ORACLE

ORACLE is a database, and also known as RELATIONAL DATA BASE MANAGEMENT SYSTEM (RDBMS)

RDBMS is a software and it is useful store data, manage data and to automate business activities.

Ex of RDBMS:

Oracle

Microsoft SQL SERVER

IBM DB2

Postgres SQL

MYSQL

TERADATA

MANGO DB. . . . etc

DATA:

Data is defined as a collection of raw facts.

Information:

Processed data is known as information.

DATA BASE:

It is a software which stores and manages the collection of information of all objects in the business.

It is a collection of programs and each program is specific for performing a task.

DATABASE MANAGEMENT SYSTEM:

DB which is along with management system services is known as DBMS.

RDBMS:

Collection of information of all related objects with in the business is known as RDBMS.

Author of RDBMS is E.F. CODD and he defined 12 Rules for an RDBMS.

In any RDBMS

Data is stored in the form of Tables.

A Table is a collection of Rows and Columns.

A Row is known as Record (collection of columns)

A Column is known as a Field.

SQL (Structured Query Language)

Definition:

It is a collection of predefined commands, key words, and syntax rules. It is used to communicate with any database. By using this language, we can write English like statements, called QUERIES.

SQL COMMANDS

SQL commands are divided into 5 categories. They are

DDL—Data Definition Language

CREATE ALTER DROP TRUNCATE RENAM**E**

DML—Data Manipulation Language

INSERT UPDATE DELETE

DCL—Data Control Language

GRANT REVOKE

TCL—Transaction Control Language

COMMIT ROLLBACK SAVEPOINT

DRL/DQL—Data Retrieval Language/Data Query Language

SELECT

SQL*PLUS

SQL * PLUS is a client application window and it is an interface between client user and database in the local system. In this window we can write queries and programs.

How to open SQL*PLUS window?

START→Programs→Oracle→SQL PLUS

Features of sql * plus:

Only one query is allowed to execute at a time. We can write queries in any case.

Each query must end with symbol **";".** SQL commands are ANSI standard.

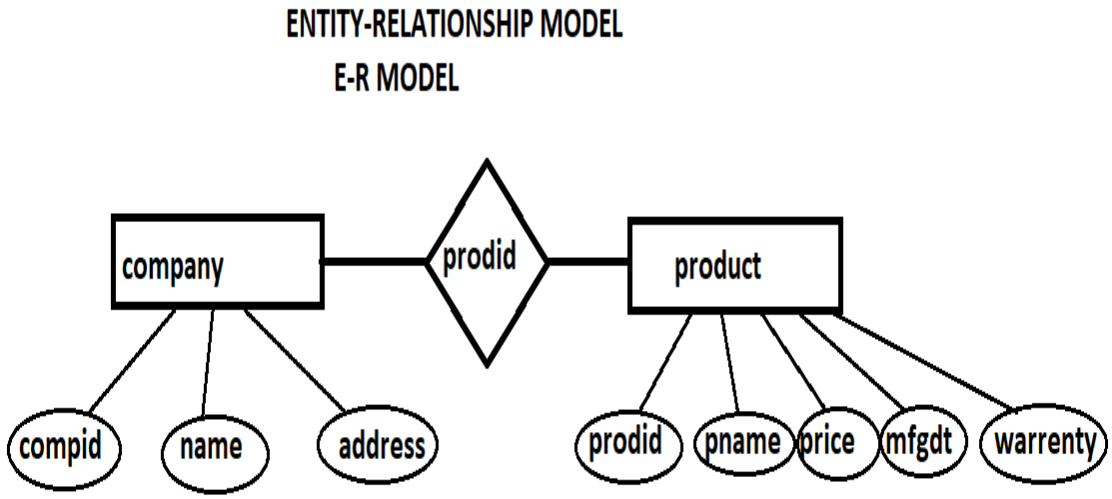
How to connect to Oracle database?

- ➔ Open SQL*PLUS
- ➔ Type as follows

Connect **username/password [Enter]**

DATA MODEL

Data model is a graphical diagram. The data model name is **E-R** (Entity-Relationship) Model. It contains OBJECTS, PROPERTIES and RELATIONS. In E-R model, any object is represented with **RECTANGLE**, each property is represented with **Ellipse** and relation between objects represented with **Rhombus**.



From the above diagram,

Consider Entitiy name as Table Name, Property name as Column name.

DDL command:

CREATE

This command is used to create any data base object like tables, views, indexes, sequences, synonyms, users, functions, procedures, triggers, packages and so on..

HOW TO CREATE A TABLE?

syntax:

CREATE TABLE <table_name>

(

<colname-1> DATATYPE (size),

<colname-2> DATATYPE (size),

);

Ex:

write a query to create a table emps with columns empid, ename, salary, desg, joindt?

Create table emps

(

Empid number(4),

Ename varchar2(20),

Salary number(5),

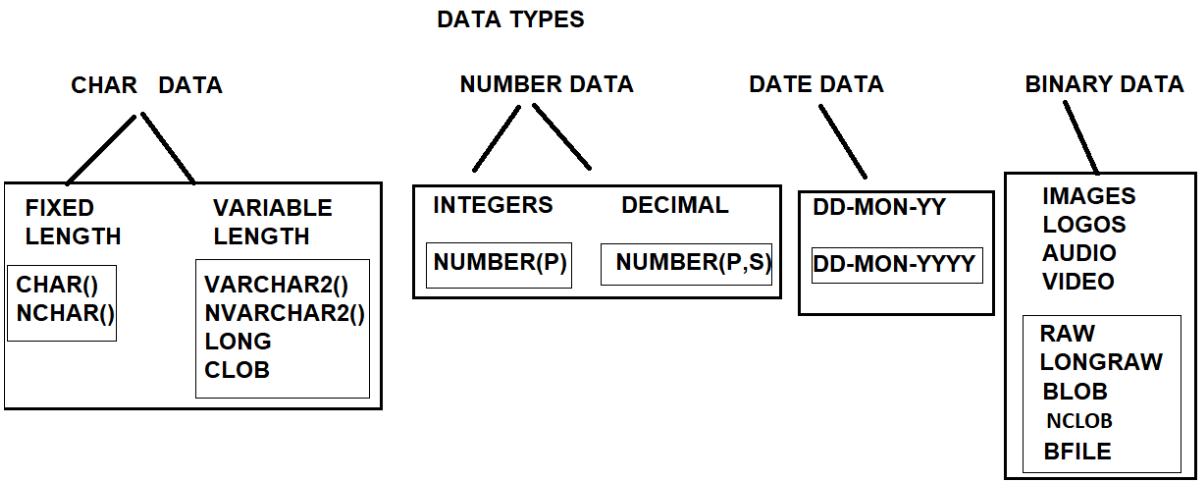
Desg varchar2(20),

Joindt date

);

Oracle Built-In DATATYPES

Oracle is providing, a set of predefined datatype names. Each data type name represent one type of data, that is allowed into specific column. Based on data type physical memory is allocated for each value in the table.



I) Character data

It contains alphabets, digits and symbols.

CHAR (size)

It is useful to store fixed length character values. By default the size is 1 character, and max size is 2000 chars or bytes.

Ex: **empid char(5)**

e0001

e0002

e0003

Ex: **gender CHAR**

M

F

M

F

VARCHAR2(size)

It is useful to store variable length character values. No default size. we must specify size and max size is 4000 chars /bytes.

Ex: **emp_names varchar2(20)**

kiran

ajay

krishna prasad

NCHAR(size)

To store fixed length UNICODE character values. Max. size is 2000 chars/bytes.

NVARCHAR2(size)

To store variable length UNICODE character values. Max size is 4000 bytes or chars.

LONG

It is similar to VARCHAR2() datatype and its maximum size is 2 GB[Giga Bytes].

NOTE:

Only for one column, LONG data type is allowed in a table.

Number Data types

we can store positive or negative numbers with or without decimal point.

i) NUMBER(Precesion)

It is used to store INTEGER values. "Precesion" represents max length of the value.

Ex:

empsal number(6)

20

0

-12780

105000

NUMBER(Precesion, [Scale])

It is useful to store integers as well as decimal values. "Scale" represents max length of decimal part.

Note:

The max value for precision is 38, Max value of scale is 10 power -28

Ex: **prod_price number(7,2)**

123.1

-99999.99

125

-0.75

0

DATE data types

DATE

It is used to store date values. Oracle has predefined date format as follows.

DD-MON-YY

DD - Digits of date

MON - First 3 chars of month name

YY - Last 2 digits of year

Binary Data types

These data types are useful to store binary data like images, thumb impressions, logos, video and audio files RAW(), LONG RAW(), BLOB

RAW(size)

Used to store binary data like images, thumb impressions, logos and so on.

Max size is 2000 bytes < 2 KB.

LONG RAW(size)

It is similar to RAW data type but supports max 2 GB size.

NOTE: Only one long raw column is allowed per a table.

LOB--large objects (data types)

we can store CHARACTER or BINARY or mixed data with max size 4 GB.

LOB datatypes are 3 TYPES

CLOB--Character LOB

It is used to store character value with max size 4 GB

BLOB-- BINARY LOB

It Used to store binary value with max size 4 GB.

NCLOB--Multi char LOB

It is used to store both binary and char data.

BFILE (BINARY FILE)

It is a pointer to the external file, that is saved on the disk. We can read data from external file and write data on to the external file through BFILE.

INSERTING RECORDS INTO A TABLE

INSERT

It is used to insert new records in to a table.

syntax:

INSERT INTO <Tbl> [(col1, col2,. . . coln)] **VALUES**(val1, val2, ... ,val-n);

Character and date type values should be enclosed in single quotes.

Ex:

```
Create table empinfo
(
Eid  number(4),
Ename  varchar2(20),
Sal  number(5),
Jdate  date,
```

Desg varchar2(20),

Gender char

);

insert into empinfo(eid,ename,sal,jdate,desg,gender)

values(1111,'dinesh',75000,'23-may-14','developer','M');

insert into empinfo values(1110,'john',11700,'23-may-14','developer','M');

insert into empinfo values('mahitha',23000,'developer','F');

Error: Not enough values

insert into empinfo (ename,sal,desg,gender) values('mahitha',23000,'developer','F');

insert into empinfo values(null,'smitha',33000,null,'developer','F');

insert into empinfo (ename,sal) values('dinesh','75000');

NULL VALUE:

A missed value is known as null value. Null value is not equal to zero or space or other null values.

HOW TO DISPLAY TABLE STRUCTURE?

A table structure contains column names, data types and sizes.

EX: describe empinfo;

or

desc emp_info;

HOW TO DISPLAY LIST OF TABLE NAMES?

EX: SELECT * FROM TAB;

Q: What is the meaning of tab?

Tab means table space, which contains list of tables created under current schema.

DATA RETRIEVAL LANGUAGE/ DATA QUERY LANGUAGE

SELECT

It will display a copy of requested data from a table or view.

syntax:

SELECT col1, col2,...., coln or * FROM table_name;

NOTE: In above syntax , * represents all columns from table.

ex: Display employee names?

select ename from empinfo;

output:

ENAME

dinesh rao

smitha panday

madhu

Ex: Get emplyee id, names and salaries?

select eno, ename, sal from empinfo;

Ex: display employee details?

select * from emp_info;

Assignments:

i) Create a table Item_dtls (Electronics)

ii) Create a table Sales_dtls

iii) create a table manufacturers

iv) Try to insert at least 10 records in the above tables

v) Try to insert at least 2 records with null values

DISTINCT CLAUSE:

It will display unique values from a column or unique records from a table.

syntax:

select DISTINCT colname1, colname2,...(or) DISTINCT ***** from <TBL>;

Ex: display list of different designitions?

select distinct(desg) " list of jobs" from emp_info;

output

salesman

developer

clerk

admin

ORDER BY Clause:

we can display the column data or table data in ascending / descending or sorting [a to z] / reverse sorting [z-a] data.

Syntax:

select col1, col2, ... / * FROM TBL order by col1, col2,.....[DESC] ;

Note:

By default table data is in random order.

Ex: Display employee names in alphabetical(sorting) order?

select ename from emp ORDER BY ename;

sample output:

ENAME

ADAMS

ALLEN

BLAKE

CLARK

FORD ...

Ex: Display employee names in reverse order?

select ename from emp ORDER BY ename DESC;

sample output:

ENAME

WARD

TURNER

SMITH

SCOTT

MILLER ...

Ex: display ename,sal, desg on the order of salary?

select ename, sal,job from emp order by sal;

OR

select ename, sal,job from emp order by 2;

sample output:

ENAME SAL JOB

SMITH	800	CLERK
JAMES	950	CLERK
ADAMS	1100	CLERK
WARD	1250	SALESMAN
MARTIN	1250	SALESMAN

OPERATORS

Arithmetic operators

RELATIONAL OPERATORS

SPECIAL OPERATORS

Relation Negation Operators

Logical Operators

Arithmetic Operators:

+ - * /

These operators are useful to perform arithmetic calculations on user's data and table data.

DUAL table:

DUAL is a system defined table. It has one column and one record. It is useful to calculate any constant expression.

