LINE(L\_Ename);

EXCEPTION

WHEN TOO\_MANY\_ROWS THEN

DBMS\_OUTPUT.PUT\_LINE('select statement returns multiple rows');

END;

/

### 3) ZERO\_DIVIDE exception

This exception will occur when we divide any number with zero.

ex:

DECLARE

A number;

BEGIN

A:=10/0;

DBMS\_OUTPUT.PUT\_LINE(A);

EXCEPTION

WHEN ZERO\_DIVIDE THEN

DBMS\_OUTPUT.PUT\_LINE('Dont divide by zero');

END;

/

### 4) VALUE \_ERROR exception

This exception will occur when there is a mismatch with datatype or size.

ex:

DECLARE

L\_Esal emp.esal%TYPE;

BEGIN

select ename into L\_Esal from emp where eid=201;

DBMS\_OUTPUT.PUT\_LINE(L\_Esal);

EXCEPTION

WHEN VALUE\_ERROR THEN

DBMS\_OUTPUT.PUT\_LINE('Please check datatype ');

END;

/

**ex:**

DECLARE

A number(3);

BEGIN

A:=12345;

DBMS\_OUTPUT.PUT\_LINE(A);

EXCEPTION

WHEN VALUE\_ERROR THEN

DBMS\_OUTPUT.PUT\_LINE('Please check the size');

END;

/

### 5) DUP\_VAL\_ON\_INDEX exception

This exception will occur when we insert duplicate value in a primary key.

ex:

alter table emp ADD primary key(eid);

BEGIN

insert into emp values(205,'Mark',70000,50,'Salesman',300);

DBMS\_OUTPUT.PUT\_LINE('Record Inserted');

EXCEPTION

WHEN DUP\_VAL\_ON\_INDEX THEN

DBMS\_OUTPUT.PUT\_LINE('Duplicate employee id not allowed');

END;

/

### 6) OTHERS

IT is a univeral angular exception which handles all types of exceptions.

ex:

DECLARE

L\_Ename emp.ename%TYPE;

BEGIN

select ename into L\_Ename from emp where eid=209;

DBMS\_OUTPUT.PUT\_LINE(L\_Ename);

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Please check employee id ');

END;

/

## 2) Userdefined exceptions

Exceptions which are created by the user are called custom/userdefined exceptions.

**steps to develop userdefined exceptions**

**step1:**

Declare the exception

ex:

<exception\_name> EXCEPTION;

**step2:**

Raise the exception

ex:

RAISE <exception\_name>;

**step3:**

Handle the exception

**ex:**

WHEN <exception\_name> THEN

**ex:**

DECLARE

L\_Sal number:=5000;

MY\_EX1 Exception;

BEGIN

IF L\_Sal>2000 THEN

RAISE MY\_EX1;

END IF;

DBMS\_OUTPUT.PUT\_LINE(L\_Sal);

EXCEPTION

WHEN MY\_EX1 THEN

DBMS\_OUTPUT.PUT\_LINE('Salary is too high');

END;

/

To see the output in PL/SQL we need to use below command.

ex:

SQL> set serveroutput on

# Cursors

Cursor is a PL/SQL block which is used to run SQL commands.

We have two types of cursors.

1) Implicit cursor

2) Explicit cursor

## 1) Implicit cursor

All the activities related to cursor like opening the cursor, processing the cursor and closing the cursor which is done automatically is called implicit cursor.

We have four types of implicit cursor attributes.

### i) SQL%ISOPEN

It is a boolean attribute which always returns false.

### ii) SQL%FOUND

It is a boolean attribute which returns true if SQL command is success and returns false is SQL command is failed.

### iii) SQL%NOTFOUND

It is completely reverse of SQL%FOUND.

It is a boolean attribute which returns false if SQL command is success and returns true is SQL command is failed.

### iv) SQL%ROWCOUNT

It will return number of records effected in a database table.

