ORACLE11g

New Notes

By

MURALI
Naresh technologies

SRI RAGHAVENDRA XEROX

Software Languages Material Available

Beside Bangalore Ayyangar Bakery, Opp. C DAC, Ameerpet, Hyderabad.

Cell: 9951596199

Maa ?

Oracle 11g

20th/01/14 (4.00pm)

Rs = 150/-

- Oracle is a database server s/w.
  - EX:- SQLServer DB2 oracle sybase Ingnes MySQL Teradata SQLite.
- All are implemented by ORDBMS (~~oracle~~ Object Relational DBMS)
- To store data permenantly in Secondary storage device.

1) Data.
2) Information.
3) Datastore.
4) Database.
5) DBMS
6) RDBMS.

Data:- collection of Rawfacts (datatypes int, char, float)

Information:-

N.B

- Oracle is a Product from oracle cooperation.
- Oracle is a database Server which is used to store data permimently in secondary storage device.
- Oracle is a object Relational DataBase Mgmt system.
- Oracle is a DB server some language are used i.e
  1) SQL (only maintain the data).
  2) PL/SQL (Procedure language SQL) to control the server.
  3) Dynamic SQL.

~~Data:-~~ • All organization stores some type of data.

Data:-

It is nothing but collection of Rawfacts.

EX:- 1) Student marks.
2) customer names.

Informations:-

when we are tracking data that we are

easyly meaningful ~~preserves~~ result this is called as Information.

EX:- 1) Student marksheet.
2) invoice of a customer.

Data
101 soma 2000
102 Patel 3000

Information

sno	name	sal
101	soma	2000
102	Patel	3000

Data store :-
It is a place where we can store data.

EX:- 1) Books & paper.
2) Flat files.
3) database.

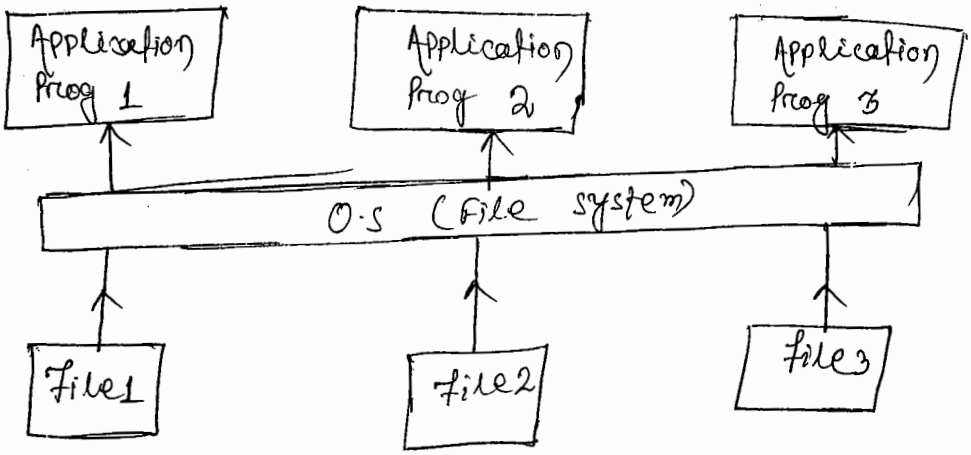
2) Flat Files :-
This is a Traditional mechanism which is used to store data or information in individual ~~file~~ unrelated files.

Disadvantages :-
* Data retrival.
* Data redudancy.
* Data integrity.
* Data security.
* Data indexing.

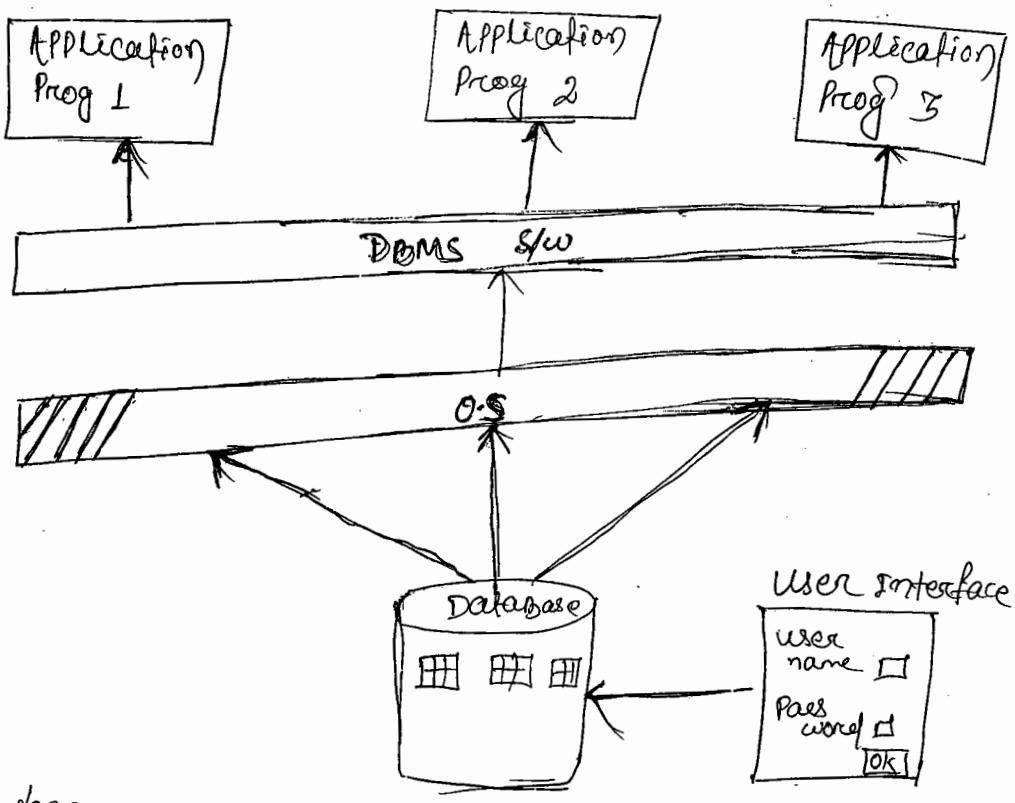
* Data retrival :-
if we want to retrive data from Flat Files that be must a develop applications Programs in Highlevel languages. where as once a data store in databases that we are retrive a data using ~~standard~~ SEQUEL ~~Query~~ language (SQL).

• Flat file mechanism every application Program make a maintain it's own file seperate from other application.

<FLAT File Mechanism>



when we use the DBMS s/w.



- when ~~Data~~ we are installing DBMS s/w automatically some places creat~~ative~~ in disk, this is called physical Database and also ~~called~~ automatically an user interfaces is created. Through this user interface directly we are interacting with the database otherwise through app Prog indirectly we are interactive with the DB.

Mar ? (24.00 Am) 21th/01/14

Data redundancy :-

→ Sometimes we r maintaining multiple copy copy same data in different locations.

→ This Data is also called as Redundant Data or Duplicate Data, in flat file mechanism where we r modifying data in one location, it is not effected in another location. This is called inconsistancy.

→ Databases automatically maintain consistance data through ACID Properties. i.e in databases every transaction internally heaving ACID Properties.

→ In databases if we want to reduce redundancy then we are using Normalization Process.

Data integrity :-

→ Integrity means to maintain Proper data in databases, we are maintaing Proper data through Constraints, triggers.

→ if we want to maintan Proper data in flat files we must a develops application Programs in a Highlevel languages.

Data security :-

→ Data stored in flat files can not be secured becoz flat files does not Provides security mechanism. where as Databases Provides Role based security.

Data indexing :-

→ If we want to retrive data very firstly that we are using indexing mechanism in databases where as flat files does not Provides indexing mechanism.

Mag

(4.00pm) 22th/01/14

Organizations suffering from some flat file mechanism to store data.

→ To overcome this Problems 1960 onwards org. useges a specialised s/w to store and manage data efficently, these s/w are called as DBMS s/ws.

DBMS :- (Database Management system)

→ It is nothing but a collection of Programs (s/w) written to manage database.

→ whenever we r installing DBMS s/w automatically some place is created in H.disk This is called as Storage area.

→ In this storage area we are storing Data.

→ Database :-

→ it is an org collection of interrelated data used by an application Programs in a org.

DBMS Architecture

ANSI → American National standart Inst has establish 3 level architecture for DBMS.

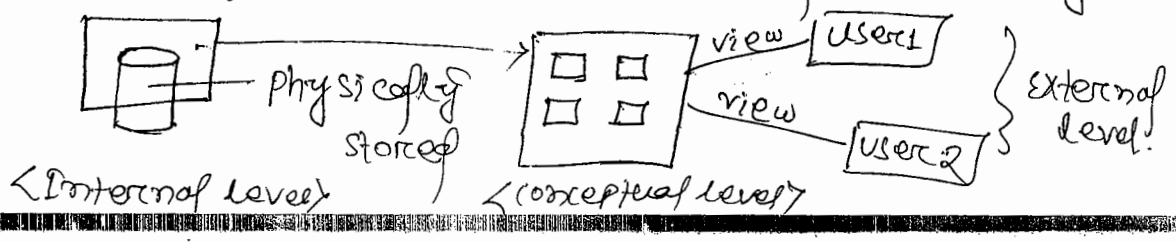
→ This Architecture is also called as ansi/sparc architecture.

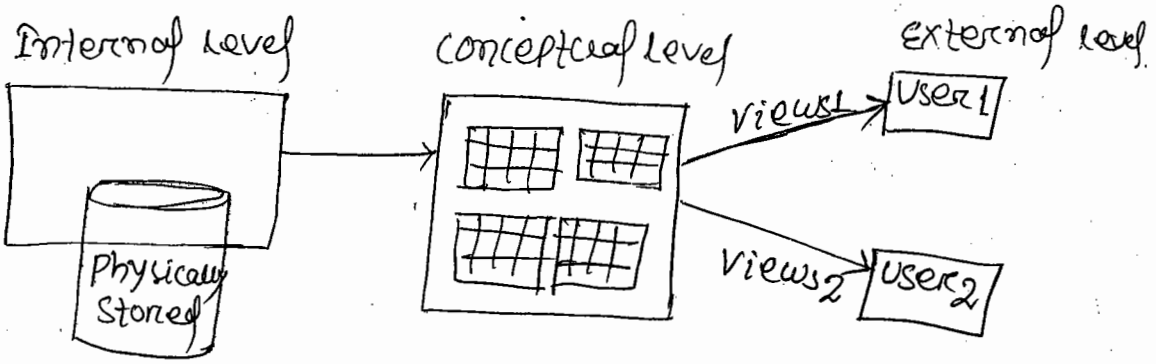
→ Object of DBMS Arch. is to seperate users view of the database from the way physically it is stored.