Ex: select 200+300 from dual;

500

Ex: select (90000*10)/100 "10% of 90000" from dual;

output

9000

Ex: select 2000+(0.10*5000)-300 " After calculation" from dual;

2200

arithmetic Calculations On Table data

Ex: Display emp salaries and 2% of salary as TA?

select sal "Basic Sal", (0.02*sal) "TA" from emp;

Ex:

select pname, price, (0.25*price) Discount, price-(0.25*price) "final price"

from prod_dtls;

RELATIONAL OPERATORS

These Operators are useful to compare values. we can write conditions on the columns. The result of a condition is a BOOLEAN VALUE, that is either TRUE or FALSE.

< > = <= >=

WHERE clause

In any SQL Query [SELECT, UPDATE and DELETE], we will write conditions under this clause.

syntax:

select cl1, cl2,.....,cl-n / * from <TBL> where <condition>;

IQ: How to filter table data?

Ans: By using conditions under WHERE clause.

IQ: what is the result of a condition?

Ans: Boolean value , that is TRUE or FALSE.

Ex: display salaries below 12000?

select sal from emp_info where sal < 12000;

Ex: Display employee details who is getting above 12000 salary?

select * from empinfo where sal> 12000;

Ex: display the details of accounting dept?

select * from dept where dname='ACCOUNTING';

Ex: display the details of managers?

select * from emp where job='MANAGER';

SPECIAL OPERATORS

BETWEEN	**NOT BETWEEN**
IN	**NOT IN**
IS NULL	**IS NOT NULL**
LIKE	**NOT LIKE**

BETWEEN

By using this operator, we will write a condition based on range of values.

Syntax

select cl1, cl2,.....,cl-n / * from <TBL>

where <ColumnName> BETWEEN <First_val> AND <Last_val>;

Ex: Get emp details with minimum salary 20000 and maximum salary 35000?

Select * from emps where salary between 20000 and 35000;

IN

By using this operator, we can write a condition based on list of values in a column.

syntax:

select cl1, cl2,.....,cl_n / * from <TBL> where <ColumnName> IN(val1, val2, ...);

Ex: Display manager and clerk details?

Select * from emps where job in ('MANAGER','CLERK');

IS NULL

It used to check the column value is null or not. If column value is null then it returns TRUE, otherwise FALSE.

syntax

select cl1, cl2,.....,cl-n / * from <TBL> where <ColumnName> IS NULL;

Ex: Get employee details if the emp not having salary ?

Select * from emps where salary is null;

LIKE

It is useful to search for pattern in the column values. Pattern is a char or group of chars or symbols or digits.

LIKE operator is using 2 symbols to represent characters.

_ (underscore) It represent anyone char/digit/symbol.

% It represents zero or any number of chars.

syntax

select cl1, cl2,.....,cl-n / * from <TBL>

where <ColumnName> LIKE'pattern' [ESCAPE 'ch'];

ESCAPE 'ch' with LIKE

ESCAPE character is useful to search for _ and % symbols.

Ex: Display 3-digit salaries

select sal from emp where sal like'___';

Ex: Display salaries beginning with digit "2"?

select sal from emp where sal like'2%';

Ex: Display employee names begins with "J" and ends with "S"?

select ename from emp where ename like'J%S';

NEGATION OPERATORS

!= (or) <> (NOT EQUAL TO)

NOT BETWEEN

NOT IN

IS NOT NULL

NOT LIKE

Ex: Display all emps details except SALESMAN?

select * from emp where job<>'SALESMAN';

Ex: Display employee details not joined in the last year?

select * from emp where hiredate NOT BETWEEN '01-jan-14' and '31-dec-14';

LOGICAL OPERATORS

These operators are useful to write Multiple conditions under where clause.

AND

It returns true, if all conditions are true in a record.

OR

It returns true if any one condition is true in a record.

syntax

SELECT cl1,cl2,....., / * FROM TBL

WHERE <c1> [AND / OR] <c2> [AND / OR] <c3> [AND / OR].....;

Ex: Display manager details getting above 2500 sal?

select * from emp where job='MANAGER' and sal>2500;

Ex: Display clerks and salesman details if their salary at least 1000

and atmost 1500?

select ename,sal,job from emp where job in('CLERK','SALESMAN') AND sal between 1000 and 1500;

DML COMMANDS

UPDATE

By using update command, We can change old values with new values in the table.

By default, it updates all values in the column without where clause.

syntax

UPDATE <tbl> SET col1 = value, col2 = value, . . .

where <cond>;

Ex: update the commission of 7369 as 500?

update emp set comm=500 where empno=7369;

Ex: update all emps commissions as 1000?

update emp set comm=1000;

Ex: update the salesman salary with 20% increment , change their designition as Sr.SALES ?

update emp set sal=sal+(0.20*sal), job='Sr.SALES' where job='SALESMAN';

DELETE

It is used to delete the records from the table. By default, it will delete all records.

Note:

To delete specific records, then we must specify conditions along with delete command.

syntax

DELETE FROM <table_name> where <cond>;

Ex: delete all customer details?

delete from cust_dtls;

Ex: delete employees information who is not getting any commission?

delete from emp where comm is null;

TRANSACTION CONTROL LANGUAGE COMMANDS

Generally, DML operations on table data are considered as transactions.

To make transactions as permanent or cancel the transactions , we will use these commands.

COMMIT

It is used to make transactions as permanent. Once committed, we cannot cancel these transactions.

ROLLBACK

It is used to cancel user transactions. By default ROLLBACK cancel all transactions in the current login session.

SAVEPOINT

It is useful as a check point while cancelling transactions with ROLLBACK.

SAVEPOINT <name>;

Example:

```
create table cust
(
cid  char(3),
cname  varchar2(20)
);
insert into cust values('c00','Sanju');
insert into cust values('c01','Manoj');
select * from cust;
rollback;
select * from cust;
insert into cust values('c00','Sanju');
insert into cust values('c01','Manoj');
commit;
select * from cust;
rollback;
select * from cust;
```

Ex-2

```
Insert into cust values('c02','hellen');
```

delete from cust where cname='Sanju';

savepoint s1;

update cust set cid='c99' where cname='hellen';

savepoint s2;

delete from cust;

select * from cust;

rollback to s2; [It will cancel transactions after "S2"]

rollback; [It will cancel all transactions by default]

DCL commands [Data Control Language commands]

By using these commands the DBA can assign permissions or get back permissions to / from the users on the data base objects.

GRANT

It is used to assign permissions on database objects to the requested users.

OWNER (If the users are in same database)

SYN:

GRANT <Privillage list> ON <objectName> TO <schemaname/username>;

DBA

syn:

GRANT <Privillage list> On <DBName.username.objectName> TO <DBName.username>;

REVOKE

By Using this command, DBA can get back or cancel permissions.

OWNER(If the users are in same database)

syn:

REVOKE <Privillage list> On <objectName> FROM <schemaname>;

DBA

syn:

REVOKE <Previllage List> On <DBName.schemaname.objectName> FROM
<DBName.schemaname>;

PREVILLAGE LIST

Previllages are known as permissions.

SELECT

INSERT

UPDATE

DELETE

CREATE [TABLE/VIEW/INDEX/SYNONYM]

ALTER [TABLE/VIEW/SEQUENCE]

DROP [TABLE/VIEW/INDEX/SEQUENCE/SYNONYM]

DDL COMMANDS

CREATE **ALTER** **DROP**

RENAME **TRUNCATE**

ALTER

It is used to change the structure of the table by doing the following

Adding new columns----------ADD

Deleting Old columns----------DROP COLUMN

Changing column name ----------RENAME COLUMN

Change table name----------RENAME TO

Changing column size /datatype-------MODIFY

ADDING COLUMNS

Alter Table <table_name> ADD

(<col_1> datatype(size), <col_2> datatype(size), ----- -----);

EX: Add gender and mobile number columns in cust table?

```
alter table cust add
( gender   char, mobile  number(10) );
```

deleting a column

We can delete an entire column from table structure.

syntax:

```
Alter Table <table_name>
DROP COLUMN <col_name>;
```

Ex: delete customer id column

```
alter table cust drop column cid;
```

Change column Name

we can change column names of a table permanently as follows.

Syntax

```
Alter Table <table_name>
RENAME COLUMN <oldname> TO <newname>;
```

Ex:
```
alter table emp
rename column ename to empnames;
```

Change table name

we can change table name permanently.

syntax

```
alter table <table_name> RENAME TO  <new table name>;
```

change datatype and size

We can change datatype of a column and we can increase or decrease size of a column.

syntax:

```
Alter Table <table_name>
MODIFY <colname> new_datatype(new_size);
```

Note

A) If the column is empty then we can do the following

→We can change from any data type to any other data type.

→We can increase / decrease column size.

B) If column is not empty then,

→ NUMBER type and CHAR type column sizes cannot be decreased, but can be increased.

→Don't change datatype.

Ex:

alter table cust modify cname varchar2(20);

DROP

we can delete any data base object by using this command.

syn:

drop <object_type> <object_name>;

we can delete the table as follows:

drop table <tablename>;

Ex: drop table emp;

Ex: drop user dinesh;

RENAME

It is used to change name of oracle table.

syntax:

RENAME <Old TableName> TO <New TableName>;

Ex:

change the table name "customers" as "custinfo"?

RENAME customers TO custinfo;

TRUNCATE

It is useful to delete all the records permanently from the table.

syn: truncate table <table_name>;

Ex: truncate table custinfo ;

DATA INTEGRITY CONSTRAINTS

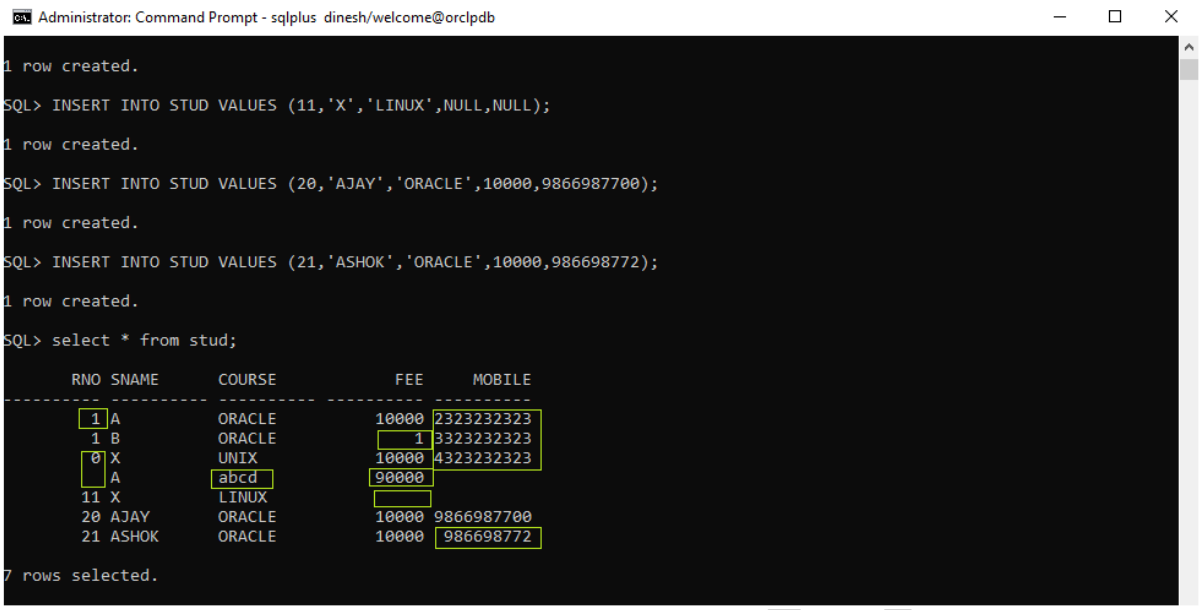
Constraints are known as set of rules / business rules which will be defined/changed at DDL level. Constraints, force the data base to accept only valid values in to the tables. Constraints ensure the user to fetch valid, complete and accurate data, from database.

For Example

Table creation without constraints:

```
CREATE TABLE STUD
(
RNO    NUMBER(2),
SNAME  VARCHAR2(10),
COURSE VARCHAR2(10),
FEE    NUMBER(5),
MOBILE NUMBER(10)
);

INSERT INTO STUD VALUES (1,'A','ORACLE',10000,2323232323);
INSERT INTO STUD VALUES (1,'B','ORACLE',1,3323232323);
INSERT INTO STUD VALUES (0,'X','UNIX',10000,4323232323);
INSERT INTO STUD VALUES (NULL,'A','abcd',90000,NULL);
INSERT INTO STUD VALUES (11,'X','LINUX',NULL,NULL);
INSERT INTO STUD VALUES (20,'AJAY','ORACLE',10000,9866987700);
INSERT INTO STUD VALUES (21,'ASHOK','ORACLE',10000,986698772);
```



Note:

In the above table the marked values are invalid according to business of client.

How to avoid / restrict invalid values into business database?

→BY defining constraints on the tables before data to be inserted into the table.

Constraints are divided into 3 categories.