**ex:**

BEGIN

IF SQL%ISOPEN THEN

DBMS\_OUTPUT.PUT\_LINE('Cursor is Open');

ELSE

DBMS\_OUTPUT.PUT\_LINE('Cursor is Closed');

END IF;

END;

/

**ex:**

BEGIN

update student set sname='rani' where sno=105;

IF SQL%FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Record Updated');

ELSE

DBMS\_OUTPUT.PUT\_LINE('Record Not Updated');

END IF;

END;

/

**ex:**

BEGIN

update student set sname='rani' where sno=106;

IF SQL%FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Record Updated');

ELSE

DBMS\_OUTPUT.PUT\_LINE('Record Not Updated');

END IF;

END;

/

**ex:**

BEGIN

update student set sname='lisa' where sno=105;

IF SQL%NOTFOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Record Updated');

ELSE

DBMS\_OUTPUT.PUT\_LINE('Record Not Updated');

END IF;

END;

/

**ex:**

BEGIN

update student set sname='raja';

DBMS\_OUTPUT.PUT\_LINE(SQL%ROWCOUNT||' Records Updated');

END;

/

## 2) Explicit cursor

* All the activities related to cursor like opening the cursor, processing the cursor and closing the cursor which is done by a user is called explicit cursor.
* We need to explicit cursor when select statement returns more then one row.
* We have four types of explicit cursor attributes.

### i) %ISOPEN

It is a boolean attribute which returns true if cursor is open and returns false

if cursor is closed.

### ii) %FOUND

It is a boolean attribute which returns true if SQL command is success and returns false is SQL command is failed.

### iii) %NOTFOUND

It completely reverse of %FOUND.

It is a boolean attribute which returns false if SQL command is success and returns true is SQL command is failed.

### iv) %ROWCOUNT

It will return number of records effecting in a database table.

**steps to work with explicit cursor**

**step1:**

Declare the cursor.

**step2:**

Open the cursor.

**step3:**

Fetch the data from cursor to local variables.

**step4:**

Close the cursor.

**Q) Write a PL/SQL program to display employee names from employee table?**

DECLARE

CURSOR C1 is select ename from emp;

L\_Ename emp.ename%TYPE;

BEGIN

OPEN C1;

LOOP

FETCH C1 into L\_Ename;

EXIT WHEN C1%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE(L\_Ename);

END LOOP;

CLOSE C1;

END;

/

**Q) Write a PL/SQL program to display employee id employee name, employee salary from emp table?**

DECLARE

CURSOR C2 is select eid,ename,esal from emp;

L\_Eid emp.eid%TYPE;

L\_Ename emp.ename%TYPE;

L\_Esal emp.esal%TYPE;

BEGIN

OPEN C2;

LOOP

FETCH C2 into L\_Eid,L\_Ename,L\_Esal;

EXIT WHEN C2%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE(L\_Eid||' '||L\_Ename||' '||L\_Esal);

END LOOP;

CLOSE C2;

END;

/

Q) Write a PL/SQL program to display employees information from emp table?

DECLARE

CURSOR C3 is select \* from emp;

A emp%ROWTYPE;

BEGIN

OPEN C3;

LOOP

FETCH C3 into A;

EXIT WHEN C3%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE(A.eid||' '||A.ename||' '||A.esal||' '||A.deptno||' '||A.job);

END LOOP;

CLOSE C3;

END;

/

# PL/SQL procedures

It is a named PL/SQL block which compiles and executes in a database for repeated execution.

**syntax:**

create or replace procedure <procedure\_name>

is

begin

-

-

-

end;

/

**Q) Write a procedure to display Hello World?**

create or replace procedure p1

is

begin

DBMS\_OUTPUT.PUT\_LINE('Hello World');

END;

/

To execute the procedure we need to use below command.

ex:

exec p1;

PL/SQL procedures may contain three parameters.

1) IN parameter

2) OUT parameter

3) IN OUT parameter

## 1) IN parameter

It takes value from the user.

Q) Write a procedure to perform sum of two numbers?

create or replace procedure sum(A IN number,B IN number)

is

C number;

begin

C:=A+B;

DBMS\_OUTPUT.PUT\_LINE(C);

END;

/

To execute the procedure we need to use below command.

ex:

exec sum(10,20);

## 2) OUT parameter

It returns the value to the user.

Q) Write a PL/SQL procedure to perform sum of two numbers and return sum?

create or replace procedure ret\_sum(A IN number,B IN number,C OUT number)

is

begin

C:=A+B;

END;

/

**steps to call PL/SQL procedure having OUT parameters**

**step1:**

Declare a bind variable

ex:

variable N number;

**step2:**

Execute the procedure

ex:

exec ret\_sum(10,20,:N);

**step3:**

Print the bind variable.

ex:

print N;

## 3) IN OUT parameter

It takes value from the user and return the value to the user.