→ 3 levels are :-

1) External level.
2) conceptual level.
3) internal level.

internal Level :- How data is stored internally.





(Architecture of 3 level)

Conceptual level :-

→ conceptual level describes enterprise View of the Database.

E.g :- SQL > create table bank(accno number (10), name varchar2 (10), balance number (10);

→ Conceptual view designed by either developers or Database Administrator. (DBA)

Internal Level :-

eg :- accno byte (10) offset name byte (10) offset

→ Internal level describes How data physically stored in Database.

External level :-

→ This level describes individual users access a data becoz individual users access personaly database.

Data Model :-

→ How data layed out at conceptual level defined by data models.

→ Based on the database designed 3 models have been used. i.e :-

1) Hierarchical datamodel.
2) Network datamodel.
3) Relational datamodel.

3) Relational Data Model :-

Primary key

Dept

Deptno	Dname	LOC
10		
20		
30		
40		

(Master table)

Emp

Empno	Ename	. . .	Deptno
			20
			30
			30
			20
			30
			30

(Child table)

→ whenever a table column are does not have duplicate data values those type of data are called as Primary.

→ whenever the table column data are stored in a children table.

→ Master table have main transaction and child table have repeated data.

→ if we can delete from master table row ~~th~~ it is impossible becoz it exit in child table repeated data are exit.

→ when we delete from repeated data from child table then we can delete row from master table.

→ To overcome this Problem we go to on delete cascade class. ~~then~~ so ~~when the master table row~~

→ 2 rules are mandatory

1) 1st delete child table data then possible to delete Master data.

2) when we insert data that data must available in master table.

1) Hierarchical Data Model :-

Ex:-

Primary key

Empno	Empname	Dept
1001	Murli	S/w
1002	abc	HR
1003	abc	Admin

(Master table)

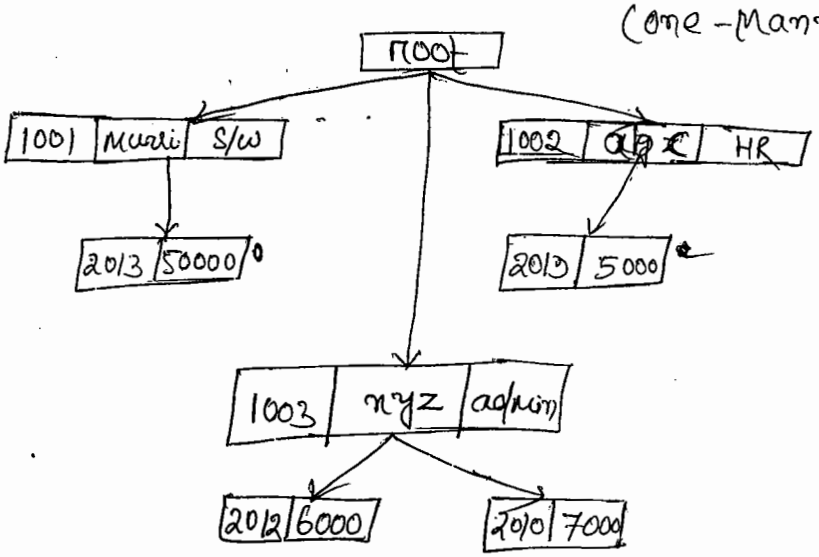
Loan Details.

foreign key

Empno	Year	Amount
1001	2013	50,000
1002	2013	5000
1003	2012	6000
1003	2010	7000

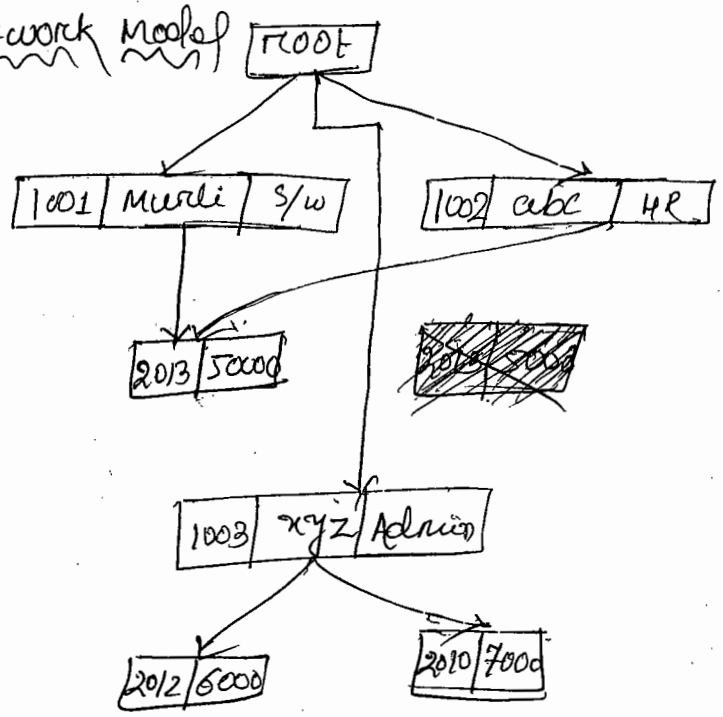
(Child table)

(one-Many) in hyrarchical.



→ In Hy. model Data are repeated. so more Reducdant occuring here.

2) Network Model



(Networking model)

→ In networking model Data are not repeated so that no redundant occuring here.

→ it is many-Many relationship.

Mag (4.00 PM) 23th/01/14

1) Hierarchical Data MODel:-

→ In Hierarchical data model organizes data in tree like structure.

→ This hierarche also called as Parent child hirarche

→ This model implemented based on one to many Relationships. i.e Many child records heaving single Parent. thats why in this structure all child data segments are repeated. thats why this structure maintains more duplicate data.

→ In hirarchical Model data stored in a format of records and also record typed is corresponding to the table in Relational Model.

→ In 1960 IBM introduced IMS (information Mgmt System) Product. Based on Hierarchial data model.

→ If we want to operate Hierarchical data model Product we are useing Procedural language.

→ If we want to retrive data from hierarchical data model Product then we are data root node onwards. thats why these type of Product retrive data veryslowely. from the database.

eg:-

2) Network Data Model :-

→ In 1970 CODASYL. (conference on Data system language commitee introduced Network Model.

→ This Datamodel based on Many to Many Relationships. thats why this model automatically reduces duplicate data, in this data model also store data in the format of records and also record

type is same as table in Relational Model.

→ In 1970 IBM introduced IDMS (Information Data Managemt system) based on network data Model.

→ If you want to operate this data model also we must use Procedural language.

eg:-

3) Relational Data Model :—

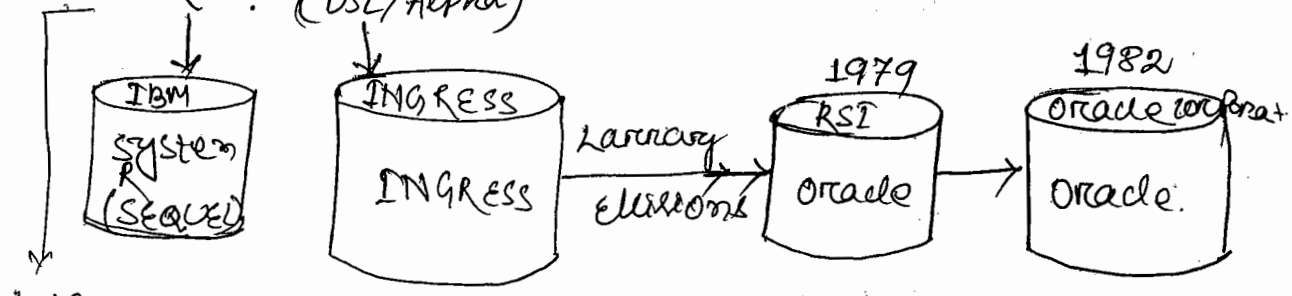
→ In 1970 E.F. CODD introduced relational data Model.

→ In this data model we are storing data in two dimentional tables.

→ Relational data model consist of 3 components i.e

1) collection of objects.
2) Set of operators.
3) Integrity rules.

→ In 1970 E.F. CODD written a paper "Relational model of Data for large shared data banks". (DSL/Alpha)



1NF
2NF
3NF.

Oracle :-

1) SQL
2) PL/SQL
3) Dynamic SQL.

Oracle Versions :—

Oracle 2.0 is 1st version released on 1979.
→ first Public realese.
→ Basic sql functionality, joins

Oracle 3.0 is 2nd version released on 1983.
→ commit, rollback.
→ rewritten in C language.

Oracle 4.0 is 3rd version released on 1984
→ Read consistancy

Oracle 5.0 is - - - 1985
→ Client Server Architecture.

Oracle 6.0 - - - - - - 1986
→ Introduced PL/SQL.
→ Row level locks.

Oracle 7.0 - - - - - - - 1992
→ Integrity constraints.
→ Stored Procedures, stored functions.
→ triggers.
→ Datatype varchar changed into varchar2.
→ truncate table.
→ view compilation.

Oracle 7.1 - - - - - - 1992
→ Dynamic SQL
→ ANSI/ISO SQL92

Oracle 7.2 - - - - - - - 1992
→ inline views.
→ ref cursors.
→ DBMS – Job package.

Oracle 7.3
→ bitmap indexes.
→ utl-file Package.