1) Key Constraints

2) Domain Constraints

3) Referential Integrity constraints / Ref Constraints

KEY CONSTRAINTS

These constraints will verify individual values.

These are divided into 3 types.

UNIQUE

It doesn't allow duplicates, but allow null values.

Ex: phone numbers, mailid, etc...

email varchar2(30) UNIQUE,

aadhar number(12) UNIQUE,

NOT NULL

It doesn't allow null values, but allow duplicate values.

Ex: empNames, CityNames, ...

sname varchar2(20) NOT NULL,

PRIMARY KEY

It doesn't allow duplicates and doesn't allow null values. Generally Primary key is used to identify any record Uniquely. Only one primary key constraint is allowed per a table.

Ex:

bank account numbers

empid

regnumbers... etc.

Primary keys are 2 types.

→ **simple primary key [SPK]**

It is defined on single column.

→**composite primary key [CPK]**

It is defined on more than one column.

(max number of columns in to CPK are 32)

SYN:-1 Creating table with key constraints:

```
create table  <table_name>
(
col1  datatype(size) <constraint_name>,
col2  datatype(size) <constraint_name>,
----    -----          ----
----    -----          ----
);
```

SYN:-2 key constraints with User Defined Names

```
create table  <table_name>
```

```
(
col1  datatype(size)
   Constraint <friendlyName> PRIMARY KEY,
col2  datatype(size)
Constraint <friendlyName> NOT NULL,
col3  datatype(size),
constraint <friendly name> UNIQUE(col),
----     -----          ----
----     -----          ----
);
```

Ex:

create a table student with columns rno, sname, course, fee and mobile along with constraints pk, nn, nn, nn and unique respectively?

```
create table student
(
rno    number(2)      primary key,
sname  varchar2(10)  not null,
course varchar2(10) not null,
fee    number(5)      not null,
mobile number(10)   unique
);
```

```
insert into student values(1,'a','oracle',9000,'8989898989');
insert into student values(0,'b','java',2000,'8787878787');
insert into student values(2,'x','oracle',9000,null);
insert into student values(11,'s','abc',100,null);
```

data:-

RNO	SNAME	COURSE	FEE	MOBILE
1	a	oracle	9000	8989898989
0	b	java	2000	8787878787
2	x	oracle	9000	
11	s	abc	100	

ERROR GENERATING RECORDS:

insert into student values(1,'kiran','java',2300,null);

insert into student values(null,'kiran','java',2300,null);

insert into student values(12,null,'java',2300,null);

insert into student values(1,'kiran',null,2300,null);

insert into student values(1,'kiran','java',null,null);

insert into student values(1,'kiran','java',2300,8989898989);

Note:

Even after the key constraints on the table, still we have invalid values.

We can eliminate them by using DOMAIN constraints.

DOMAIN constraint:

By using Domain constraint, we can define conditions on the columns. Before inserting a value, the value is verified based on condition. We can write conditions by using any relational operator.

The name of domain constraint is CHECK.

syntax:

col datatype(size) <key constraint>,

check (col <condition>)

Ex:

rno number(3) primary key,

check (rno between 1 and 999)

How to get constraints from a table

Use the table USER_CONSTRAINTS

Ex:

select constraint_name, constraint_type from USER_CONSTRAINTS

where table_name='STUDENT';

CONSTRAINT_NAME	**CONSTRAINT_TYPE**
SYS_C007050	C -either Not null or Check
SYS_C007051	P --Primary key
SYS_C007052	U --Unique key
SYS_C007053	R --Foreign key

CONSTRAINTS with user-defined names

Syntax:-1

<col> datatype(size) CONSTRAINT <Friendly name> <Actual_name>

Ex:

rno number(3) constraint pk_rno_student primary key

Ex:

rno number(3),

constraint pk_rno_student primary key(rno)

Note:

The second example is not suitable for NOT NULL constraint.

create table stud_dtls

```
(
rno    number(2)
constraint  pk_rno_stud_dtls  primary key,
sname   varchar2(10)
constraint  nn_sname_stud_dtls not null,
course   varchar2(7)
constraint  nn_course_stud_dtls  not  null,
fee   number(5)
constraint nn_fee_stud_dtls not null,
mobile  number(10)
constraint  uk_mobile_stud_dtls  unique,
CONSTRAINT ck_rno_stud_dtls
      check(rno between 1 and 60),
     constraint ck_course_stud_dtls
check (course in('oracle','java','unix')),
     constraint ck_fee_stud_dtls
     check(fee between 12000 and 20000),
constraint ck_moblen_stud_dtls
     check(length(mobile)=10),
constraint ck_mob_valid_stud_dtls
check(mobile like'7%' or mobile like'8%' or mobile like'9%' or
            mobile like'6%')
);
```


Ex:

```
select constraint_name,constraint_type from user_constraints
where table_name='STUD';
```

NORMALIZATION AND DENORMALIZATION

According to key and domain constraints we can maintain valid data in the tables.

Whenever we are accessing related information then data is not retrieved from tables. This is known as communication gap between database server and client user. To eliminate communication gap, we can use either DENORMALIZATION or NORMALIZATION methods.

emp

eid	ename	sal
1	a	2000
2	x	1200
3	a	3400
4	z	5000
5	c	1000
6	s	1300
7	d	2300
8	x	1200
9	b	2200

dept

dno	dname	loc
10	production	hyderabad
20	sales	hyderabad
30	finance	chennai

Note

By using above tables we are unable fetch the complete data of an object like, department name of any employee, number of emps in dept and etc.

To answer such kind of requirements we have to maintain the data in 2 methods.

1) Maintaining all the information in one table. [DENORMALIZATION]

2) Maintaining data in different tables and implement relationships between the tables. [NORMALIZATION]

DENORMALIZATION

Maintaining all necessary information in one big table is known as Denormalized method.

Emp_Dept_Details

eid	ename	sal	dno	dname	loc
1	a	2000	10	productior	hyderabad
2	x	1200	10	productior	hyderabad
3	a	3400	10	productior	hyderabad
4	z	5000	10	productior	hyderabad
5	c	1000	20	sales	hyderabad
6	s	1300	20	sales	hyderabad
7	d	2300	20	sales	hyderabad
8	x	1200	30	fin	chennai
9	b	2200	30	fin	chennai

Advantage:

Communication gap is eliminated.

DRAWBACKS

→ data duplicacy

→ Occupy more disk space

→ Data retrieval time is very long

Note:

To decrease above drawbacks we can apply normalization process

as follows.

NORMALIZATION

The Process of Dividing big table in to sub tables, until Max. data duplicacy is reduced is known as normalization process.

Ist NF(normal form)

Dividing big table into sub tables based on repeated groups of data.

EMP

eid	ename	sal
1	a	2000
2	x	1200
3	a	3400
4	z	5000
5	c	1000
6	s	1300
7	d	2300
8	x	1200
9	b	2200

DEPT

dno	dname	loc
10	productior	hyderabad
10	productior	hyderabad
10	productior	hyderabad
10	productior	hyderabad
20	sales	hyderabad
20	sales	hyderabad
20	sales	hyderabad
30	fin	chennai
30	fin	chennai

IInd NF

Eleminate duplicate records and defining primary keys in the above tables

EMP

PK

eid	ename	sal
1	a	2000
2	x	1200
3	a	3400
4	z	5000
5	c	1000
6	s	1300
7	d	2300
8	x	1200
9	b	2200

DEPT

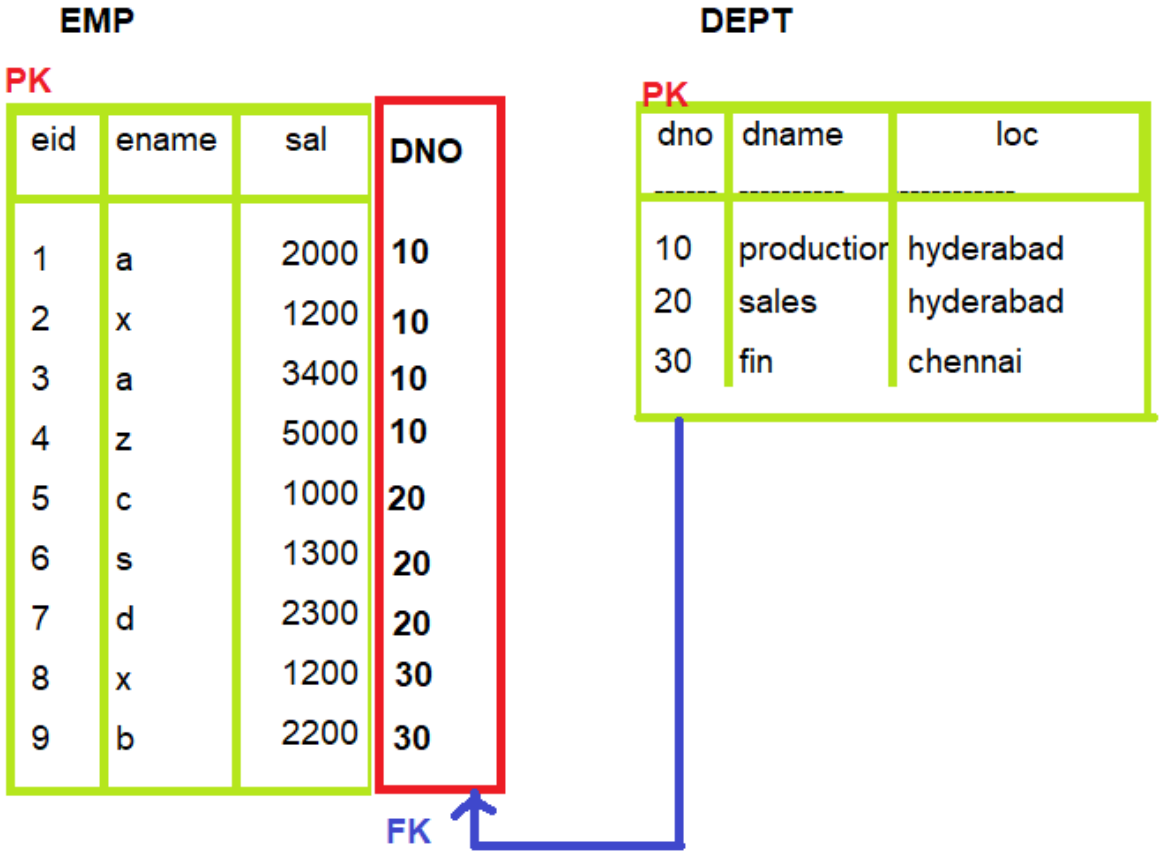
PK

dno	dname	loc
10	productior	hyderabad
20	sales	hyderabad
30	fin	chennai

IIIrd NF [Boyce Codd Normal Form] (BCNF)

We can define relation between the tables. Consider PK of parent table and define Fk column in the child table.

We can define FK column, by using Referential Integrity Constraint.



Adv:

→No communication gap.

→Duplicacy is reduced

→Occupy less disk space.

→Data search time is reduced

REFERENTIAL INTEGRITY CONSTRAINT / REF CONSTRAINT

BY USING THIS, WE CAN DEFINE RELATION BETWEEN TABLES.

→Foreign key column contains only values from primary key.

→Foreign key contains duplicates and null values also.

→A table which has primary key is considered as parent/Master/Base table.

→A table which has foreign key is known as Child/Detailed/Derived table.

REFERENCES

It is used to define a FK column in the child table, by using parent table PK column.

Defining FK with default name

Syntax:

<col_name> datatype(size) REFERENCES <parent_table>(PK_column)

Note

Generally, PK and FK column names and datatypes are similar.

SYNTAX:

FK with user Defined name:-

constraint <friendly_name> FOREIGN KEY(ChildTablecolumn name)

REFERENCES <Parent_Table>(PK_colname)

Examples:

consider the tables comp_dtls and prod_dtls.

And we are maintaining set of products from each company.

Based on this create the tables and maintain relation between

the tables?

Ex: create comp_dtls as parent table

Ex: create prod_dtls as child table

```
create table  comp_dtls
(
cmpid  char(5)
constraint pk_cmpid_cmp primary key,
cmpname    varchar2(20) not null,
Country     varchar2(20) not null
);
```

```
create table prod_dtls
(
pid             char(4) primary key,
pname       varchar2(20) not null,
cost          number(7,2),
mfg          date,
warrenty   varchar2(10),
cmpid         char(5),
constraint fk_cmpid_prod FOREIGN KEY(cmpid)
REFERENCES COMP_DTLS(cmpid)
        );
```

Managing constraints

ADDING / DELETING / CHANGE CONSTRAINTS on existing table:-

→we can add constraints

→we can delete constraints

→we can change constraints

By using "ALTER" command we can add/remove/change constraints on the existed table.

ADDING PRIMARY KEY

Syntax

alter table <table_name> ADD

CONSTRAINT <friendly name> PRIMARY KEY(colname);

Ex:

add primary key on RNO column in the table "student"?

```
ALTER TABLE STUDENT
ADD
CONSTRAINT pk_rno_student PRIMARY KEY(rno);
```

Adding CHECK constraint

syntax

alter table <table_name> add constraint <friendly name> CHECK(condition);

Ex:

check student fee between 1000 and 5000?