**Q) Write a PL/SQL procedure to return square of a given number>**

create or replace procedure ret\_square(N IN OUT number)

is

begin

N:=N\*N;

END;

/

**steps to call PL/SQL procedure having IN OUT parameters**

**step1:**

Declare a bind variable.

ex:

Variable N number;

**step2:**

Initialize the bind variable.

ex:

BEGIN

:N:=5;

END;

/

**step3:**

Execute the procedure.

ex:

exec ret\_square(:N);

**step4:**

Print the bind variable.

ex:

print N;

**Q) Write a query to display list of procedures present in database?**

select object\_name from user\_objects where object\_type='PROCEDURE';

**Q) Write a query to see the source code of a procedure?**

select text from user\_source where name='P1';

DML operations are allowed in PL/SQL procedures.

**Q)Write a PL/SQL procedure to insert a record into student table?**

create or replace procedure insert\_record(L\_Sno IN student.sno%TYPE,

L\_Sname INstudent.sname%TYPE,

L\_Sadd IN student.sadd%TYPE)

is

begin

insert into student values(L\_Sno,L\_Sname,L\_Sadd);

DBMS\_OUTPUT.PUT\_LINE('Record Inserted');

END;

/

To execute the procedure we need to use below command.

ex:

exec insert\_record(106,'Alan','USA');

**Q) Write a PL/SQL procedure to update student name based on student id?**

create or replace procedure update\_student(L\_Sno IN student.sno%TYPE)

is

begin

update student set sname='Jack' where sno=L\_Sno;

DBMS\_OUTPUT.PUT\_LINE('Record Updated');

END;

/

To execute the procedure we need to use below command.

ex:

exec update\_student(106);

**Q) Write a PL/SQL procedure to delete student record based on student number?**

create or replace procedure delete\_student(L\_Sno IN number)

is

begin

delete from student where sno=L\_Sno;

DBMS\_OUTPUT.PUT\_LINE('Record Deleted');

END;

/

To see the output in PL/SQL we need to use below command.

ex:

exec delete\_student(106);

**Q) Write a query to drop the procedure?**

drop procedure p1;

drop procedure sum;

drop procedure ret\_sum;

drop procedure ret\_square;

drop procedure update\_student;

drop procedure insert\_record;

drop procedure delete\_student;

### Interview Question

You are a professional robber planning to rob houses along a street. Each house has a certain amount of money stashed, the only constraint stopping you from robbing each of them is that adjacent houses have security systems connected and it will automatically contact the police if two adjacent houses were broken into on the same night.

Given an integer array nums representing the amount of money of each house, return the maximum amount of money you can rob tonight without alerting the police.

**Example 1:**

Input: nums = [1,2,3,1]

Output: 4

Explanation: Rob house 1 (money = 1) and then rob house 3 (money = 3).

Total amount you can rob = 1 + 3 = 4.

**Example 2:**

Input: nums = [2,7,9,3,1]

Output: 12

Explanation: Rob house 1 (money = 2), rob house 3 (money = 9) and rob house 5 (money = 1).

Total amount you can rob = 2 + 9 + 1 = 12.

**ex:**

class Solution

{

public static void main(String[] args)

{

int[] nums={2,7,9,3,1};

Solution s=new Solution();

System.out.println(s.rob(nums));

}

public int rob(int[] nums)

{

int sum=0;

for(int i=0;i<nums.length;i++)

{

if(i%2==0)

{

sum+=nums[i];

}

}

return sum;

}

}

To see the output in PL/SQL we need to use below command.

ex:

SQL> set serveroutput on

# PL/SQL Functions

It is a named PL/SQL block which must and should returns a value.

syntax:

create or replace function <function\_name>

return datatype

is

begin

-

-

-

end;

/

**Q)Write a PL/SQL function to perform sum of two numbers and return sum?**

create or replace function f1(A number,B number)

return number

is

C number;

begin

C:=A+B;

return C;

END;

/

Function can be invoked by using select command.

ex:

select f1(10,20) from dual;

**Q) Write a PL/SQL function to accept one salary and return 10% of TAX deduction?**

create or replace function f2(L\_Esal emp.esal%TYPE)

return number

is

L\_Tax number;

begin

L\_Tax:=L\_Esal\*10/100;

return L\_Tax;

END;

/

We can invoke a function as follow.

ex:

select f2(10000) from dual;

select eid,ename,esal,f2(esal) as TAX from emp;

Note:

DML operations are not allowed in functions.