Oracle 8.0 ---------- 1997.
→ object technology.
→ Nested table, varrays.
→ lobs (large objects).
→ colums increase per a table up to 1000.
→ ~~Auto~~ instead of triggers.

Oracle 8i (Internet) ------ 1998.
→ Materialized based indexes.
→ enable, disable triggers.
→ Rollup, cube.
→ autonomous transactions.

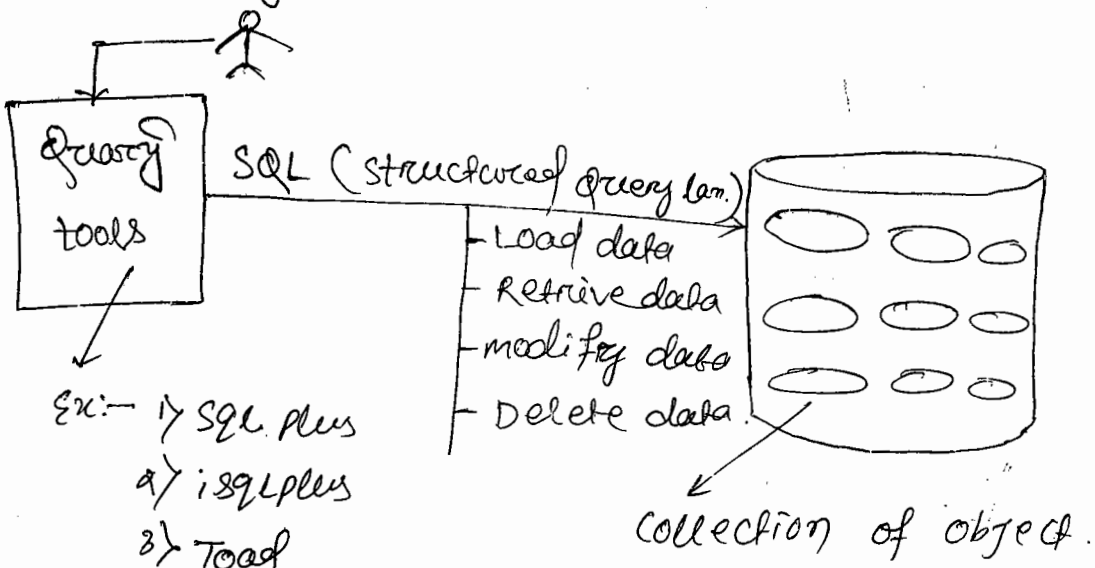
Oracle 9i ---------- 2001
→ Renameing a column
→ merge statement.
→ multitable insert.
→ flashback querys.
→ ansi (or) 9i joins.

Oracle 10g (grid technology) ------ 2005
→ Recycle bin concept.
→ flashback table.
→ Wm_concat()
→ Regular expression
→ indices of clause.
→ Rename tablespace.

Oracle 11g ---------- 2007
→ continue statement. introduce in PL/SQL.
→ Read only tables
→ Vertical column
→ simple_integer datatype introduce in PL/SQL
→ follows clause in triggers.
→ sequences used in PL/SQL without dual table.
→

(4.00 PM) Dt 24/01/14

→ Enable, disable clauses used in trigger specification
→ Named, mixed notations are used in a sub program executed using select statement.



To show the current user the command is

```
SQL> show user;
     User is "SCOTT";
```

Oracle 10g (Enterprise)

```
Username: scott
Password : Triger ↵
Error → Account locked.
```

to unlocking a user :-

```
Username [ \sys as sysdba ] ↵
Password [ sys ] ↵
SQL> Alter user scott acount unlock;
SQL> Conn scott/Triger;
     Password: tiger;↵
     Confirm pwd: tiger↵
```

→ This Process is only 1st time.

→ To View All tables:-

- SQL> select * from tab;

  Ex:- DEPT → master table.
  EMP → child table.

- SQL> set line 100;
  show Proper table;

- SQL> select * from emp;

- SQL> cl scr; or shift + delete.
  → clear the screen.

SQL

→ SQL is a non Procedural language.

→ if you want to operate relational database that we are using non Procedural language.

→ In 1970 E.F CODD introduced DSL/Alpha language which is used to operate Relational datamodels. Later IBM modify this language in to square which is used in system or project. Again idea modified squar into seqeul and sequal into SQL.

→ In 1986 introducing ANSI.
→ In 1987 " ISO.
→ In 1989 " ANSI / ISO SQL89
→ 1992 " ANSI / ISO 192
→ 1999 " ANSI / ISO SQL 99
→ 2003 " ANSI / ISO SQL 2003.

SQL command types :-

- DDL (Data definetion language)
  - → Create
  - → Alter
  - → drop
  - → truncate
  - → rename (oracle 9i)
- DML (Data manupulate language).
  - → Insert
  - → update
  - → Delete
  - → Merge (oracle 9i)
- DQL (Data Query language) OR DRL (Data Retrival lan)
  - → Select.
- TCL (Transactional control language)
  - → commit (save).
  - → Rollback (like undo).
  - → Save Point
- DCL (Data control language)
  - → Grant (Permission).
  - → Revoke (cancle).

→ By default. DDL commands are automatically save it; it is a standart.

→ DML and TCL commands are used directly to the PLSQL.

Mag

(4.00 PM) 25th, Jan, 14

Datatypes :-

→ Datatypes identifies — type of data with in a table column

1) Number (P, S) Ex:- Number (7,2), Number (4).
   → for float
   → for integer
2) Char
   → varchar2 (Max size)
3) Date

Number (P,S) :-

→ Number (P, S) → scale
   → Presision → Total no. of digites.

→ It is used to stored fixed, floating Point nos.

syntax :-

columnname number (P, S)

→ Max limit of Presision is upto 38.

Char :-

→ It is used to store fixed length of Alpha-numeric data in bytes.

syntax :-

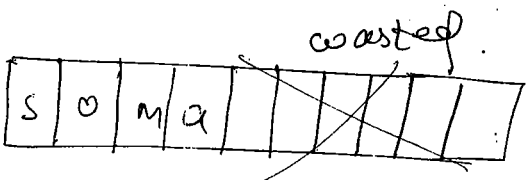
columnname char (size)

→ By default char datatype heaving 1 Byte.

→ Max limit is upto 2000 Bytes.

T1 → char(10)

name
soma



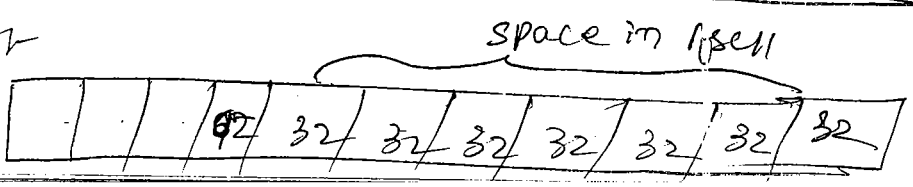
This type Problem is occured then we go to varchar.

To Check interstorage data the query is

sql> Select ~~*~~ ~~from~~ dump (name) ~~from~~ t;

→ Table name

o/p



Varchar$_2$ (max size) :—

→ It is used to stored variable length alphanumeric in bytes.

→ max limites is up to 4000 bytes.

Synax :—

columnname varchar$_2$ (maxsize)

Date :—

→ it is used to stored ~~fixed~~ dates in oracle date formate.

Syntax :—

columnname date.

by Defacult oracle date format is DD-MON-YY.

DDL :— (Data def^n language)

→ These commands are used to describes the structure of the table.

1) Create :—

it is used to create database objects like Tables, views, excess, sequences. etc.

★ creating a table : keyword

syntax - create table tablename(columnnames datatype (size), columnname2 datatype(size),);

Ex:- create table n_1 (sno number(10), name varchar$_2$(10));
— Table is created.

→ SQL> desc n_1 ;

ex:-

Table n_1	
sno - - - - - - -	number (10)
name - - - - - - -	varchar$_2$(10)

To view structure of the table :-

SQL> desc tablename;

2) Alter :-

it is used to change existing table structure.

→ Alter is a Add, modify and drop.

* **Add** :- It is used to add no. of columns into the existing tables.

Syntax :

```
Alter table tablename add (col1 datatype (size);---);
```

Ex:- alter table n1 add (salary numer(10));

sno ---------- number(10).
Name ---------- varchar2(10)
salary ---------- number(10)

* **Modify** :-

→ it is used to change column datatype or column datatype size.

syntax :

```
alter table tablename modify(columnname datatype (size), ....);
```

Ex:- I want to change my sno ~~into~~ which in name is chang to Date.

sql>alter table n1 modify sno date; ↳ desc n1;

Table n1.

Sno ---------- Date
Name ---------- varchar(10)
salary ---------- number(10)

* **Drop** :-

→ it is used to remove colums from the table.

Syntax :-

```
alter table tablename drop column sno
```

Method 1 :-

if we want to drop a single column at a time without using Paranthess then we are using following syntax.

Mon

(4.10 PM) 27th Jan 14

alter table tablename drop column columnname;

keyword

Ex:- alter table n1 drop column sno. ↳ desc n1 ↲

Table n1

Name ..... varchar2(10),
Salary ..... number(10).

Method 2:-

→ if we want to drop a single ~~and~~ or multiple column with using parantheses then we are using following syntax:-

alter table tablename drop (column1, column2 ...);

Ex:-
alter table n1 drop (name); ↳ desc n1;

Table n1.

Salary ..... number(10)

Ex:- alter table n1 drop (sal);
* Error: Can not drop all colums in a table;

3) Drop:-

→ It is used to remove database of objects from database.

Syntax:

drop	objecttype	objectname.

eg:-

1) drop	table	tablename
2) drop	Prodedure	Procedurename
3) drop	view	viewname

dropping a table:-

Syntax:

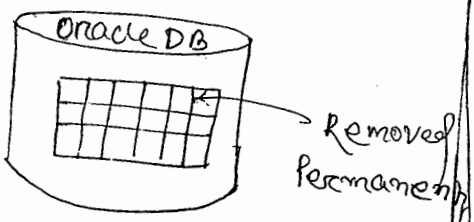
drop table tablename

eg:- drop table n1;

Table dropped.