ALTER TABLE student

add

constraint ck_fee_student

check(fee between 1000 and 5000);

Adding UNIQUE constraint

syntax:

ALTER TABLE <name> ADD constraint <friendlyName> UNIQUE(col);

Ex:

Add unique constraint on email column?

ALTER TABLE student

add

constraint uk_email_student UNIQUE(email);

Adding FOREIGN KEY

syntax:

alter table <table_name> ADD CONSTRAINT <friendly name> FOREIGN KEY(child_col) REFERENCES <parent>(pk_col);

Ex:

Add foreign key on cmpid column in product table?

ALTER TABLE product

add

CONSTRAINT fk_cmpid_product FOREIGN KEY(cmpid)

references company(cmpid);

How to add / remove not null constraint?

MODIFY

syntax

alter table <TBL> MODIFY col datatype(size) NOT NULL;

or

alter table <TBL> MODIFY col datatype(size) constraint <friendlyname> NOT NULL;

Deleting NOT NULL constraint:

ALTER TABLE <name> MODIFY col datatype(size) NULL;

How to delete any constraint?

Syntax

alter table <name> DROP CONSTRAINT <DefaultName/FriendlyName>;

Ex

Delete unique constraint on mobileno?

ALTER TABLE student DROP CONSTRAINT uk_mobileno_student;

DEFAULT

We can define a default value into a column. That is if we did not insert any value into default column, then the default value will be inserted. Default column also accept NULL values and other values.

→To stop other values , define CHECK constraint

→To stop NULL values, define NOT NULL constraint

SYNTAX

col datatype(size) DEFAULT 'value'

Example

Create table customers

(

cid number(2),

cname varchar2(10),

city   varchar2(10) DEFAULT 'DELHI'

);

Insert into customers(cid,cname) values(11,'krish');

Insert into customers(cid,cname) values(22,'raj');

Insert into customers(cid,cname) values(33,'john');

Select * from customers;

11	krish	DELHI
22	raj	DELHI
33	john	DELHI

SEQUENCES

Sequence is a database object. It is useful to generate sequential integers.

In real time projects, we can define sequences to generate primary key column values.

syntax:

******

CREATE SEQUENCE <seq_name>

START WITH <val>;

By default , sequence value start with 1 and increment by 1

PSEUDO COLUMNS of sequence

The following are virtual columns associated with any sequence.

CURRVAL

It will access current value of sequence.

Ex:

select <seq_name>.currval from dual;

NEXTVAL

It will access next value from sequence.

Ex:

select <seq_name>.nextval from dual;

SEQUENCE USES

i) we can use sequence values while inserting records.

ii) we can use sequence values in updating a column.

Ex:

create sequence custid

start with 11111;

Ex:

select custid.nextval from dual;

11111

Ex:

select custid.nextval from dual;

11112

Ex:

select custid.currval from dual;

11112

Ex:

create a table custinfo with columns CID,CNAME,CITY.

Insert customer names and cid values genereted by

sequence "custid".City value by default "HYDERABAD".

1)

```
create sequence custid
start with  11111;
```

2)

```
create table custinfo
(
cid          number(6)  DEFAULT  custid.nextval,
cname    varchar2(20) constraint nn_cname_custinfo  NOT NULL,
city        VARCHAR2(10)  DEFAULT  'HYDERABAD'
);
```

SET OPERATORS

We can display combined data from multiple tables.

The operators are

UNION ALL

UNION

INTERSECT

MINUS

UNION

It will display combined data from multiple tables without duplicates.

SYNTAX

```
SELECT . . . . . FROM  T1
UNION
SELECT  . . . . .  FROM  T2;
```

UNION ALL

It will display combined data from multiple tables with duplicates.

SYNTAX

```
SELECT . . . . . FROM  T1
UNION ALL
SELECT  . . . . . FROM T2;
```

INTERSECT

It will display common data from multiple tables. (From multiple Select stmts)

SYNTAX

```
SELECT . . . . . FROM  T1
INTERSECT
SELECT  . . . . .  FROM T2
-----  ;
```

MINUS

It will display values from first selection by eliminating values which are repeating in second selection.

SYNTAX

```
SELECT . . . . .  FROM  T1
MINUS
SELECT  . . . . . FROM T2
MINUS
-----;
```

Sample Tables:

```
CREATE TABLE CUST_BR1
(
```

```
CID CHAR(3),
CNAME VARCHAR2(20),
MOBILE NUMBER(10),
CITY  VARCHAR2(20),
GENDER  VARCHAR2(10)
);

INSERT INTO CUST_BR1 VALUES('C1','VIJAY',9800198001,'HYD','MALE');
INSERT INTO CUST_BR1 VALUES('C2','JOHN',9800298002,'DELHI','MALE');
INSERT INTO CUST_BR1 VALUES('C3','SWATHI',9877987700,'HYD','FEMALE');

CREATE TABLE CUST_BR2
(
CID CHAR(3),
CNAME VARCHAR2(20),
MOBILE NUMBER(10),
CITY  VARCHAR2(20),
GENDER  VARCHAR2(10)
);

INSERT INTO CUST_BR2 VALUES('C1','KIRAN',9898989898,'HYD','MALE');
INSERT INTO CUST_BR2 VALUES('C2','JOHN',9800298002,'DELHI','MALE');
INSERT INTO CUST_BR2
VALUES('C3','LAKSHMI',8989898989,'DELHI','FEMALE');

CREATE TABLE CUST_BR3
(
CID CHAR(3),
CNAME VARCHAR2(20),
MOBILE NUMBER(10),
CITY VARCHAR2(20),
```

```
GENDER VARCHAR2(10)
);

INSERT INTO CUST_BR3 VALUES('C1','KIRAN',9898989898,'HYD','MALE');
INSERT INTO CUST_BR3 VALUES('C2','JOHN',9800298002,'DELHI','MALE');
INSERT INTO CUST_BR3 VALUES('C5','VINAY',7878787878,'DELHI','MALE');
```

Examples:

```
--female customers from br1 and b3

select * from cust_br1 where gender='f'
union all
select * from cust_br3 where gender='f';

--
select cname,mobile from cust_br1
intersect
select cname,mobile from cust_br2
intersect
select cname,mobile from cust_br3;

--Display all customers info from all branches

select * from cust_br1
union all
select * from cust_br2
union all
select * from cust_br3;
```

--Get the customer details without duplicates

```
select * from cust_br1
union
select * from cust_br2
union
select * from cust_br3;
```

--DISPLAY COMMON CUSTOMER NAMES AND MOBILE NUMBERS FROM ALL BRANCHES

```
SELECT CNAME,MOBILE FROM CUST_BR1
INTERSECT
SELECT CNAME,MOBILE FROM CUST_BR2
INTERSECT
SELECT CNAME,MOBILE FROM CUST_BR3;
```

--DISPLAY CUSTOMERS DETAILS WHO IS THE ONLY CUSTOMER FOR BRANCH 2

```
SELECT * FROM CUST_BR2
MINUS
(
SELECT * FROM CUST_BR1
UNION ALL
SELECT * FROM CUST_BR3
```

);

Note:

What are the limitations of set operators?

--we need to select equal number of columns from each table

--we need to select same data type of data in the same sequence under each select query.

Ex:

```
SQL> select cname,mobile from cust_br1
  2  union all
  3  select cid from cust_br2;
select cname,mobile from cust_br1
*
ERROR at line 1:
ORA-01789: query block has incorrect number of result columns
```

```
SQL> select cname,mobile from cust_br1
  2  union all
  3  select mobile,cname from cust_br2;
select cname,mobile from cust_br1
       *
ERROR at line 1:
ORA-01790: expression must have same datatype as corresponding expression
```

LIMITATIONS of SET Operators:

--All select queries should contain equal number of columns

--The selecting columns must have same data type in the same order

PSEUDO COLUMNS

These are virtual columns associated with any table.

ROWNUM

ROWID

1) ROWNUM

It maintains serial number for each record in the output.

By using this we can select "n" number of records from the begining of the table.

Ex: select * from emp
where rownum<=5;

2) ROWID

It maintains physical address of each record.

By using this value, oracle engine identify any record in the database server.

JOINS

Joins are useful to display data from multiple related tables.

Types of joins: 4

i) CROSS JOIN or CARTESIAN PRODUCT

ii) EQUI JOIN / INNER JOIN

iii) SELF JOIN

iv) OUTER JOINS

→Left Outer Join

→Right Outer Join

→Full Outer Join

CROSS JOIN

It will display all possible combinations from multiple tables.

Note:

Cross join is also known as Join / Cartesian Product.

Total No. of Combs= No. of recs in TBL1 X No. of recs in TBL2 X . . . .

syntax

select col1, col2,...,coln / *

from table1, table2, ...

where <cond>

Ex:

Get all emp details and dept details?

Select * from emp, dept;

Or

Select e.*, d.* from emp e, dept d;

Note:

In the above example, e and d are known as alias names for the tables emp and dept respectively.

EQUI JOIN

It will display only matched data from multiple tables based on join condition.

A condition is known as join condition if it is specified between primary key of one table and foreign key of other table.

syntax

select col1, col2,...,coln / *

from table1, table2, table3, ...

where t1.col=t2.col

and

t2.col=t3.col .....;

Ex:

Get employee details and corresponding dept details?

Select e.*, d.*

From emp e, dept d where e.deptno=d.deptno;

INNER JOIN

It will display only matched data from multiple tables like Equi join.

In the inner join query, table names are separated with INNER JOIN key word.

And we should write join condition under **ON** clause.

Syntax

select . . . .

from tb1 INNER JOIN tb2

ON tb1.col=tb2.col;

Ex: Consider the tables **PRODUCT** and **COMPANY** with common column **cmpid**

1) Display all product details along with company name?

Select p.*, c.cmpname from product p inner join company c on p.cmpid=c.cmpid;

SELF JOIN

A table which is joined itself is known as self join. In this join, we will consider 2 alias names for one physical table. Based alias names , we can write join condition.

We can display output from 2nd alias table only.

synta:

```
select alias2.*
from TBL alias1, TBL alias2
where
    alias-1.col='value'
      and
     alias1.col=alias2.col;
```

employee

ename	city
kiran	mumbai
hari	hyd
madhu	hyd
smith	delhi
scott	mumbai
allen	hyd
soumya	mumbai
john	delhi

Ex:

Get emp details from a city where "ALLEN" is living?

```
select * from employee
        where ename='allen';  [ NOT CORRECT ]
```

Ex:

display emplyoee details who is living in a city

where "john" is living?

```
select e2.ename,e2.city
from employee e1 inner join employee e2
on e1.ename='JOHN' and e1.city=e2.city;
```

OUTER JOINS

Outer joins will display all the data from one table and only matched data from other table.

In any outer join query, we should write outer join type between table names

and join condition should be specified under ON clause.

We have 3 types of outer joins.

1) **Left outer join / left join**

Display all data from left table and only matched data from right table.

2) **Right outer join / Right join**

Display all data from right table and only matched data from left table.

3) **Full outer join / Full join**

It will Display

→matched data from both tables

→unmatched records from left table

→unmatched records from right table

syntax

```
select col1, col2, col.....  / *
from    TBL_1  [Left Join / Right Join / Full Join]   TBL_2
ON tbl1.col=tbl2.col
```

--Display products information along with company information

/* EQUI JOIN */

SELECT P.*,C.* FROM PROD_DTLS P, COMP_DTLS C

WHERE P.PROD_COMP_ID=C.COMP_ID;

/*LEFT OUTER JOIN */

SELECT P.*,C.* FROM PROD_DTLS P LEFT OUTER JOIN COMP_DTLS C

ON P.PROD_COMP_ID=C.COMP_ID;

/*RIGHT OUTER JOIN */

SELECT P.*,C.* FROM PROD_DTLS P RIGHT OUTER JOIN COMP_DTLS C

ON P.PROD_COMP_ID=C.COMP_ID;

/*FULL OUTER JOIN */

SELECT P.*,C.* FROM PROD_DTLS P FULL OUTER JOIN COMP_DTLS C

ON P.PROD_COMP_ID=C.COMP_ID;

SUB QUERIES

A query with in the other query is known as Nested query. **Sub query** is one type of Nested Query.

"A select query under where clause of other select query is known as Sub Query."

Sub queries are preferable to display output from one table and having an input value from the other table.

To write a subquery we need at least one common column.

syntax

select ..... from table...

where <cond> (select ..... from table...where <cond> (select.....).....);

Hints

i) Based on output table we can write outer query.

ii) Based on input value table we can write sub query.

NOTE: Execution process of a subquery is from innermost query

to the outermost.

outer query<---o/p<--Inner query<---o/p<---Inner query

TYPES OF SUBQUERIES:

1) **Single row subquery**

A sub query, which select a value from single record. In this case in between the outer and inner queries we can use = operator.

syntax

select . . . . from . . . where col =(select . . . . from . . . where. . .);

2) **Multi row sub query**

A sub query, which select values from multiple records. In this case in between the outer and inner queries we can use IN operator.