### Interview Queries

**Q) Write a query to see the list of functions present in database?**

select object\_name from user\_objects where object\_type='FUNCTION';

**Q) Write a query to see the source code of a function?**

select text from user\_source where name='F1';

**Q) Write a query to drop the function?**

drop function f1;

drop function f2;

Q) What is the difference between procedures and functions?

**procedures**  **functions**

Procedure may or may not returns a value. Function always returns a value.

DML operations are allowed. DML operations are not allowed.

Can't be invoked by using select command. Can be invoked by using select command.

# Packages

A package is a collection of logical related sub programs.

Here logical related sub programs means PL/SQL procedures and PL/SQL functions.

In general, package is a collection of procedures and functions.

Package creation involved in two steps.

1) package specification

It contains declaration of logical related sub programs.

2) package body

It contains definition of logical related sub programs.

**ex:1**

**package specification**

create or replace package pkg1

is

procedure sum(A IN number,B IN number);

end pkg1;

/

**package body**

create or replace package body pkg1

is

procedure sum(A IN number,B IN number)

is

C number;

begin

C:=A+B;

DBMS\_OUTPUT.PUT\_LINE(C);

END;

end pkg1;

/

To execute the procedure we need to use below command.

ex:

exec pkg1.sum(10,20);

**ex:2**

**package specification**

create or replace package pkg2

is

function ret\_sum(A number, B number)

return number;

end pkg2;

/

**package body**

create or replace package body pkg2

is

function ret\_sum(A number, B number)

return number

is

C number;

begin

C:=A+B;

return C;

END;

end pkg2;

/

To see the output of a function we need to use below command.

ex:

select pkg2.ret\_sum(40,60) from dual;

### Interview Queries

Q) Write a query to see the list of packages present in database?

select object\_name from user\_objects where object\_type='PACKAGE';

Q) Write a query to see the source code of a package?

select text from user\_source where name='PKG1';

Q) Write a query to drop the packages?

drop package body pkg1;

drop package pkg1;

drop package body pkg2;

drop package pkg2;

# Triggers

* Trigger is a PL/SQL block that is executed based on events.
* We have three types of trigger events i.e insert, update and delete.
* We have three types of trigger timings i.e before, after and insteadof.

syntax:

create or replace trigger <trigger\_name> <timing> <event> on <object\_name>

begin

-

-

-

END;

/

create or replace trigger trg1 before insert on student

begin

DBMS\_OUTPUT.PUT\_LINE('Thanks! For Inserting Record');

END;

/

insert into student values(102,'ravi','delhi');

Trigger can be created on multiple events also.

ex:

create or replace trigger trg2 after update or insert or delete on student

begin

if inserting then

DBMS\_OUTPUT.PUT\_LINE('Yahoo! Record Inserted');

ELSIF updating then

DBMS\_OUTPUT.PUT\_LINE('Yahoo! Record Updated');

ELSE

DBMS\_OUTPUT.PUT\_LINE('Yahoo! Record Deleted');

END IF;

END;

/

insert into student values(103,'ramana','vizag');

update student set sname='rani' where sno=103;

delete from student where sno=103;

Triggers are classified into two types.

1) Statement level trigger

2) Row level trigger

## 1) Statement level trigger

If a trigger execute only for one time irrespective of number of records effected in a database table is called statement level trigger.

By default every trigger is a statement level trigger.

ex:

create or replace trigger trg3 before delete on student

begin

DBMS\_OUTPUT.PUT\_LINE('Thanks! For Deleting Record');

END;

/

delete from student; // trigger will execute only for 1 time.

## 2) Row level trigger

If a trigger executes irrespective of number of records effected in a database table is called row level trigger.

To create a row level trigger we need to "for each row" clause.

ex:

create or replace trigger trg4 before delete on student FOR EACH ROW

begin

DBMS\_OUTPUT.PUT\_LINE('Thanks! For Deleting Record');

END;

/

delete from student; // trigger will execute for multiple times

### Interview Questions

**Q) Write a query to drop the trigger?**

drop trigger trg1;

drop trigger trg2;

drop trigger trg3;

drop trigger trg4;

Q) Write a query to see the list of triggers present in database?

select object\_name from user\_objects where object\_type='TRIGGER';

Q) Write a query to see the source code of a trigger?

select text from user\_source where name='TRG1';