→ before Oracle 10g

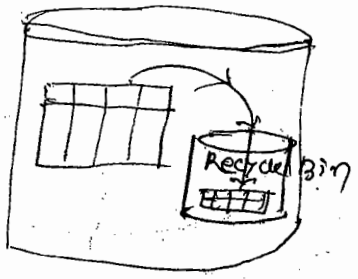
→ drop table tablename;



droping a table

oracle10g (Enterprise edition)

→ drop table tablename;



→ get it Back from Recyclebin

→ flashback table tablename to before drop;

→ To drop Permanently

→ drop table tablename Purge

Keyword

Recyclebin which is stored:-

eg:- SQL> flashback table n1 to before drop;

flashback complete. (it come from recyclebin).

Permanently drop:-

eg:- SQL> drop table n1 purge;

Table dropped.

(table is not exist even in recycle been).

testing

SQL> flash back table n1 to before drop;

Error: object not in recycle bin.

Recyclebin :-

→ Recyclebin is a data dictonary. i.e whenever we are installing Oracle server automatically recyclebin readonly table created.

→ This table stores ~~Propo~~ droped objects, if we want to view structure of the recyclebin then we are using desc command.

eg:-

sql> desc recyclebin;

Oracle Database.

Original-Name:	object-name

→ we can also drop tables from recyclebin using following syntax:

Purge table table name

N:B* → we can also drop or tables from recyclebin using following syntax:

Purge recyclebin;

sql> create table E7(sno number(10)); sql> drop table C7;

sql> desc recyclebin; ← (to view recyclebin;)

sql> select ORIGIONAL-NAME from recyclebin;

ORIGINAL NAM

C7.

sql> Purge table C7; (← droping from recyclebin;)

4) Truncate :-

→ It is used to delete total data from a table, and also this data Permamently deleted.

syntax:

truncate table tablename;

→ eg:

sql> create table C9 as select * from emp;

sql> Select * from cq;

sql> Truncate table cq;

Testing:—

→ want to view all data.

sql> Select * from cq;
no row selected.

→ want to view Table;

sql> desc cq;

5) Rename:—

→ It is used to rename a table and rename is a column also.

Renameing a table:—

→ syntax:

rename oldtable name to newtablename;

eg:- sql> rename cq to dq;
table created.

→ Renameing a column (oracle 9i):—

Syntax:

alter table tablename rename column oldcolumn to newcolumnname.

→ only one column at a time.

eg:- alter table emp rename ~~empno column~~ empno to slno.

sql> desc emp;

N:B → By default all DDL commands are automatically committed.

Mag

(4.00 PM) 27/01/14

DML :(Data Manipulation language)

→ Insert
→ update
→ Delete
→ Merge. (oracle 9i).

→ These commands are used to manipulate data in a table.

→ **Insert :-**

→ it is used to insert data into a table.

Method 1 :-

syntax :

```
insert into tablename values (value1, value2...);
```

eg :- Create table G1 (SLno Number (10), Name varchar2 (10));

SQL> insert into G1 values (1, 'soma');

1 row inserted.

SQL> select * from G1;

SNO	Name
1	some

← These are called record.

Method 2 :- (useing substitutional operator (&))

→ syntax :

```
insert into tablename values (&col1, &col2...);
```

eg :- SQL> insert into G1 values (&SNO, '&Name');

Enter value for SNO : 3

Enter value for Name : somapatel.

one row inserted.

SQL> /

Enter value for SNO : 4

Enter value for Name : Patel.

sqly select * from G1;

SNO	Name
1	Soma
3	SomaPatel
4	Patel.

→ It will work if we can write something another name in place of column name. like 1, 2.

eg:- s SLNO. → s1

iii) Method 3 :- (Skipping columns).

syntax :-

```
insert into tablename (col1, col2 --) values
                      (val1, val2 --);
```

eg:- insert into g1 (name) values ('Om');

1 row created.

sqly select * from g1;

SLNO	Name
1	Soma
3	SomaPatel
4	Patel
X	Om

→ Here, if we want to insert a name into name column then it is possible, but it is no identify or SLNO is empty.

→ Add a another column to the table g1 i.e add.

sqly alter table g1 add address.

1 column is created.

sqly select * from g1;

SLNO	Name	Address
1	Soma	
3	SomaPatel	

→ So, we want to add a data in address column of SLNO 2.

→ It is Possible using the '<u>update</u>' command.

<u>UPDATE</u>:-

It is used to modify data in a table.

syntax:

```
update tablename set columnname = newvalue
where column = old value;
```

eg:- ~~update~~ ~~set~~

```
update emp set sal = 1000 where ename = 'SMITH';
```

<u>NB</u>:-

In All database systems if we want to insert Particular cell values, then also used 'update'.

eg:- update g1 set add = 'mumbai' where SLNO = 1 ;
Name = 'soma';

1 row updated;

sql> select * from g1;

SNO	Name	Address
1	soma	Mumbai
3	somapatel	
4	Patel.	

→ if I want to delete a Particular data from table like in add column mumbai is delete, it is Possible using <u>update</u> command.

eg:- sql> update g1 set address = null where address = 'mumbai';

sql> select * from g1;

SNO	Name	Address
1	soma	
3	somapatel	
4	Patel	

Delete :-

It is used to delete all rows or Particular rows from a table.

syntax :-

```
delete from tablename ; ← delete all rows
        OR
delete from tablename where columnname =
                                  value;
```

delete Particular rows.

eg:-

say delete from g1 ;

4 rows are deleted ;

say rollback ; (without using commit command).

say select * from g1;

SLNO	Name	Address
1	soma	
3	somapatel	
4	patel.	

Q) diff^n bet^n truncate and delete command :-

→ when ever we are useing truncate are delete from tablename then Oracle server automatically deletes total data.

→ when we are useing delete from tablename then automatically detete a data stored in a buffer.

→ we can get it back this data using rollback (without using commit).

→ when we are useing truncate we can not get it back deleted data using rollback becoz truncate is a DDL command bydefault all DDL commands are automatically commited.

(4.00 pm) 29/01/14

Data retrival language or Data query language (DQL) :-

⇔ Select :-

syntax:

```
select columnname1, col2 from tablenamelwhere
condition | group by columnnamel heavingl conditionl
order by colname [asc/desc];
```

1) Select all columns (represent '*') and all rows. (by default oracle using 'True'.)
2) Select all col and Particular rows. (always represent 'where')
3) Select Particular cols (represent 'col1, col2' (column name)) and all rows.
4) Select Particular cols and Particular rows.

Creating a new table from existing table using select :-

syntax:

```
create table newtablename as select * from
existing tablename;
```

eg:-

sql> Create table first as select * from emp;

sql> Select * from first;

emp ← existing table

—	—	—	—
—	—	—	—
—	—	—	—
—	—	—	—

first ← new table

—	—	—	—
—	—	—	—
—	—	—	—
—	—	—	—

NB:

In all database systems when we are copy from one table to another table constraints never copyed.

Creating a new table from existing table without coping data :-

syntax:

```
create table newtablename as selecte * from
existingtablename where falsecondition;
```

eg:-

sql> create table second as select * from emp where 1=2;

sql> select * from second;
no row selected (becoz ~~this type~~ ther is no row are exist).

sql> desc second;

Operaters ~~using~~ in select statement :-

1) Arithmatic operators. (+, -, *, /) → After select col1, col2.
2) Relational operators. or comprehension operators. (<>, ≤, ≥, =, :=)
3) logical operators. (AND, OR, NOT)
4) special operators.

must use 'where'.

→ ~~1) is used~~ Arithmatic oper. are used in either where condition or After select col1, col2.

→ Relational, logical or special operators are used in 'where'.

→ Arithmatic operaters are used in number, date datatype column.

Q. write a query to display ename, sal, Annual from emptable?

```
sql> select ename, sal, sal*12 "annual" from emp;
sql>
```

Q. write a query to display a employee except Job ~~and~~ as clerk from emp table?

```
sql> select * from emp where Job <> 'CLERK';
sql>
```

Q: write a query to ~~for~~ display the emp who are more then 2000 sal from emp table.?

```
SQL> select * from emp where SAL>2000;
```

```
SQL> select * from emp where JOB='CLERK' AND SAL>2000;
```

individual condition are checked the we go to 'OR'.

```
SQL> Select * from emp where JOB='clerk' OR SAL>2000;
```

Q: write a query to display the emp who are belongs to 20, 30, 50, 70 & 90 deptnos from emp table.

```
SQL> select * from emp where DEPTNO=20 OR DEPTNO=30
     OR DEPTNO=50 OR DEPTNO=70 OR DEPTNO=90;
```

4) Special operators :-

1) in	**Oppsite**	
2) Between	not in	All are used in where
3) isnull	notbetween	
4) like	is not null.	
	not like	

IN :-

→ it is used to pick the value one by one from list of values.

→ generally inplace of all operator we are useing 'IN' operator becazuse 'IN' oper. Performance is very first compare to 'OR' operator.

→ Generally in all database systems multiple rows ~~and~~ some query we must use 'IN' Operator.

syntax:

```
Select * from tablename where columnname in
(list of values);
```

→ any datatype column values.

Note (4.00 PM). 30th/Jan/14

eg:- Select * from emp where deptno in (20,30,50,70,90);

- Select * from emp where ename in ('SMITH','JONES');
- Select * from emp wher deptno IN(10,20,NULL). (it work)
- Select * from emp wher deptno NotIN (10,20,NULL). (it doesn't work)

Error: no rows selected.

NB:-

NOT IN operator doesn't work with null values.

Between :-

→ This operator is used to retrive range of values

→ This operator is is also called as between and operator.

Syntax :-

```
Select * from tablename where colname betn
AND highvalue;
```

eg:- select * from emp where sal betn 5000 and 2000

Error: no row selected.

becoz :- betn operator is not work high to low value.

so,

sol) select * from emp where sal betn 2000 & 5000;

Not Between: is used to when we no more required.