Syntax

select . . . . from . . . where col IN(select . . . . from . . . where. . .);

Ex: display department details of employee smith?

```
select * from dept
where deptno IN(select deptno from emp where ename='SMITH');
```

Ex: Display employee details who is working under PRODUCTION?

```
select * from emp where deptno=(select deptno from dept where
      dname='PRODUCTION');
```

Ex: display the department details of all managers?

```
select * from dept
where deptno IN(select deptno from emp where job='MANAGER');
```

VIEWS

View is a database object like table. View contains logical copy of table data.

View is created based on table or other view. It can be created based on frequently using data. It improves the performance at data access level. ***View*** *maintains data dynamically, means if table data is changed then the changes reflected into view dynamically.*

View Types

1) Simple or Updateable view

2) Composite or read-only or Join view

NOTE: By default we don't have permission to create views. That is assigned by DBA.

i) connect as DBA

ii) grant ***create view*** to ***dinesh***;

iii)connect as dinesh;

iv) Now try to create a view.

Example:

How to get create view permission?

i)

ex: connect as dba in oracle 11g

sql> conn system/manager

Ex: grant create view to manju;

grant succeeded

iii) Connect as client again

In oracle 11g

Ex: conn manju /welcome

1) simple view

This view is created based on single table data. we can execute select, insert, update and delete commands on this view. That's why simple view is also known as Updateable view.

syntax

create view <view_name>

AS

select ..... from table_name ;

Ex:

write a query to create a view which contains managers information

from emp table?

create view vw_managers

as

select * from emp where job='MANAGER';

Ex:

Select data from a view:

select * from vw_managers;

2) Composite view:

This view is created based on more than one table data. It doesn't allow insert operation on it. It is also known as read only view or Join view.

Advantage:

It will reduce no. of times writing join queries again and again.

syntax:

create view <vw_name>

AS

select .....

from table1, table2,...

where <join-cond>;

EX:

Write a query to create a view to maintain product information along with company details?

Create view vw_prod_company

As

Select * from product p, company c

Where p.cmpid=c.cmpid;

INDEXES

Index is a database object like table. It is useful to search data as much as fast based on conditions.

Index has 2 parts.

They are

→ data part

→ address part.

Data part contains ordered values of a column.

Address part contains physical address of each record.

Index occupy physical disk space.

TYPES OF INDEXES

1) simple index:

This index is created on single column of a table.

syntax

create index <idx_name> on table_name(colname);

Ex:

create index , to search data based on salary column?

create index idx_sal on emp(sal);

2) composite index

This index is created on multiple columns of a table.

Syntax

create index <idx_name> on table_name(col1 ,col2,....);

Ex:

create an index on the columns cost,comp_code in the table prod_dtls?

```
create index idx_prod_search
on prod_dtls(cost,comp_code);
```

ORACLE SQL -- BUILT IN FUNCTIONS

Oracle is providing a set of predefined functions. Each function is useful to perform one task. These functions are divided into 2 CATEGORES.

1) GROUP / AGGREGATE FUNCTIONS:

It will execute on set of values and display single output value. We can also say, it can execute on column level / field level.

Note: Group functions cannot applicable on user's data.

Ex: sum(), avg(), min(), max(), count(), count(*)

2) SCALAR / SINGLE ROW FUNCTIONS:

It will execute on set of values and display a set of output values. We can also say, it is be executed on Record level / row level.

Ex: lower(), length(), trim()

NUMERIC FUNCTIONS [GROUP Category]

These functions are executing on columns only.

SUM(col) , AVG(col), MAX(col), MIN(col), COUNT(col), COUNT(*)

i) SUM(colname)

It will display addition of values from given column.

Ex: display addition of all salaries?

```
select sum(sal) from emp;
```

ii) AVG(colname)

It will display average value from given column.

Ex: display average product cost?

```
select avg(cost) from prod_dtls;
```

iii) MAX(colname)

It will Display highest value from given column.

Ex: Display highest salary among all salesman?

```
select max( sal ) from emp where job='SALESMAN';
```

iv) MIN(colname)

It will Display least value from the given column.

Ex: Find out least cost among all laptops?

```
select min(cost) from prod_dtls where prod_name='LAPTOP';
```

v) COUNT(colname)

It will Display count of NOT NULL values from given column.

Ex: find how many number of emps getting commission?

```
select count(empno) from emp where comm is not null;
```

vi) COUNT(*)

It will Display number of records from given table.

Ex: display number of transactions on current day?

```
select count(*) from trans_dtls where trans_date=sysdate;
```

GROUP BY clause:

Internally, it will make logical groups based on each distinct value from given column. On each logical group, aggregate functions are executed individually.

HINT:

IN any select query, if there exist column names and aggregate functions then we must use GROUP BY clause.

→***"The column names from select list, should be present after GROUP BY clause."***

Note: Don't use GROUP BY and DISTINCT clauses together in a query.

SYNTAX:

```
select  colname, colname,..., aggregate1, aggregate2,.....
from  table WHERE  <cond> GROUP BY <col1>,<col2>,...HAVING aggfunc1,..., ORDER BY  col1,col2,.....;
```

Ex: find out number of emps working under each dept ?

select deptno, count(*) " No. of emps" from emp GROUP BY deptno;

output:

deptno	No. of emps
30	6
20	7
10	5

HAVING clause:

we can specify conditions on aggregate functions. By this, group by output is filtered.

Note: Without group by clause, don't use HAVING clause.

Ex: find out number of emps working under each dept on order of deptno if a dept contains at least 10 emps?

```
select deptno, count(*) " No. of emps"
from emp
GROUP BY deptno
HAVING count(*)>=10
order by deptno;
```

NUMERIC FUNCTIONS (SCALAR Category):

These functions are executing on record level.

1)ABS(-n) [ABSOLUTE]

It will convert given negative value as positive value.

Ex: select abs(-9) from dual;

9

2) MOD(m,n) [MODULUS]

It will display remainder value after m devided by n.

Ex: select mod(17,3) from dual;

2

3)POWER(m,n)

It will display "m" to the power of "n"th value.

Ex: select power(5,4) from dual;

625

4) SQRT(n) [Square Root]

It will display square root value of "n".

Ex: select sqrt(625) from dual;

25

5)ROUND(m,n)

It will Display value "m" which is rounded to the "n" number of places.

If "n" is positive, then given value is rounded in decimal part.

If "n" is negative, then given value is rounded in integer part.

Before displaying "nth" digit ,round() will check " n+1"th digit, if it is >= 5 then "n"th digit increment by 1.

Syntax SELECT ROUND(M,N) FROM DUAL;

Ex:

select round(2519.8235621,2) from dual;

6) TRUNC(m,n)

Display value "m" which is truncated to the n number of decimal places.It will never check any "n+1" digit and never make any increments.

Ex: select trunc(63.354,1) from dual;

63.3

Ex: select trunc(69.554) from dual;

69

7) FLOOR(n)

It will display highest integer below or equal to given value.

Ex: select floor(64.2) from dual;

64

8) CEIL(n)

It will display least integer above or equal to given value.

Ex: select ceil(64.2) from dual;

9) LEAST(val1,val2,....)

It will display minimum value from the given values and expression results. And also, least value from each record of given columns.

EX: select least(32,(6*5), (20-10), (36/2)) from dual;

10

10) GREATEST (val1, val2, ...)

It will display highest value from given values, expressions and columns. And also, highest value from each record of given columns.

Ex: select greatest (32,(6*5), (20-10), (36/2)) from dual;

32

Example:

```
create table stud_marks
(
sid        number(3),
telugu     number(3),
english    number(3),
hindi      number(3),
physics    number(3),
social     number(3),
chemistry  number(3)
);
insert into stud_marks values
(111,'98','87','90','95','78','80');
insert into stud_marks values
(222,'80','87','91','96','78','80');
insert into stud_marks values
(333,'60','79','56','78','78','89');
insert into stud_marks values
(444,'61','89','67','71','92','54');
insert into stud_marks values
```

(555,'51','62','79','79','60','82');

STRING/CHAR FUNCTIONS

1) ASCII('ch')

[American Standard code For Information Interchange]

It will display ascii value of given character.

Ex: select ascii('a') "a",ascii('A') "A" ,ascii('@') "@" from dual;

2) LENGTH('str'/col)

It is used to count number of characters in the given value.

Ex: Get employee names and length of each employee name?

select ename, length(ename) " length" from emp;

Ex: select length('oracle') from dual;

3) LOWER('str'/col)

It will display the given chars or column values in lower case.

Ex: Display employee names in lower case?

select ename,lower(ename) from emp;

Ex: select lower('HAI') from dual;

hai

4) UPPER('str'/col)

It will display given character value or column values in upper case.

Ex: select upper (pname) from products;

5) INITCAP('str'/col) [Initial Capital]

It will display the given char value or column values with beginning char as capital.

Ex: select **initcap**('welcome to oracle') from dual;

Welcome To Oracle

6) SUBSTR('str'/col,m,n) (substring)

It is used to display a substring from the given string. **"m"** is indicating from which charater, **"n"** is indicting number of characters.

If **"n"** is omitted then, it display from **"m th"** char till the end of string. If "m" value is negative, then substring is taken from end of string.

ex: select substr('secured',3,4) from dual;

cure

7) INSTR('str'/col,'ch',m,n) [instring]

It is used to display "n th" occurance of given char ,either from begin or end of given string.

Here "m" value is either +1(default), or -1

+1 Means search the character from the beginning of string.

-1 Means search the character position from the end of string.

Here "n" is nth occurance of given character.

ex: select instr('welcome','e') from dual;

2

8) TRANSLATE('str'/col, 'sourcechars','targetchars')

It will display given value by translating source chars with corresponding target chars.

Ex: select translate('welcome','em','xy') from dual;

wxlcoyx

Ex: select translate('welcome','em','x') from dual;

wxlcox

[char "m" eleminated from output]

9) REPLACE('str'/col, 'source str','target str')

It will display given value by replacing source string with target string.

Ex: select replace('welcome','come','sys') from dual;

welsys

Note: How do i remove a char from given string?

Ex: select replace ('welcome to oracle','e','') from dual;

10) TRIM('str'/col)

Display given string by deleting spaces before and after the string.

Ex:

select trim(' welcome to ') " trim", initcap('oracle') from dual;

welcome to Oracle

11) LTRIM('str'/col) [Left Trim]

Display given string by removing blank spaces from left side of string only.

Ex: select ltrim(' welcome to '), initcap('oracle') from dual;

welcome to Oracle

12) RTRIM('str'/col) [right trim]

Display given string by removing blank spaces from right side of string only.

Ex: select rtrim(' welcome to '), initcap('oracle') from dual;

output: welcome to Oracle

TRIM KEYWORDS

These keywords are useful to delete given character, either from left side or from right side or from both sides.

13) LEADING 'ch' FROM 'str'/col

Display given string by removing similar char from left side.

Ex: select trim(leading 'x' from 'xxx2489xxxx') from dual;

2489xxxx

14) TRAILING 'ch' FROM 'str'/col

Display given string by removing similar char from right side.

Ex: select trim(trailing 'x' from 'xxx2489xxxx') from dual;

xxx2489

15) BOTH 'ch' FROM 'str'/col

Display given string by removing similar char from both sides.

Ex: select trim(both 'x' from 'xxx2489xxxx') from dual;

2489

16) LPAD('str'/col,n,'ch') [left padding]

Display given string along with the special char in the left of given value. Special character is printed for "n-length ('str') " number of times.

str/col → input value

n → output length

ch → printing char

Ex: select lpad('page 1',12,'*') from dual;

******page 1

17) RPAD('str'/col, n,'ch') [right padding]

Display given string along with special char in the right of string. Special character is printed for " n-length('str') " no. of times.

Ex: select rpad('page 1',12,'*') from dual;

page 1******

Ex: print "welcome" with symbol "*" in both sides(10 times)?

select lpad(rpad(' welcome ',19,'*'),29,'*') from dual;

DATE FUNCTIONS

Date functions are executing on date type data.

1) SYSDATE

It will display system date. We can also add or subtract "n" number of days to the sysdate.

Ex: select sysdate from dual;

DD-MON-YY

26-DEC-21

Ex: find out the date of 10 days back?

select sysdate-10 from dual;

Ex: find out date value after 41 days?

select sysdate+41 from dual;

2) LOCALTIMESTAMP

It will display current date value along with time component as follows.

DD-MON-YY HH.MM.SS.ns [AM/PM]

Ex: select localtimestamp from dual;

26-DEC-21 12.20.52.694000 PM

3) CURRENT_TIMESTAMP [from oracle 11g]

It will display system date along with time component and time zone as follows.

DD-MON-YY HH.MM.SS.ns [AM/PM] timezone

Ex: select current_timestamp from dual;

DD-MON-YY HH.MM.SS.ns [AM/PM] timezone

26-DEC-21 06.15.23.000023 AM +5:30

4) TO_DATE('Date_Value','Value Format')

It will convert any non-Oracle date value into oracle's date format and display the result. It accepts any char format of date(dd/mm/yy or dd-mm-yyyy or dd:mon:yyyy or yyyy-mm-dd) and converts it into oracle's default date format [DD-MON-YY].