NULL :-

→ Null is a undefined, unavailable, unknown value.

→ it is not same as '0'.

→ Any arithmatic operation perform null again it becomes NULL.

Ex:- null + 50 = null.

Q: write a query to display ename, salary, commission, Salary + commission of the Employee SMITH from emp table.

eg:-

sql> select ename, sal, ~~totcomm~~ commision, sal+comm from emp where ename='SMITH';

Ename	SAL	COMM	SAL+COMM
SMITH	400	—	—

→ SMITH salary is 400 & no comm but sal+comm is 400 but it is null

→ To overcome This Problem oracle Provided NVL().

NVL():-

→ NVL is a Predefined function which is used to replace or substitute userdefined value implace of null.

Syntax:-

NVL (exp1, exp2)

→ Here express1, exp2 must belongs to same datatype.

→ If Exp1 is null then it returns exp2 otherwise it returns exp2, ~~otherwise~~.

Ex:- i) NVL (null, 50) ii) NVL (30, 50)

o/p:- 50 o/p:- 30

eg: sql> select NVL (null, 50) from Dual.

50

sql> select NVL (50, 70) from Dual

50.

Solution:-

select ename, sal, comm, sal + NVL (comm, 0) from emp where ename='SMITH';

ENAME	SAL	comm	SAL+COMM
SMITH	400	—	400

Sal + NVL(comm, 0)
⇒ 400 + NVL(NULL, 0)
⇒ 400 + 0
⇒ 400.

→ if exp1 & exp2 datatypes are not same then it does not work so we convert to type casting.

eg:- select ~~ename~~, sal, ~~com~~, nvl(to-char(comm),0) from emp ~~tab~~ where name='smith';

ENAME	SAL	comm
SMITH	400	no commin.

is null, is not null:-

→ These two special operators are used in where condition.

→ Thes two special oper. Test whether a col having null Values or not.

Syntax :-

Select * from tablename where columnname is null.

OR

select * from tablename where columnname is not null;

eg:- select * from emp wher mng=null;
no row selected (not error but show this type of msg.

- Select * from emp where comm is null;

Q: write a query to display the emp who are not getting commission from emp table.

-sol) Select * from emp where comm is null;

Like:-

→ This operator is used to retrive data based on char ~~data~~ pattern.

→ along with like oper. we are using char, 2 special char these are.

1) % (Percentage) → strong or group of character.

2) _ (underscore) → single character

syntax:-

```
Select * from tablename where colname like
'chacter partern';
```

Q: write a query to display the emp whose ename start with capital latter 'M' from emp table using like operator.

Q) Select * from emp where ename like 'M%';

o/p :- MARTIN

Q: write a query to display the emp whose ename second letter would be capital 'L' from emp table using like operator.

Q) Select * from emp where ename like '_L%';

o/p:- ALLEN
BLAKE
CLARK

Q: write a query to display the employee who are joining in the year 81 from emp table using like operator.

A) Select * from emp where hiredate like '%81'.

Q: W.A.Q to display the employees who are joining in the month Dec from emp table using ~~tong~~ like operator.

A) select * from emp where hiredate like '%Dec%'.

eg:-

emp
hiredate
27-Dec-81
23-Dec-80
20-Dec-81.

Concatenation operator (||) :-

→ This operator is used to display column data along with lateral strings.

eg:-

select 'my employee names are' || ename from emp;

select ename || ' ' || sal from emp;

display space.

Functions :-

→ functions are used to solve some particular task.

→ Oracle also having 2 types of functions

1) Predefined fun.

2) user defined fun.

Predefined functions :~

The Predefined functions are :

1) Number functions
2) characters functions
3) Date functions.
4) Group functions (or) Aggregate fun^n.

Number function :~

These fun^n operates number data.

1) abs() :~

→ These fun^n is used to convert -ve values into +ve values.

eg:- select abs(-50) from dual;

o/p :- 50.

→ Dual :~

→ Dual is a Predefined Virtual table, which is used for testing the Predefined, User defined fun^n(). which contain only one row & one column.

→ Generally if we want to generate sequence values using select statement there must be used dual.

eg:- This table also used to perform mathematical operations.

sql> select 50+90 from dual;

o/p :- 140

→ By default Dual Table column DUMMY ← VARCHAR2

sql> select * from dual;

D

X

sql> desc dual;

Sol) Select ename, comm, sal, abs(comm-sal) from emp where comm is not null;

2) **Mod (m,n) :-**

→ it returns reminder of the (m/n) reminder.

→ eg:- Select mod (10,5) from dual.

O/P :- 0.

3) **Round (m,n) :-**

→ it round ~~a~~ even floated value no. 'm' based on 'n'.

eg:- Select round (1.7) from dual.

O/P:- 2

Select round (1.2) from dual.

O/P:- 1

Select round (1.23456, 3) from dual.

O/P :- 1.235

1.234/56 , 3

100

56 above 50% add 1

1.234
1
1.235

N:B

round always checks reminding no. if reminding no. above 50% then 1 added to rounded no.

eg:- select round (1285.356, -1) from dual.

O/P:- 1290

Step1:- 1280

(above 50% replace 0)

Step2:-

5 → 0

5 → above 5%

1280

Step3:-

1 is add to the round no i.e 1280
+ 1
1290 — Ans

-4 -3 -2 -1
1285.356 , -1

5 → 0

above 50%

1280
1
1290

Q say select round (1295, -2) from dual

o/p :- 1300

Step 1 : 1200

Step 2 : 95 . . . . 00 | 95 → 0

95 - - - above 50% | 5 → 50% above

1200 | 1290

Step 3 : 1200 | 1290

1 | + 1

1300 — Ans | 1300

say select ename, sal, round(sal/22,1) from emp.

say select ename, sal, round(sal/22), round(sal/22,1) from emp;

4) Trunc (m, n) :-

→ it truncates given floated valued no. ~~emp~~ in based on 'n'. ~~select~~ ~~round~~.

say select trunc(1.8) from dual

o/p :- 1

say select trunc(1.23456, 3) from dual.

o/p :- 1.234

5) Greatest(exp1, exp2 - - -), least (exp1, exp2 - - -) :-

→ greatest returns maximum values within given expression where as least returns minimum values among given expression.

eg :- select greatest (4, 7, 9) from dual;

o/p :- 9

→ if we pass only one column it doesn't work. it pass one or more column required.

select ename, sal, comm, greatest (sal, comm) from emp where ~~column~~ comm is not null.

01/02/14

Mar ? **Ceil(), floor():~**

→ Ceil date nereast greatest integer. where as floor returns nearest lowest integer.

eg:- select ceil (1.4) from dual.

o/p - 2.

- sql> select floor (1.9) from dual;

o/p - 1

characters functions:~

1) **upper():-**

it is used to convert column values or string into uppercase.

eg:- sql> select upper('abcd') from dual;

o/p :- ABCD

sql> select upper(ename) from emp;

↳ here no need (' ') becoz it is a column name.

2) **lower():-**

eg:- sql> select lower('ABCD') from dual;

o/p :- abcd

sql> select lower(ename) from emp;

→ it is used to convert column value or string into lowercase.

→ I want to change my ename column all data uppercase to lower case then:

sql> update emp set ename = lower(ename);

3) **initcap():-**

→ it is use to convert 1st letter is capital and remaining all letter convert into small.

→ sql> Select initcap(ename) from emp;

→ sql> Select initcap('ab cd ef') from dual;

O/P :- Ab Cd Ef

4) length() :-

This fun^n always returns no. datatypes it is used to return total length of the string including spaces.

eg:-

sql> Select length('AB CDE') from dual;

O/P :- 6 (AB CDE / 123456)

5) Substring :- substr()

→ it will extract portion of the string with in given string based on last two parameters.

eg:- select substr('ABCDEFGHI', ~~from~~ -5) from dual;

O/P :- EFGHI

sql> select substr('ABCDEFGHI', 2, 3) from dual;

O/P :- BCD

sql> select substr('ABCDEFGHI', -2, 3) from dual;

O/P :- HI

sql> select substr('ABCDEFGHIJ', -6, 3) from dual;

O/P :- EFG

syntax :-

Substr(columnname or 'stringname' , starting position , number of char from position)

must number.

Q: write a query to display the employee whose ename second letter would be 'AR' from emp table using substring from.

```
say Select * from emp where substr(ename,2,2)
                                    = 'AR';
```

O/P:- ENME
WARD
MARTIN

NB:
→ ~~group~~
→ we are not allowed to use group functions in where class. but we allowed to use number, char, Date functions in where class.

6) instr() :-
eg:-
say select instr('ABCD', '*') from dual;
O/P:- 0
say select instr('ABCD', 'C') from dual;
O/P:- 3
say select instr('AB*CD', '*') from dual;
O/P:- 3
say select instr('ABCDEF', 'DE') from dual;
O/P:- 0

→ These fun always returns number datatype. it is used to return position of the Delimiter (or) position of the character, position of the string within given string.
→ eg:- say select instr('AB*CDEFG*KLMN*ZXV', '*', 5,2) from dual;

Ans:- 14

say Select instr('AB*CDEFG*KLMN*ZXV', '*', -5, 2) from dual;
o/p :- 3

say Select instr('AB*CDEFG*KLMN*ZXV', '*' -4, 2) from dual
o/p:- 9

Syntax :-

instr(columnname of or 'stringname' , 'str', starting searching string, numbers of occurance)
from position
must be numbers

→ always intstr returns position based on 3rd, 4th parameters and also oracle server count no. of char left side 1st position onwards.

Mad ?

02/02/14

Lpad():- (Left Padding)

→ It will fills remaning spaces with this specified character on left side.

eg:- syntax :-

Lpad(colname or 'stringname' , total length, 'specified char')

eg:- Select Lpad('ABCD', 10, '*') from Dual;

o/p :- ******ABCD

RPad() :- (Right padding).

eg:- Select Rpad('ABCD', 10, '#') from dual;

o/p: ABCD ######

→ syntax and desn same as Lpad.

```
SQL> select Rpad(Ename, '30', '-') from emp;
```

```
SQL> Select Rpad(Ename, 30, '-') || SAL from emp;
```

LTRIM() :~

→ it is used to remove ~~space~~ specified char of the left side of the ~~screen~~. string.

syntax:-

```
LTRim(colname or stringname, {set of chars})
```

```
SQL> select Ltrim('SSMISSTHSS', 's') from Dual;
```

O/P :- MISSTHSS → SSMISSTHSS.

```
SQL> select JOB, LTrim(Job, 'c') from emp;
```

O/P :- LERK → clerk

```
SQL> select JOB, LTrim(JOB, 'CSM') from emp;
```

O/P :- LERK → clerk
ALesman → salesman
MAnager → Anager.

(either C or S or M)

RTrim() :~

eg:-

```
SQL> select rtrim('ssmissthss', 's') from dual;
```

O/P :- ssMissth → ssMissthss

Trim() :~

Oracle 8.0 introduce trim() function.

→ It is used to remove left and right specified character.

syntax:

```
trim('specified char' from 'stringname');
```

sql> Select trim ('s', from 'ssMISSTHss') from dual;

o/p :- MISSTH

N/B: it is used to remove 1st and last spaces from the given strings.

sql> select Length(trim(' smith ')) from Dual;

o/p :- 5 → to check the character length.

Translate(), Replace() :-

eg :- sql> select translate('JOHN', 'H', 'N') from dual.

o/p :- JONN.

same o/p.

sql> select Replace ('JOHN', 'H', 'N') from dual.

o/p :- JONN.

sql> select Replace ('JOHN', 'H', 'SHKJILO') from dual

o/p :- JOSHKJILON ← it replace one char by string. but not do in a translate.

sql> select Translate ('JOHN', 'H', 'EHKLMN') from dual.

o/p :- JOEN → translate() is translate single char.

→ Translate is used to replaces char by char where as replace is used to replaceses char by string or string by string.

syntax of translate() :-

Translate ('str1', 'str2', 'str3');

eg :- select translate ('ABCDEF', 'FEDCBA', '123456') from dual.

o/p :- 654321.

eg :- select JOB, Replace (JOB, 'SALESMAN', 'marketing') from dual;

o/p :- salesman → marketing;

SQL> select Replace ('SSMISSTHSS', 'S') from dual;

O/P :- MITH ← it replace all char 's';

DATE :-

→ By default oracle date format is DD-MON-YY.

1) sysdate
2) ADD_MONTHS()
3) LAST_DAY()
4) NEXT_DAY()
5) MONTHS_BETWEEN()

1) SYSDATE :-

it is used to return current date of the system in oracle Date format.

eg :- 03-FEB-14

2) ADD_MONTHS() :-

it is used to add or substract no. of months to the specified date based on Second Parameter.

syntax :-

Add_Months ('date', number);

eg :- SQL> select add_months(sysdate, 1) from dual;

O/P :- 03-MAR-14

SQL> Select add_months(sysdate, -1) from dual;

O/P :- 03-JAN-14

3) LAST_DAY() :-

last_day → it returns last date of the specified month.

eg :- select last_day(sysdate) from dual;

O/P :- 28-FEB-14;

next-day() :~

it returns next occurance day from it's specified date based on second parameter.

eg:- select next-day(sysdate, 'sunday') from dual;

O/P :~ 09-FEB-14

→ MONTHS_BETWEEN() :~

syntax :-

months_between (date1, date2);

→ This fun always returns number datatype.

→ it returns no. of months betn two specified dates.

→ Here always date1 > date2, otherwise it returns -ve sign.

eg:- Select moths_between ('16-jan-05', '16-aug-05') from dual;

O/P :- -7.

→ always this fun returns floating point values when we not passing exact dates, to overcome this problem we must use round function.

eg:- select round (months_between ('26-jan-05', '16-aug-05')) from dual;

O/P :- -7

eg:- Select ename, sal, round (months_between (sysdate, hiredate)) from emp;

DATE Arithmatic :~

→ Date + number ✓
→ Date − number ✓
} possible

→ Date1 + Date2 ✗
→ Date1 − Date2 ✓

eg:- SQL> Select sysdate+1 from dual;

SQL> Select sysdate-1 from dual;

SQL> Select sysdate-sysdate from dual;

→ write a query to display 1st date of the current month using predefined functions.

eg:-

```
SQL> Select add_months(last_day(sysdate),-1)+1
                                    from dual;
```

O/P:- 01-FEB-14.

DATE conversion functions :-

Rules :-

1) to_char()

2) to_date()

to_char() :-

→ It is used to convert date type into character type.

→ It is used to convert date type into date string.

eg:-

SQL> Select to_char(sysdate,'DD/MM/YYYY')
from dual;

O/P:- 04/02/2014

NB:-

To_char() is a case sensitive function.

eg:- Select to_char(sysdate,'DY') from dual;

O/P:- TUE

select to_char(sysdate,'DAY') from dual;

O/P:- TUESDAY.

DAY → TUESDAY

day → tuesday

D
DD
DDD
MM
YY
YYYY } ← number

DAY
DY
MONTH
MON
YEAR } ← character.

D → Day of the week (sunday → 1, MONDAY → 2 ...)

DD → DAY of the MONTH

DDD → DAY of the YEAR.

eg:- select to_char (sysdate, 'DDSPTH') from dual;

o/p:- ~~fourth~~ FOURTH

SP → SPELL OUT..

eg:- select to_char(sysdate, 'HH:MI:SS') from dual;

sql>select to_char('20-jun-05', 'DD/MONTH/YY') from dual;

o/p:- error...

NB:-

- whenever we are using to_char, always 1st parameter must datetype.

to_date():-

→ It is used to convert chartype into datetype i.e it converts date string into datetype (oracle date formate).

eg:- select to_date('20/June/04') from dual;

o/p:- 20-JUN-04
→ oracle date format.

eg:- sql>select to_date('27/06/04') from dual;

o/p:- error - not a valid month

becox:- DD-MON-YY → 27/06/04
no. char no. → 27/June/04.

~~select~~

NB:-

when ever we are useing to-date passed Parameter return value match with default date format return values otherwise oracle server returns error. to overcome this Problem used in second Parameter as same as 1st Parameter format.

→ Then only oracle server automatically converts date string in to datetype.

```
select to-date ('27/06/04', 'DD/MM/YY') from dual;
```

O/P :- 27-JUN-04.

```
sql> select To-char (to-date ('15-JUN-05'), 'DD/MONTH/YY') from dual.
```

O/P :- 15/JUNE/05

```
sql> select to-date ('07-feb-05')+6 from dual;
```

O/P :- 14-FEB-05

```
sql> select to-date('08-02-05', 'DD-MM-YY')+6 from dual;
```

O/P :- 14-FEB-05

Q: write a query to display the employees who r joining the month Dec from emp table. using To-char().

A) select * from emp where to-char (~~to-date~~ hiredate, 'MM') = '12';

(in number)

→ select * from emp where to-char(hiredate, 'MON') = 'DEC';

Q: write a query to display employee who are joining in the year 81 from emp table using ~~to-char~~ to_char().

A: Select * from emp where to_char(Hiredate, 'YY')='81';

~~Select * from emp~~ O/P:- 19-MAR-91

select Hiredate to_char(hiredate, 'YYYY') from emp;

O/P:- 19-MAR-1991

NB: when ever we are parsing date string into date() then oracle server automatically converts date string into date type. But here passed parameter return value matched with the default date ~~values~~ datatype return value.

eg:-

say select last_day('12-jun-05') from dual;

O/P:- 30-June-05

say select last_day('12-06-05') from dual;

O/P:- not a valid month.

Sol^n:-

say select last_day(to_date('12-06-05', 'DD-MM-YY')) from dual;

O/P:- 30-JUN-05.

Round, trunc function used in dates:-

date datatype contains both date and time. whenever we are using round and trunc() that date can be changed based on time position.

eg:-

say select to_char(sysdate, 'DD-MON-YY HH:MI:SS) from dual;

O/P:- 05-Feb-04 48:00:00

→ whenever we are useing round function then automatically one day added to the current date and also time portion automatically set to zero, if time portion >= 12:00

```
SQL> Select to_char (Round (sysdate), 'DD-MON-YY HH24:MI:SS') from dual;
```

O/P :- 06-FEB-14 00:00:00
↳ (one day added)

→ whenever we are useing truncate function oracle server automatically set to time portion '0', and also returns same date.

```
SQL> select to_char (TRUNC (sysdate), 'DD-MON-YY HH24:MI:SS) from dual;
```

O/P :- 05-FEB-14 00:00:00.
↳ (no day added current day show)

Q: write a query to display the employee who are joining today from emp table.

A) SQL> insert into emp(empno, ename, hiredate) values (1, soma, sysdate);

SQL> select * from emp where hiredate=sysdate;

O/P :- no rows selected;

Soln :-

```
SQL> select * from emp where trunc(hiredate)=trun (sysdate);
```

Group function or Aggregate function :-

→ 1) max()
2) min()
3) avg()
4) sum()
5) count(*)
6) count (column name)

→ All group() operate, data and returns a single value.

Max():~

→ it returns a max value within a column.

```
sqly select max(sal) from emp;
```

O/P :- 28,000.

```
sqly select max(hiredate) from emp.
```

O/P :- 23-MAY-87.

```
sqly select max(ename) from emp;
```

O/P :- ward.

Min():~

→ Select min(sal) from emp;

O/P :- 800

→ sqly select min(hiredate) from emp;

O/P :- 17-DEC-80.

N:B we are not allowed to use groupfunction in where class.

Avg():~

```
sqly select avg(sal) from emp;
```

O/P :- 128.43.

```
sqly select avg(comm) from emp;
```

O/P :- 550

N:B By default all group functions ignores null values except count (*).

```
sqly select avg(nvl(comm,0)) from emp;
```

O/P :- 157.142857.