Ex: select to_date('02/12/2019','dd/mm/yyyy') from dual;

Ex: select to_date('2020:01:21','yyyy:mm:dd') from dual;

5) ADD_MONTHS(d,n)

It will display resultant date value after adding/subtracting "n" number of months to the given date.

"d" for date value.

"n" for number of months.

Ex: get date value after six months from today?

select add_months(sysdate,6) from dual;

Ex: get date value before 10 months from the date "22/may/2014"?

select add_months(to_date('22/may/2014','dd/mon/yyyy'),-10) from dual;

6) MONTHS_BETWEEN(d1,d2)

We can find out number of months between d1 and d2 dates. If first date is lessthan second date, then result is in negative.

Ex: select months_between(sysdate,'21-may-18') from dual;

7)LAST_DAY(d)

It will display last date value of month in the given date.

Ex: select last_day(sysdate) from dual;

31-JAN-20

8)NEXT_DAY(date,'dayname')

It will display date value of given day after given date.

Ex: select next_day(sysdate,'saturday') from dual;

Ex: select next_day(sysdate,'monday') from dual;

9) EXTRACT(Date PART FROM DATE_VALUE)

The Oracle/PLSQL EXTRACT function extracts a value from a date or interval value.

Syntax: The syntax for the EXTRACT function in Oracle/PLSQL is:

EXTRACT (YEAR or MONTH or DAY FROM date_value)

Note

You can only extract YEAR, MONTH, and DAY from a DATE.

YEAR to extract year number

MONTH to extract month number

DAY to extract digits of date

HOURto extract hours

SQL> select extract (YEAR from sysdate) from dual;

SQL> select extract (DAY from sysdate) from dual;

SQL> select extract (MONTH from sysdate) from dual;

10) TRUNC Function (with dates)

The TRUNC function returns a date truncated to a specific unit of measure.

Syntax

TRUNC (date [, format])

For example:

year date is truncated to beginning of year

Q to the beginning of quarter

Month to the beginning of the month

Ex: select TRUNC(TO_DATE('22-AUG-03'), 'YEAR') from dual;

iv) CONVERSION FUNCTIONS

1) TO_NUMBER(char_value_with_digits)

It will convert the given character values into number type.

Here the character value should contain sequence of digits.

Ex: salary varchar2(10)

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

$1200

$11005

$107069.12

```
create table empinfo
(
eid  number(3),
sal varchar2(10)
);
insert into empinfo values(111,'$1200');
insert into empinfo values(222,'$3000');
insert into empinfo values(333,'$900');
insert into empinfo values(444,'$1900');
insert into empinfo values(555,'$3000');
insert into empinfo values(666,'$14000');
```

Ex: select sum(to_number(substr(sal,2))) totalsal from empsal;

2) TO_CHAR(num, [char format])

It is converting number type values into character format.

Ex:

price target_format

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

199.78 00,00,199.78

1200.12 00,01,200.12

5463.00(number data) 00,05,463.00(char format)

100700.00 01,00,700.00

1223501.01 12,23,501.01

Ex:

```
select  ename,sal,to_char(sal,'00,00,000.00') from emp order by sal;
```

3) TO_CHAR(date,[char format])

It is useful to convert given data value into character format.

Date Formats: (char format values)

DD	Print digits of date
Day	Print Dayname
day	Print dayname
DAY	Print DAYNAME
Mon	Print 3-chars of month with begining char capital
MON	Print 3-chars of month in caps
mon	Print 3-chars of month in small case
month	Print full month name(in lower case)
MONTH	Print full month name(caps)
Month	Print full month name with begining char capital
yy	Print last 2 digits of year
YY	Print ''
yyyy	Print complete year number
YEAR	Year number can be spelled.

ANALYTICAL FUNCTIONS

RANK()

It will generate non sequential rank numbers based on result of window clause.

syntax

select rank() over(window clause) from table;

DENSE_RANK()

It will generate sequential rank numbers based on result of window clause.

Syntax

select col1, col2, dense_rank() OVER(window clause) from table;

Ex:

Display ename, salary and rank() and dense ranks() based on highest salary to least salary?

select ename,sal,rank() over(order by sal desc) rank,

dense_rank() over(order by sal desc) drank

from emp;

NVL(colname,value)

It will verify the given column value is null or not. If column value is null, then given value is substituted. Here column data type and given value data types should be similar.

Ex:

select sal,comm,(sal+nvl(comm,0)) total_sal from emp;

NVL2(col,'val1','val2')

It will verify given column value is null or not. If column value is not null then " value1 " is substituted. If column value is null then, " value2 " is substituted. In this function, value1 and value2 should be with same data type.

Ex:

select ename,comm,nvl2(comm,'Available','N/A') comm_status from emp;

NULLIF('val1','val2')

It will return NULL if the input values are equal. It will return val1 if both are not equal.

EX:

SQL> SELECT NULLIF('PVDINESH','PV123') FROM DUAL;

NULLIF('

PVDINESH

SQL> SELECT NULLIF('PVDINESH','PVDINESH') FROM DUAL;

N

-

DECODE (colname,val1,'str-1' ,val2,'str-2' , . . ,'DefaultValue')

It will consider column value and it is comparing with list of given values. If both are matched, then corresponding string value will be substituted. If there is no matched value, then default value is substituted.

Ex:

```
select empno,ename, deptno,decode(deptno,10,'Accounting',
                                  20,'Research',
                                  30,'Sales',
                                  40,'Operations',
                                     'Unknown'
                               ) "deptname"
from empcp  order by deptno;
```

PL/SQL

PROCEDURAL LANGUAGE / SQL

PL/SQL is a Procedural Language developed by Oracle is an extension of Oracle SQL having the functionalities of functions, control structures, and triggers.

How to define PLSQL?

PLSQL is a collection of User defined objects like Programs, procedures, Functions, Triggers, Types, Packages and so on.

PL/SQL objects are divided into 2 categories.

They are

i) Programs / Anonymous Blocks

These objects not saved in the database. These program's logic is included in the User Interface application development.

ii) Sub Programs

These objects permanently saved in the database server.

EX: Procedures and Functions

ANONYMOUS BLOCK:

STRUCTURE

DECLARE

<declaration stmts >;

BEGIN

<Assignment stmt>;

<Output stmts>;

<Data processing stmts>;

EXCEPTION

<Error handling stmts>;

END;

/ [enter] (to compile and execute program in SQL * PLUS)

NOTE: In the above program , DECLARE and EXCEPTION blocks are optional.

DECLARE block

It contains declaration statements to declare variables. Variable is a name. Variables are useful to store values temporarily. So these variables get memory space from the database engine based on their datatype and size.

syntax

varname DATATYPE(size);

Ex: v_eno int;

v_name varchar2(20);

BEGIN block

It is also known as Execution block. It contains 3 types of statements.

→**Assignment statements**

→**OutPut statements**

→**Data Processing statements**

Assignment statements

we can store values into the declared variables by using either assignment operator **:=** or SELECT query with INTO keyword.

Syntax

BY USING ASSIGNMENT OPERATOR

var **:=** value / expression / Function_calling stmt;

Ex: v_eno := 7654;

Syntax

By Using SELECT Query with INTO keyword

select col1, col2, ...,coln INTO var_1, var_2,....var-n from tablename

where <condition>;

Note:

1) In the above syntax, Number of columns and number of variables should be similar.

2) column data type and variable data type should be similar in the same sequence.

Ex: select ename,sal,hiredate INTO V_name,v_sal,v_jdate

from emp where empno=v_eno;

ii) Out Put statement:

It is Used to display text messages and variable values. Oracle provided a predefined output function as follows.

DBMS_OUTPUT.PUT_LINE ('messages' or varname);

Ex:

```
dbms_output.put_line(' employee information ');
dbms_output.put_line(' ---------- ');
dbms_output.put_line(v_eno);
```

iii) Data Processing statements:

Any sql query, select, insert, update, delete, commit, rollback, and Truncate with in begin block , is known as data processing statement.

EXCEPTION Block

It contains error handling statements to handle runtime errors.

END;

It is indicating end of a program.

/ to compile and execute a program in SQL * PLUS

NOTE: By default, any program or procedure should not display output. To display output, execute following statement.

SET SERVEROUTPUT ON;

This is valid for current session.

Ex: write a program to display welcome message .

```
begin
dbms_output.put_line('welcome to oracle pl/sql');
end;
/
```

Ex: write a program to display addition , average, max and min of 3000 and 5000?

```
DECLARE
X       INT;
```

```
Y       INT:=5000;
S       INT;
A       NUMBER(7,2);
MX      INT;
MN      INT;
BEGIN
X:=3000;
S:=X+Y;
A:=S/2;
SELECT GREATEST(X,Y) INTO MX FROM DUAL;
SELECT LEAST(X,Y) INTO MN   FROM DUAL;
DBMS_OUTPUT.PUT_LINE(' SUM=');
DBMS_OUTPUT.PUT_LINE(S);
DBMS_OUTPUT.PUT_LINE('AVERAGE=');
DBMS_OUTPUT.PUT_LINE(A);
DBMS_OUTPUT.PUT_LINE('HIGHER VALUE=');
DBMS_OUTPUT.PUT_LINE(MX);
DBMS_OUTPUT.PUT_LINE('Least VALUE=');
DBMS_OUTPUT.PUT_LINE(MN);
END;
/
```

Types of programs: 2

1) Static program:

It will not accept input values, at run time.

2) Dynamic program:

In this case the program can accept runtime input values. we can make a program as Dynamic Program by using "&" (Substitution Operator).

Syntax

Varname **:=** '&varname';

Ex: As per above syntax, at runtime it will prompt for variable value as follows.

Enter value for varname: <value>

STATIC PROGRAMS:

Ex: write a program to display the employee information of empID 7654?

```
declare
veno        int:=7654;
vname      varchar2(20);
vsal         number(5);
vjob        varchar2(20);
vhiredate date;
vcomm      number(4);
vdeptno    number(3);
BEGIN
select ename,sal,job,hiredate,comm,deptno
                                INTO
        vname,vsal,vjob,vhiredate,vcomm,vdeptno
from emp
where empno=veno;
dbms_output.put_line(' Info of 7654');
dbms_output.put_line('---------------');
dbms_output.put_line(vname);
dbms_output.put_line(vsal);
dbms_output.put_line(vjob);
dbms_output.put_line(vhiredate);
dbms_output.put_line(vcomm);
dbms_output.put_line(vdeptno);
END;
```

Ex:

Write a program to display employee name, salary, designation, joindate, commission, and deptno of empid 7654?

```
declare
veno int:=7654;
vname varchar2(20);
vsal number(5);
vjob varchar2(20);
vhiredate date;
vcomm number(4);
vdeptno number(3);
BEGIN
select ename,sal,job,hiredate,comm,deptno INTO
        vname,vsal,vjob,vhiredate,vcomm,vdeptno
from emp
where empno=veno;
dbms_output.put_line
(' Info of Emp id:      7654');
dbms_output.put_line
('========================================');
dbms_output.put_line('Name of emp: '||vname);
dbms_output.put_line('Salary of emp: '||vsal);
dbms_output.put_line('Designition of emp: '||vjob);
dbms_output.put_line('Join date of emp: '||vhiredate);
dbms_output.put_line('Commission of emp: '||vcomm);
dbms_output.put_line('Deptno of emp: '||vdeptno);
dbms_output.put_line('========================================
====');
END;
/
```

output:

Info of 7654

==========

Name of emp: MARTIN

Salary of emp: 1250

Designition of emp: SALESMAN

Join date of emp: 28-SEP-81

Commission of emp: 1400

Deptno of emp: 30

==========

PL/SQL procedure successfully completed.

Ex:

Write a program to display number of male customers and number of female customers?

```
declare
male_cnt      int;
female_cnt    int;
begin
select count(*) into male_cnt from cust_dtls where gender='M';
select count(*) into female_cnt from cust_dtls where gender='F';
dbms_output.put_line(' Number of males= '||male_cnt);
dbms_output.put_line(' Number of Females='||female_cnt);
end;
```

Assignments:

1) Write a program to display designition of empid 7788?

2) Write a program to display the city and mobile number of customer id " cust-5"?

3) Write a program to display the location of department " RESEARCH "?

Dynamic Progarms:

By using & (Substitution operator) operator we will make a program as a dynamic program. Instead of assigning a constant value in to a variable, While the execution of the program, the program has to take a value from the end user and that value stored it into a variable.

Varname **:=** '&varname';

Ex:

write a program to display the details of employee for the given empno?

```
declare
v_eno number(4);
v_ename varchar2(20);
v_sal number(5);
v_job varchar2(20);
v_jdate date;
v_comm  number(5);
V_dno int;
BEGIN
v_eno:='&v_eno';
select ename,sal,job,hiredate,nvl(comm,0),deptno
        INTO
        v_ename,v_sal,v_job,v_jdate,v_comm,v_dno
from emp      where empno=v_eno;
```

```
dbms_output.put_line
(chr(10)||' Details of emp id:      '||v_eno||chr(10)||
'----------------------------------------'||chr(10)||
'Name:      '||v_ename||chr(10)||
'Working with  Salary:     '||v_sal||chr(10)||
'Working   as  :       '||v_job||chr(10)||
'Joined   on:    '||v_jdate||chr(10)||
'Getting  Commission:    '||v_comm||chr(10)||
'Working under:    '||v_dno
);
end;
```

Ex:

Write a program to display the number of emps in the given deptno?

```
declare
vdno        number(2);
e_count    int;
begin
vdno:='&vdno';
select count(empno) into e_count
from emp
where deptno=vdno;
dbms_output.put_line
(' number of emps in Department : '||vdno||'   = '||e_count);
end;
```

output:

number of emps in 20 = 5

Assignment:

1) write a program to display the total salary i am paying to deptno 30 employees?