Sum():~

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

O/P :- 29125

Mqa (4:10PM) 06/02/14

count(*) :~

→ it counts no. of rows in a table. ~~including null value~~.

say select count(*) from emp;

o/p:- 14

count(columnname) :~

→ select count(empno) from emp;

o/p:- 14

→ it counts no. of notnull values within a column.

say select count(comm) from emp;

o/p:- 4.

- say select count(mgr) from emp;

o/p:- 13.

say select count(distinct(deptno)) from emp;

o/p:- 3

Mqa ? group by :~

This ~~logic~~ class is used to arrenged similar data item into set of logical group.

→ This class is used in select statement only.

syntax :-

```
select colname ....
from tablename
group by columnname;
```

say select deptno, count(*) from emp group by deptno;

o/p :-

deptno	count(*)
10	3
20	5
30	6

Q: write a query to display no. of employee to deptno wise from emp table useing groupby class.

sol)
Q: write a query to display no. of employee Job wise from emp table. useing group by.

sol) select Job, count(*)
from emp
group by Job;

o/p:-

Job	count(*)
CLERK	4
SALESMAN	4
PRESIDENT	1
MANAGER	3
ANALYST	2

sol) select deptno, max(sal), minsal from emp
group by ~~Job~~, deptno;

o/p:-

Deptno	max(sal)	min(sal)
30	2850	950
20	3000	800
10	5100	1300

NB:- we can also use group by class without useing group functions.

eg:- sol) select deptno from emp group by deptno.

o/p:-

dept no
30
20
10

Rule:-

→ otherthen group fun^n columns specified after select those columns must use after group by otherwise oracle server returns an error "not a group by expression"

eg:- Select deptno , sum(sal) Job from emp group by deptno, Job;

o/p :-

deptno	Job
100	—
100	—

eg:- select deptno, max(sal), ename from emp group by deptno, ~~job~~ ename;

deptno	max(sal)	ename

NB:-
if we are try to display group function with another column then database server return error to overcome this proble we must use group by class.

Step1:-
say select avg(sal) from emp;

avg(sal)
2080.35714

Step2 :-

say select Job, avg(sal) from emp.
error: not a single-group group function

Soln:-
say select Job, avg(sal) from emp group by Job;

Q: ~~say~~ display those dept heaving more than 4 empl from emp table useing group by.

say select deptno, ~~job~~ count(*), from emp group by deptno where count(*) >4 ;

error:-

Mon (4.wpm) 07/02/14

Soln:-

sqy select deptno, count(*) from emp group by deptno heaving count(*) > 4.

O/P:-

Deptno	count(*)
30	6
20	5

Heaving:-

→ After group by class we are not allowed to use 'where' class in place this one ANSI/ISO SQL Provided heaving clause.

→ Generally if we want to restic rows in a table, then we are using 'where' as if we want to restic groups after group by then we must be use heaving clause.

→ generally we are not able to use group functions in where class. but we can also use group fun^n in heaving class.

sqy select deptno, sum(sal) from emp group by deptno heaving sum(sal) > 9000

Deptno	sum(sal)
30	94000
20	108750

Mon Order by:-

→ This class is used to arrange data in short in order along with this class, we are useing two keywords i.e:

1) Asc
2) desc

→ By defaulet order by class heaving Ascending order.

eg:- Select sal from emp order by sal;

O/P:- 1000
1500
2450
2850

Lowest to higest.

Syntax:-

Select * from tablename order by columnname [asc/desc].

eg: Select sal from emp order by sal desc.

Select * from emp order by ename desc.

NB:-

in order by class we can also use column position which is use to improves performance of the query.

eg:- Select * from emp order by 6 desc.

say Select deptno, count(*) from emp
where sal > 1000
group by deptno
having count(*) > 3
order by desc.

O/P:-

deptno	count(*)
30	6
20	4

Rollup, cube :-

→ Oracle 8i introduced rollup, cube class these classes are used along with group by class only.

→ These clause are used to calculate sum total, grand total automatically.

syntax :- For Rollup

```
select col1, col2 ----
from tablename
group by Rollup (col1, col2 --);
```

syntax :- for cube.

```
select col1, col2 ---
from tablename
group by cube (col1, col2 --);
```

→ Rollup is used to calculate sum total values based on a single column, if we want to calculate sum total, grand total based no. of column then we are useing cube.

eg :- sql> select deptno, job, sum(sal) from emp group by rollup(deptno, job);

sql> select deptno, job, sum(sal), count(*) from emp group by cube (deptno, job);

I want to display ename & sal with total sal.

sql> select ename sum(sal) from emp group by rollup (ename);

Joins :-

↳ Joins are used to retrive ~~every~~ data from multiple tables, If we are joining n tables then we are using (n-1) joining conditions.

↳ Oracle have following types of joins :-

1) equi join / inner join
2) Non equi join
3) self join
4) outer join

These joins are also called as 8i joins.

Oracle 9i Joins or ansi Joins :-

1) inner joins
2) left outer joins
3) right outer join
4) full outer join
5) natural join.

→ In oracle without useing joining condition also ~~without a~~ retrive data from multiple ~~rows~~ tables. In this case oracle server internally useage Cross join.

→ Cross join internaly works based on cartian Product that why this join returns more duplicate data.

eg:-

sql> select ename, sal, dname, loc from emp, dept ;

8i Joins :-

1) Equi Join / inner Join :-

→ Based on equality condition we are retriving data from multiple tables, here joining condition of column must belongs to same datatype.

→ whenever tables contains common col names then only we are useing equi join.

syntax :-

```
Select col1, col2 ....
From tablename1, tablename2 ....
where tablename1.commoncolname =
joincondition tablename2.commoncolname.
```

Mar 7 (4:00) 08/02/24

eg:- Select ename, sal, deptno, dname, loc
from emp, dept where emp.deptno = dept.deptno
/

error.

O/P:- error:- column ambiguously defined.

Solⁿ:-
```
Select ename, sal, dept.deptno, dname, loc
from emp, dept
where emp.deptno = dept.deptno;
/
```

NB:- generally to avoid ambiguite feature, we must specified every column along with table-name.

Mar 7 **using table aliasnames:-**

Select ename, sal, dept.deptno, dname, loc from emp, dept
where emp.deptno = dept.deptno.

→ we can also use aliasnames for the table in form clause.

→ These aliashname must be different name.

syntax:-

```
Tablename1 aliashname, Tablename2 aliashname2
```

→ Select ename, sal, d.deptno, dname, loc from emp e, dept d
where e.deptno = d.deptno.

NB: using equi joins we are retriving matching rows only.

(here deptno 40 does not display if u r using d.deptno also)

Q: write a query to display where the employee who are working in the location chicago from emp, dept table using equijoin.

A: Select ename, loc from emp, dept where emp.deptno = dept.deptno AND loc = 'CHICAGO'.

O/P:-

ename	loc
WARD	CHICAGO
TURNER	"
ALLEN	"
JAMES	"
BLAKE	"
MARTIN	"

NB: If we want to restrict row after joining condition then we must use ~~joining~~ ~~oper~~ AND operation in Q: Joins where as Q: Joins we can also used where classes.

Q: write a query to display d.name sum of sal from emp, dept table using equijoin.

A: Select d.name sal(sum) from emp e, dept d where e.deptno = d.deptno group by dname.

O/P:-

D.DNAME	SUM(SAL)
A/C	9050
RESEARCH	11275
SALES	9600.

Select d.deptno, d.name, sum(sal) from emp e, dept d where e.deptno = D.deptno group by d.deptno, Dname.

Deptno	deptname	sum(sal)
10 A/C	A/c	9050
20 Res	Res	11275
30 Sales	Sales	9600

→ Select DNAME, SUM(SAL) FROM emp E, DEPT D
where E.Deptno = D.deptno group by Dname
heaving sum(sal) > 1000.

o/p.

Dname	sum(sal)
Research	11275.

→ Select Dname, sum(sal) from emp e, dept d
where e.deptno = d.deptno
group by Rollup (dname)

o/p:-

Dname	sum(sal)
A/c	9050
Research	11275
Sales	9600
	29925

Outer Join :-

This join is used to retrive all rows from one table and matching row from another table.

→ generally wherever we are useing equi join we are retrive matching rows only.

→ If we want to retrive non matching rows also then we are useing 'JOIN' operator (+) with in equi join. This is called oracle 8i outer-Join.

→ This Join operator can be used only one side at a time in joining conditions.

Syntax :-

eg:- Select ename, sal, d.deptno, dname, loc from
emp e, dept d
where e.deptno (+) = d.deptno

matching rows → 40 operations

→ All rows display → BOSTON

Self join :-

- joining a table to itself is called <u>self join</u>.
- here joining conditional cols must belongs to same datatype.
- if we want to compare two different coln values in a single values then we must use self ~~join~~ join
- here these two col must belongs same data-type.
- If we want to compare single col value also then we must use self join.

Emp

eno	eno1
10	10
20	30
30	40

two diff coln compare in a single col.

Emp

eno.	
10	
20	
30	

one coln value is compare to each other then we are use self join.

- in databases whenever we are referencing a single table more then one time then we must use table aliashes in self join or we are referencing a table more then one table that why we must create aliashname in form clause
- These aliashname creates memory at the time of execution of the query.

→ These aliashname must be diff^n name otherwise DB server returns a ambiguity.

syntax:—

From tablename aliashname, tablename aliash name2

eg: SQL> from test a, test b where a.city = b.city and a.name = 'murali'.

SQL> create table test (name varchar2(10), city varchar2(10));

SQL> insert into test ('soma', 'BBSR');

SQL> select * from test;

name	city
soma	hyd
Reema	sbp
omkar	BBSR

Q: W.A.Q to display the employee who are working in same city as murli is working, using self join

SQL> select b.name, b.city from test a, test b where a.city = b.city and a.name = 'murali'.

name	city
murali	hyd
ZZZ	hyd.

test.a

name	city
murali	x

test.b

name	city
?	x

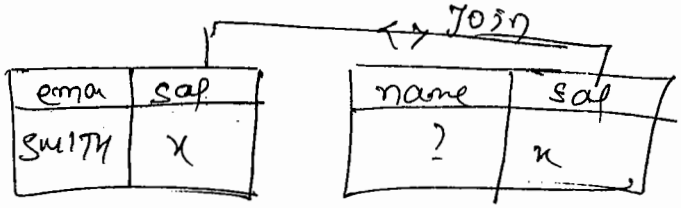
Join.