2) write a program to display the "number of male customers" from the given city?

3) Write a program to display the number of emps working under given deptno?

4) write a program to find the number of emps working with given designition?

5) Write a program to find and display total salary paying to given dept name?

VARIABLE TYPES

In PLSQL, we have 3 types of variables.

They are

a) Scalar Variable

b) Composite Variable

c) Collection Type Variable

i) Scalar variable

It is able to store one value at a time. We can declare scalar variables using 2 methods.

a) var datatype(size);

In this method of declaration, sometimes we will get size and datatype incompatibility issues. These issues are eliminated as follows.

b)

%TYPE

we can declare any scalar variable with column datatype. By this method, we will not get any data type and size related errors in future.

syntax

var TableName.ColumnName**%TYPE**;

Ex:

vdno emp.deptno%TYPE;

ii)Composite variable

A variable which is able to store one record at a time. It can decrease number of variable declarations. It is decreasing length of program and execution time.

Composite variables are 2 types.

a) Store a record from single table. [**%ROWTYPE**]

b) Store a record from multiple tables. [**TYPE**]

%ROWTYPE

It is used to declare a variable as a RECORD type variable. It will store one record from one table.

ADVANTAGE: It is decreasing number of variable declarations.

syntax

varname tblname%ROWTYPE;

Ex:

vemprec emp%rowtype;

SAVE A RECORD IN TO COMPOSITE VARIABLE

Ex:

select * into vemprec from emp where empno=7788;

ACCESS values from COMPOSITE variable

Syntax In any output statement, use below syntax

var_name **.** colname;

Ex: dbms_output.put_line (' emp salary :'|| emp_rec **.** sal);

Ex:

Write a program to display the information of employee for the given employee id?

```
declare
--Dynamic declaration of variable
veno   emp.empno%type;
--Declaring table based record type variable
```

```
  e_rec    emp%rowtype;
  begin
  veno:='&veno';
  select * into e_rec from emp
  where empno=veno;
  dbms_output.put_line
  (chr(10)||' Emp id:  '||veno||chr(10)||
 '*****************************'||chr(10)||
 'Name: '||e_rec.ename||chr(10)||
 'Desg: '||e_rec.job||chr(10)||
 'Salary: '||e_rec.sal||chr(10)||
 'Join Dt: '||e_rec.hiredate||chr(10)||
 'Comm:   '||e_rec.comm||chr(10)||
 'Deptno:  '||e_rec.deptno||chr(10)||
 '*****************************'
);
end;
```

Ex: write a program to display the information of product for the given product id?

```
  declare
  vpid   prod_dtls.prod_code%type;
  p_rec  prod_dtls%rowtype;

  begin
  vpid:='&vpid';
  select * into p_rec
  from prod_dtls
  where prod_code=vpid;
  dbms_output.put_line
  (chr(10)||' Information of prodid: '||vpid||chr(10)||
```

```
p_rec.prod_name||chr(10)||
p_rec.cost||chr(10)||
p_rec.mfg||chr(10)||
p_rec.warrenty||chr(10)||
p_rec.comp_code
   );
end;
```

SUB PROGRAMS

Database programs which are saved under database server permanently.

Sub programs are 2 types.

They are

1) PROCEDURES

2) FUNCTIONS

STORED PROCEDURES (or) PL/SQL PROCEDURES

A named pl/sql program which is stored / created under the data base is known stored procedure. It is also known as a subprogram to perform a task or set of tasks.

Any PROCEDURE has 2 PARTS.

i) Specification

It contains object type, object name, and argument variables. Argument is a variable , that is declared under specification part.

Note: By default arguments are useful to take input values.

ii) Body

It contains declarations, begin block, exception block, and end of procedure.

syntax:

```
/*Procedure specification*/
create or replace  PROCEDURE <proc_name>
[(arg_1 MODE  datatype, arg_2 MODE datatype,.....)]
IS / AS
```

/* Procedure body */

<declaration stmts>;

BEGIN

EXCEPTION

--

--

END <proc_name>;

/ (to compile the procedure in SQL * PLUS)

Procedure created.

HOW TO EXECUTE A PROCEDURE?

We can execute a procedure in 2 methods.

1) By calling procedure

We can make a call to the procedure from program, procedure, function, package, and trigger.

NOTE: Procedure never called using SELECT Query.

Syntax

proc_name[(Argval1,Argval2,....)];

2) By using EXECUTE command

we can execute any procedure explicitly by using ***"EXECUTE"*** command, as follows,

EXECUTE <proc_name>[(arg_val1, arg_val2,.....)];

or

EXEC <proc_name>[(arg_val1, arg_val2,.....)];

Arguments

It is known as a variable to receive runtime input value or to return output value.

Arguments are 3 types

IN

OUT

IN OUT

By default, arguments are IN type arguments, that is receive input value.

Ex: write a procedure to display customer 1 account details?

```
CREATE PROCEDURE PROC_CUST_ACT
IS
cust_rec  cust_act_dtls%rowtype;
begin
select * into cust_rec from cust_act_dtls where cno='cust-1';
dbms_output.put_line
(chr(10)||
'Actno: '||cust_rec.actno||chr(10)||
'Act_type: '||cust_rec.act_type||chr(10)||
'Open Dt: '||cust_rec.act_open_date||chr(10)||
'Balance: '||cust_rec.act_bal
);
end proc_cust_act;
/ ( To Compile Procedure )
```

Ex:

Write a procedure to find number of customers from the city "Delhi"?

```
create or replace procedure proc_cust_cnt_delhi
is
vcnt int;
begin
select count(*) into vcnt
from cust_dtls where city='Delhi';
dbms_output.put_line
(chr(10)||
```

```
'City: Delhi'||chr(10)||
'Customer Count: '||vcnt
);
end;
```

How do i execute a procedure implicitly?

By making a call to the procedure we can implicitly execute the procedure.

→Calling a procedure from any program

```
BEGIN
proc_ecount_30; --Procedure calling stmt
dbms_output.put_line(' calling procedure is finished');
dbms_output.put_line(' Program Execution is finished');
END;
```

DYNAMIC PROCEDURES

WRITING THE PROCEDURE TO DISPLAY THE NUMBER OF EMPS IN THE GIVEN DEPTNO?

```
CREATE OR REPLACE PROCEDURE PROC_empcnt_did
(VDNO  EMP.DEPTNO%TYPE)
IS
CNT INT;
BEGIN
SELECT COUNT(*) INTO CNT
FROM EMP
WHERE DEPTNO=VDNO;
DBMS_OUTPUT.PUT_LINE
(chr(10)||' NUMBER OF EMPS IN DEPT :  '||VDNO||':   ARE : '||CNT);
END PROC_empcnt_did;
```

SAMPLE EXECUTIONS:

```
EXEC PROC_2(10);
EXEC PROC_2(20);
EXEC PROC_2(30);
```

EXCEPTIONS

Exception is a PLSQL runtime error. It is handled by programmer to display user friendly error message.

Default oracle error format is

ERROR:

ORA-xxxxxx: <message>

<error code>: <error message >

SQLCODE SQLERRM

Exceptions are raised under begin block.

Exceptions are handled under Exception Block.

Each exception will be handled with one " when clause ", under exception block.

WHEN <excep_handler> THEN

-- Do This ----

Exceptions are 2 types.

a) Pre defined exceptions

b) User defined exceptions

i) Pre defined/system defined/ Named Exceptions

These can be managed by data base engine as follows.

--While execution if there exists any error then it will look for exception block.

--With in that it will look for corrsponding exception handler.

--Then it executes the statements within the exception handler.

Exception handlers / Pre defined Exception names

i) TOO_MANY_ROWS

It will be raised when the select statement is selecting data from multiple records.

ii) NO_DATA_FOUND

It will be raised if select query not selecting data from the table.

iii) ZERO_DIVIDE

It will be raised if a value is divided by zero.

iv) CURSOR_ALREADY_OPEN

It will be raised if we are trying to open a cursor which is already opened.

v) VALUE_ERROR

It will be raised if there exist datatype or size mismatch between variable and value.

vi) CASE_NOT_FOUND

It will be raised if case structure don't has matched case and and don't has ELSE part.

EXAMPLE PROCEDURES

Ex: Write a procedure to display employee name and job for the given salary?

```
CREATE OR REPLACE PROCEDURE PROC_no_EXCEP (VSAL  NUMBER)
IS
VNAME EMP.ENAME%TYPE;
VJOB  EMP.JOB%TYPE;
BEGIN
SELECT ename,job INTO vname,vjob from emp where sal=vsal;
dbms_output.put_line('Name of emp:  '||vname||';  Desg of emp:  '||vjob);
dbms_output.put_line(' End of procedure ');
end proc_no_excep;

SET SERVEROUTPUT ON;

EXEC PROC_NO_EXCEP(5000);
EXEC PROC_NO_EXCEP(800);
EXEC PROC_NO_EXCEP(10000);
```

Ex: Above procedure with exceptions

```
CREATE OR REPLACE PROCEDURE PROC_EXCEP
(VSAL IN  NUMBER)
IS
VNAME EMP.ENAME%TYPE;
VJOB  EMP.JOB%TYPE;
BEGIN
SELECT ename,job INTO vname,vjob
from emp
where sal=vsal;
dbms_output.put_line
(CHR(10)||
'Name of emp:  '||vname||CHR(10)||
'Job of emp:   '||vjob
);
dbms_output.put_line('<---- End of procedure ----> ');
EXCEPTION
WHEN TOO_MANY_ROWS THEN
DBMS_OUTPUT.PUT_LINE(CHR(10)||' SAL IS DUPLICATED');
DBMS_OUTPUT.PUT_LINE
(CHR(10)||' selecting data from multiple records
is not yet possible without using cursors');
WHEN NO_DATA_FOUND THEN
DBMS_OUTPUT.PUT_LINE
(CHR(10)||'No emp Getting the salary'||vsal);
end proc_excep;
```

EXEC PROC_EXCEP(800);----Only one employee is getting sal-display output

EXEC PROC_EXCEP(8000);---No emp getting this salary , so raise an error no_data_found

EXEC PROC_EXCEP(1250);---Multiple emps getting the salary, so raise an error Too_many_rows.

Ex:

write a procedure to display employee details who is

working with given designition?

```
create or replace procedure proc_emp_dtls_desg
(vjob   IN emp.job%type)
is
vrec    emp%rowtype;
begin
select * into vrec from emp where job=vjob;
dbms_output.put_line
(' The given job is '||vjob);
dbms_output.put_line
(vrec.empno||' '||vrec.ename||' '||vrec.job||' '||vrec.sal||
'  '||vrec.hiredate||' '||vrec.comm||' '||vrec.deptno);
EXCEPTION
WHEN TOO_MANY_ROWS THEN
 DBMS_OUTPUT.PUT_LINE
 (' MULTIPLE EMPS WORKING AS:   '||VJOB);
 DBMS_OUTPUT.PUT_LINE
 (' I can display if there exists 1 emp with the given job,
 Otherwise i am        unable to display multiple records');
WHEN NO_DATA_FOUND THEN
```

```
DBMS_OUTPUT.PUT_LINE
('No EMPS WORKKING AS:   '||VJOB);
end proc_emp_dtls_desg;
```

sample outputs:

anonymous block completed

MULTIPLE EMPS WORKING LIKE SALESMAN

PROCEDURE PROC_EMP_DTLS_DESG compiled

anonymous block completed

MULTIPLE EMPS WORKING LIKE SALESMAN

I can display if there exists 1 emp with the given job, Otherwise i am unable to display multiple records

anonymous block completed

The given job is PRESIDENT

7839 KING PRESIDENT 5000 17-NOV-81 2000 10

2) User defined exceptions

Here Programmer has to manage the exceptions. Programmer has to RAISE the exception if he is expecting any error. The exception is to raised by using exception variable name. The exception variable is acts as an EXCEPTION HANDLER in exception block.

i)

Declare exception type of variable in declare block.

This variable is known as exception handler, since we can handle one error with this name.

syntax: var exception;

Ex: comm_miss exception;

ii) Programmer can Raise the exception by using RAISE stmt (or)

Raise_Application_Error();

syntax: RAISE <excep_var>;

Ex: RAISE COMM_MISS;

iii) Define exception, under exception block.

syntax: WHEN <excep_var> THEN

stmts;

stmts;

---

ex:

```
when comm_miss then
dbms_output.put_line(' comm value  is null');
```

Ex: write a procedure to display commission of employee of given employee id?
If commission is null, then display any user friendly error message?

```
create or replace procedure proc_emp_comm
(veno IN  emp.empno%type)
IS
vcomm    emp.comm%type;
comm_miss    exception;
BEGIN
select comm into vcomm
from emp
where empno=veno;
if vcomm is null then
RAISE comm_miss;
else
dbms_output.put_line
```

```
(chr(10)||' comm of  '||veno||'  is  :'||vcomm);
end if;
EXCEPTION
when no_data_found then
dbms_output.put_line
(chr(10)||' No employee with given id:  '||veno);
when Too_many_rows then
dbms_output.put_line
(chr(10)||' Duplicate empid existed');
when comm_miss then
dbms_output.put_line
(chr(10)||' Employee  not getting commission ');
end proc_emp_comm;
```

sample output:

PROCEDURE PROC_EMP_COMM compiled

EX:

EXEC PROC_EMP_COMM(7654);

anonymous block completed

comm of 7654 is 1400

anonymous block completed

comm of 7788 is 2000

anonymous block completed

comm of employee is null

anonymous block completed

No employee with given id

FUNCTIONS

It is database object Like Procedure. It is also known as a sub program. By default, It should return a value to the calling object. In pl/sql the functions are known as User Defined Functions (UDF). Generally, functions are created based on common logic.

PLSQL functions are always executed **by calling function.**

Syntax:

var:=func_name(arg_val_1, arg_val_2,.....);

Function has 2 parts. That is Specification and Body.

syntax:

/*Spec*/

create or replace function <func_name> (argvar datatype, argvar datatype.....)

RETURN <value_datatype>

IS

/*body*/

<declaration stmts>;

begin

RETURN(value/var_name);

END <func_name>;

/ [to compile function in SQL * PLUS]

→How to execute any function ?

→ By writing function calling stmt in any object.

var := func_name(val1,. . .);

Ex:

Create a sample function to add 2 given integers?

create or replace function f_add(x IN int, y IN int)

```
return   number
         is
         Z  int;
         begin
         Z:=x+y;
         return(Z);
         end f_add;
```

Note:

A Function can be called through any SELECT or PROGRAM or Procedure

or packages.

Ex: select f_add(100,200) from dual;

300

Ex: write a procedure to display empid, sal, bonus and total salary (sal+ bonus) for the given employee id?

Ex: Write a procedure to display above details for all employees?

Ex: Write a procedure to display above details for the given job category of emps?

[Common Functionality is " Finding Bonus Based on salary Range"]

Bonus is to be calculated as follows

sal<1000	2%
>=1000 & < 2000	5%
>=2000 & < 3000	10%
>=3000 & < 5000	20%
>=5000	25%

Creating a function to calculate the Bonus?

```
create or  replace function func_bonus (s IN emp.sal%type)
         return number
         is
         vbonus    number(7,2);
         begin
         if (s <1000) then
```

```
vbonus:=0.02*s;
elsif (s>=1000 and s<2000) then
vbonus:=0.05*s;
elsif (s>=2000 and s<3000) then
vbonus:=0.10*s;
elsif (s>=3000 and s<5000) then
vbonus:=0.20*s;
else
vbonus:=0.25*s;
end if;
return(vbonus);
end func_bonus;
/
```

```
--Procedure to display empid, salary, bonus and total salary for
--given empid?

create or replace procedure proc_emp_fsal
(veno IN emp.empno%type)
is
vsal emp.sal%type;
b    number(7,2);
Tsal number(7,2);
begin
select sal into vsal
from emp
where empno=veno;
------>function calling stmt
b:=func_bonus(vsal);
tsal := vsal+b;
```

```
        dbms_output.put_line
        (chr(10)||
        ' The salary details of:  '||veno||chr(10)||
        '-------------------------------------------------'||chr(10)||
        'EMPID     salary    bonus    Total salary'||chr(10)||
        '-------------------------------------------------'||chr(10)||
        veno||'    '||  vsal||'     '||b||'     '||tsal
        );
        end proc_emp_fsal;
        /
--Procedure to display empid, salary, bonus and total salary for  all emps?

create or replace procedure proc_emps_fsal
        is
        CURSOR c  is   select empno,sal from emp;
        veno emp.empno%type;
        vsal emp.sal%type;
        b    number(7,2);
        t    sal number(7,2);
        BEGIN
        OPEN c;
        dbms_output.put_line
        (chr(10)||'empid    salary    bonus      final salary'||chr(10)||
                '-----------------------------------------------------------');
        LOOP
        FETCH c into veno,vsal;
        b := func_bonus(vsal); /* function calling stmt */
        tsal := vsal+b;
        EXIT WHEN (c%notfound);
        dbms_output.put_line
```

```
        (veno ||'      '||vsal||'      '||b||'      '||tsal);
        END LOOP;
        CLOSE c;
        end proc_emps_fsal;
        /
```

-->write a procedure to display empid, sal, bonus and total salary for all emps
--> under given job category?

```
create or replace procedure proc_emps_fsal_job
        (vjob  IN emp.job%type)
        is
        cursor c is select empno,sal from emp where job=vjob;
        veno emp.empno%type;
        vsal emp.sal%type;
        b    number(7,2);
        tsal number(7,2);
        begin
        open c;
        dbms_output.put_line
        ('Given Job Category :          '||vjob);
        dbms_output.put_line
        ('veno      salary      bonus      final salary');
        dbms_output.put_line
        ('-----------------------------------------------------');
        loop
        fetch c into veno,vsal;
        b:=func_bonus(vsal); /* function calling stmt */
        tsal := vsal+b;
        exit when (c%notfound);
```

```
dbms_output.put_line
(veno ||'     '||vsal||'     '||b||'     '||tsal);
end loop;
close c;
end proc_emps_fsal_job;
/
```

How do i delete function?

drop function <func_name>;

How to display list of function names?

select object_name from USER_PROCEDURES;

PACKAGES

Package is a data base object. It is useful to group related objects logically. It will reduce search time for required procedure or function. Packages also reduce disk I/O Operations , by loading all objects into buffer area.

Package has 2 parts.

→ Specification

→ Body

These parts are created separately.

1) package specification:

It contains procedure and function calling stmts and also variables. The variables declared in the package are known as Global Variables, Since these variables are accessible to all procedures and functions within the package.

syn-1: package specification

```
create or replace package <pkg_name>
AS
<variable declarations>;
PROCEDURE <procedure calling stmts>;
FUNCTION   <function calling stmts> RETURN <datatype>;
end <pkg_name>;
/ [ to compile package ]
```

2) package body

It contains procedure bodies and function bodies.

```
create or replace package Body <pkg_name>
        IS

Procedure  <proc-1-name> (.....)
is
        --
        --
end <proc-1>;

Procedure  <proc-2-name> (.....)

is
        --
        --
end <proc-2>;

Function  <func-1>(.....)
return <datatype>
is
        --
        --
end <func-1>;

end <pkg_name>;
/ [ to compile package body ]
```

Ex:

write a package which contains the procedures and functions to calculate sal, bonus, final salaries for given empid , for all emps and for given job category employees?

```
CREATE OR REPLACE PACKAGE emppkg
 AS
PROCEDURE PROC_EMP_FSAL
(VENO EMP.EMPNO%TYPE);
PROCEDURE PROC_EMPs_Fsal;
PROCEDURE PROC_EMPs_FSAL_JOB
(vjob emp.job%type);
FUNCTION  FUNC_bonus(S EMP.SAL%TYPE)
RETURN NUMBER;
END emppkg;

/
```

EX: package body

```
CREATE OR REPLACE PACKAGE BODY emppkg
IS
/* PROC_EMP_FSAL  */
Procedure proc_emp_fsal(veno emp.empno%type)
is
vsal emp.sal%type;
b number(7,2);
fsal number(7,2);
begin
select sal into vsal from emp where empno=veno;
```

```
b:=func_bonus(vsal); /* function calling stmt */
fsal:=vsal+b;
dbms_output.put_line(' The salary details of  '||veno);
dbms_output.put_line('-------------------------------------------------');
dbms_output.put_line('salary       bonus        final salary');
dbms_output.put_line(  vsal||'       '||b||'        '||fsal);
end proc_emp_fsal;



/* function to calcualte the BONUS */

Function func_bonus(s emp.sal%type)
return number
is
bonus number(7,2);
begin
if (s<1000) then
bonus:=0.02*s;
end if;
if (s>=1000 and s<2000) then
bonus:=0.05*s;
end if;
if (s>=2000 and s<3000) then
bonus:=0.10*s;
end if;
if (s>=3000 and s<5000) then
bonus:=0.20*s;
end if;
if (s>=5000) then
bonus:=0.25*s;
```

```
        end if;
        return(bonus);
        end func_bonus;

        /* Procedure to display the sal info of all emps */

        Procedure proc_emps_fsal
        is
        cursor c is select empno,sal from emp;
        veno emp.empno%type;
        vsal emp.sal%type;
        b number(7,2);
        fsal number(7,2);
        begin
        open c;

        dbms_output.put_line
('-------------------------------------------------');
        dbms_output.put_line
('salary       bonus       final salary');
        dbms_output.put_line
('*************************************************');
        loop
        fetch c into veno,vsal;
        dbms_output.put_line(' The salary details of  '||veno);
        b:=func_bonus(vsal); /* function calling stmt */
        fsal:=vsal+b;
        EXIT WHEN (C%NOTFOUND);
        dbms_output.put_line(vsal||'       '||b||'         '||fsal);
        END LOOP;
```

```
        CLOSE C;
        end proc_emps_fsal;

/* procedure  proc_emps_fsal_job  */
        procedure proc_emps_fsal_job(vjob emp.job%type)
        is
        cursor c is select empno,sal from emp where job=vjob;
        vsal emp.sal%type;
        veno emp.empno%type;
        incrval number(7,2);
        fsal number(7,2);
        begin
        open c;
        loop
        fetch c into veno,vsal;
        incrval:=func_bonus(vsal);  /* function calling stmt  */
        fsal:=vsal+incrval;
        EXIT WHEN c%NOTFOUND;
        dbms_output.put_line(' sal,incrval,final salary of  '||veno);
        dbms_output.put_line(vsal||' , '||incrval||' , '||fsal);
        end loop;
        close c;
        end proc_emps_fsal_job;

        END emppkg;

        /
```

HOW TO EXECUTE A PROCEDURE FROM A PACKAGE?

syn: exec pkg_name.proc_name(argval, argval.....);

Ex: exec emppkg.proc_emp_fsal(7654);

How to call function from a package?

select pkg.func_name() from dual;

How to delete a package?

drop package <pkg_name>;

IQ:

What is a Local Procedure ?

If you created any procedure inside the package,

then it is known as local procedure.

This procedure not available outside the package.

TRIGGERS

TRIGGER is a data base object like a procedure to perform back ground task.

Triggers are executed automatically without EXEC statement or calling statement.

Triggers are defined at 3 levels.

1) DDL level triggers

Invoked and executed automatically for any DDL command.

2) DML level triggers

Invoked and executed automatically for any DML command.

3) SESSION level triggers

Invoked and executed automatically for any session level command.

DML Level triggers:

These are responsible to maintain uniform business data and to maintain audit information of table data.

DML triggers are defined at 2 levels.

1) Row Level trigger

It is executed once for each record affected by dml stmt.

It will be defined with the stmt " FOR EACH ROW ".

2) Statement Level Trigger

It is executed only once for each dml stmt

irrespective of number of affected records.

Pseudo record Functions:

:NEW.colname

It can access the new value from the column.

[insert & After update]

:OLD.colname

It can access old value from the column.

[Before update & before delete]

Trigger has 3 parts:

i) Trigger Event:- Reason for trigger execution.

ii) Trigger Restriction:- It will control trigger execution, that is before or after trigger event.

iii) Trigger Action:- It is the task of trigger.

syntax:

```
create or replace trigger <trig_name>
[before / after]
[ insert / update / delete ]
ON  <table name>
[FOR EACH ROW]
declare
------
```

```
------
BEGIN
------------
------------
END <trig_name>;
/  [ to compile the trigger in sql * plus ]
trigger created.
```

ex: table

```
create table employee
(
ename  varchar2(20),
city  varchar2(20)
);

insert into employee values('Kiran','Delhi');
insert into employee values('madhu','Delhi');
insert into employee values('DINESH','HYD');
insert into employee values('smith','texas');
insert into employee values('jack','delhi');
insert into employee values('soumya','DELHI');

select * from employee;

ENAME                CITY
-------------------- --------------------
Kiran                Delhi
madhu                Delhi
DINESH               HYD
smith                texas
```

Ex:

write a trigger to insert or update the data of employee in upper case?

[Generally to maintain uniform data in the business Database]

```
create or replace trigger trig_conversion
before
insert or update  on employee
for each row
begin
:new.ename:=upper(:new.ename);
:new.city:=upper(:new.city);
end trig_conversion;

/
```

How to view list of Trigger Names?

Select trigger_name from user_triggers;

How to delete any trigger?

Drop trigger <trigger_name>;

How to alter Trigger?

Alter trigger <trig_name> ENABLE / DISABLE;