SQL> Update emp

Q: W.A.Q to display the employee who are getting same salary as 'smith' employee from emp table.

(4.00PM) 10/02/14

Q) From emp e_1, emp e_2 where e_1.sal = e_2.sal AND e_1.name = 'SMITH'



Sol) Select e_2.ename, e_2.sal from emp e_1, emp e_2 where e_1.sal = e_2.sal AND e_1.ename = 'SMITH'.

Mar Q: write a query to display employee name, & there Manager names from emp table using self Join.

Q) Sol) Select e_1.name "employees", e_2.name "manager" from emp e_1, emp e_2 where e_1.mgr = e_2.empno.

or

Sol) Select e_1.name "employees" e_1.mgr, e_2.empno, e_2.ename "managers" from emp e_1, emp e_2 where e_1.mgr = e_2.empno.

Q: w.A.Q to display the employee who are getting more then there Manager salaries from emp table using self Join.

Q) Sol) Select e_1.ename "employees", e_1.sal, e_2.sal, e_2.ename "managers" from emp e_1 emp e_2 where e_1.mgr = e_2.empno and e_1.sal > e_2.sal

Q: WAQ to display the employees who are join before there managers from emp table useing self join.

A: Select e_1.ename "employees", e_1.hiredate, e_2.hiredate, e_2.ename "managers" from emp e_1, emp e_2 where e_1.mgr = e_2.emp and e_1.hiredate < e_2.hiredate

Q: W.A.Q to display the employies who are getting same salary in different dept from emp table useing ~~same~~ self join.

A: sol: Select e_1.ename, e_1.sal, e_1.deptno, e_2.ename, e_2.sal, e_2.deptno
where e_1.sal = e_2.sal
and e_1.deptno <> e_2.deptno.

o/p:-

ename	sal	deptno	ename	sal	deptno
SMITH	1500	30	MILLER	1500	10

Q: WA.Q to display the employees who are joining in same mongth and same day of the week from emp table useing self join.

A: Select e_1.ename, e_1.hiredate, e_2.ename, e_2.hiredate from emp e_1, emp e_2
where to-char(e_1.hiredate, 'DMON') = to-char(e_2.hiredate, 'DMON') and
e_1.empno < e_2.empno.

o/p:-

ename	hiredate	ename	hiredate
JAMES	03-DEC-81	FORD	03-DEC-81

Non-equijoins :~

Based on otherthen equality Condition (<>, <, <=, >, >=, in, between) we are retrieve data from multiple tables.

→ This joins is also called as between and join.

→ when tables does't have common cols then only we are this join.

sql> select ename, sal, losal, hisal from emp, salgrade where sal>= losal and sal<= hisal ;

(OR)

sql> select ename, sal, losal, hisal from emp, salgrade where sal betn losal and hisal;

9: Joins (OR) ANSI Joins :~

1) inner join.
2) left outer join.
3) right outer join.
4) Full outer join.
5) natural Join.

1) Inner Join :~

This join also return Matching row only.

→ when ever tables heaving common cols thenonly we are useing inner joins.

→ Here also joining conditional cols must belongs to same datatypes.

→ This join performs is very high compare to oracle 8i inner join equi join.

sql> select ename, sal, d.deptno, dname, loc from emp e join dept d on e.deptno = d.deptno.

in place of where.

Q: WAQ to display the employees who are working in a location chicago from emp, dept tables using ql inner joins.

A) Select ename, loc from emp join dept on emp.deptno = dept.deptno AND where loc = 'CHICAGO'.

* → in ql in place of 'and' we can use 'where' but in 8i it will not worked.

o/p:-

Ename	loc
WARD	CHICAGO
TURNER	CHICAGO
ALLEN	CHICAGO

NB:- we can also use 'using' clause in place of on clause in ql joins.
generally 'using' clauses performance is very high compare to 'on' clause.

eg:-
sql> Select ename, loc from emp join dept using (deptno) where loc = 'CHICAGO'.

Mar? NB:- Always using clauses returns common cols one time only. 11/02/14

eg:- Select * from t1;

A	B	C
x	y	z

select * from t2;

A	B
x	y

say select * from t_1 join t_2 on $t_1.a = t_2.a$ and $t_1.b = t_2.b$ } useing join s on.

o/p :-

A	B	C	A	B
x	y	z	x	y

say select * from t_1 join t_2 useing(a, b); } useing 'useing' clause

o/p :-

A	B	C
x	y	z

N:B (Rule)

whenever we are useing 'useing' clause we are not allowed to use alias name on joining conditional col.

eg:- say select ename, sal, deptno, dname, loc from emp e join dept d useing(deptno).

leftouter join :-

leftouter join returns all rows from left-side table and matching rows from right side table and also returns null values in place of non-matching rows in another table.

eg:- select * from t_1;

o/p.

A	B	C
x	y	z
p	q	r

select * from t_2;

o/p :-

A	B
x	y
s	t

sql> select * from t_1 left outer join t_2
on $t_1 \cdot a = t_2 \cdot a$ and $t_1 \cdot b = t_2 \cdot b$;

o/p :-

A	B	C
x	y	z
p	q	r

A	B
x	y
□	□

→ matching row are only show.

left side all data are show.

↳ non matching value are null value displayed.

Right outer join :-

~~eg:- sql> select~~

→ it returns all rows from rightside table and matching rows from left side table and also returns null values in place of non-matching row in another table.

eg:- sql> select * from t_1 right outer join t_2
on $t_1 \cdot a = t_2 \cdot a$ and $t_1 \cdot b = t_2 \cdot b$;

o/p :-

A	B	C
x	y	z

A	B
x	y
s	t

Full outer join :-

It is a combination of left, right outer join. it returns matching and non matching rows

from both the table and also returns null values in place of non-matching rows.

eg:- sql> select * from t_1 full outer join t_2 on $t_1 \cdot a = t_2 \cdot a$ and $t_1 \cdot b = t_2 \cdot b$;

o/p:-

A	B	C	A	B
x	y	z	x	y
p	q	r	□	□
□	□	□	s	t

Natural join :-

→ This join also returns matching rows only.

→ This join performance is very high compare to inner join.

→ when tables heaving common col. then only we allowed to use 'natural join'. when ever we specify the natural join internally DB server only establish joining condition that way explicitly we are not allowed to use joining condition.

→ This join internally uses 'using' class that why this join returns common col. one time only and also we are not allowed to use alish name on joining conditional col.

eg:- select * from t_1 natural join t_2;

o/p:-

A	B	C
x	y	z

sqly Select ename, sal, deptno, dname, loc from emp e natural join. dept d.

Cross Join:-

sqly Select ename, sal, dname, loc from emp cross join dept.

Join more then two tables

eg:- (Join 3 tables)

8i Joins

syntax:-

select col_1, col_2, col_3 ——
from $tablename_1$, $tablename_2$, $tablename_3$
where $tablename_1$.commoncol = $tablename_2$.commoncol
AND
$tablename_2$.commoncol = $tablename_3$.commoncol;

9i Joins

synax:-

Select col, col, col ——
from $tablename_1$ join $tablename_2$ on $tablename_1$.commoncols = $tablename_2$.commoncol
join $tablename_3$ on $tablename_2$.commoncol = $tablename_3$.commoncol;

~: CONSTRAINTS :~

Constraints are used to Prevents invalid data entry into our tables.

→ Constraints are created table columns.

→ In DB if we want to maintain Proper data then we must defines set of rules. These Rules are also called as <u>Business Rules</u>.

→ if we want to defined these business rules all DB system uses <u>constraints</u>, <u>triggers</u>.

→ Oracle server heaving following constraints is :

1) Not null.
2) unique.
3) Primary key.
4) Foreign key.
5) check.

Note: All the above constraint are defined two ~~tables~~ levels

1) column level
2) Table level

<u>Column level</u> :~

In this method we are defining constraint on individual columns that is whenever we are defining col then only we are specifying constraint type.

syntax:-

```
create table tablename (col1 datatype(size)
constraint type, col2 datatype(size), - - - - );
```

Table Level :-

in this method we are defining constraints on group of columns that is:

1st we are defining all col and last only we must specify constraint type along with group of cols.

syntax:-

```
create table tablename (col1 datatype (size), col2
datatype (size), ....., constraint type (col1, col2 ---));
```

NOT NULL constraint :-

→ it does't support table level.

→ This constraint does't accepts null values but it accepts duplicate values.

column Level :-

→ sql> create table x1 (sno number(10) notnull, name varchar2(10));

Table created.

sql> insert into x1 values (null, 'abc');

Error:-1400 : cannot insert NULL into SNO.

Unique constraint :-

→ Unique constraint ~~depend~~ defind col level or table level.

→ This constraint does not accepts duplicate values but it will accepts null values.

→ whenever we are creating any constraints oracle server automaticaly creates btree indexes on those columns.

col level :-

SQL> create table tablename X_2(sno number(10) unique, name varchar2(10));

Table level :-

SQL> create table tablename X_3(sno number(10), name varchar2(10), unique (sno, name));

SQL> insert into x_3 values (1, 'Murali');
1 row inserted.

SQL> insert into x_3 values (1, 'abc');
inserted.

SQL> insert int x_3 values (1, 'abc');
error:- 00001 : unique constraint violated.

SQL> select * from x_3

SNO	name
1	Murali
1	abc

Primary key :-

→ Primary key uniquely a identifies record in a table. ~~where~~ There can be only one prime key in a table and also Primary key does not accepts duplicate, NULL values.

→ whenever we are created a Primary key internally oracle server creates btree index on those columns.

column levels :-

syntax :-

create table ~~tablename~~ x_4 (sno number(10) primary key, name varchar2(10));

Table level :-

```
create table xs( sno number(10), name varchar2(10),
Primary key (sno, name));
```

→ Table created;

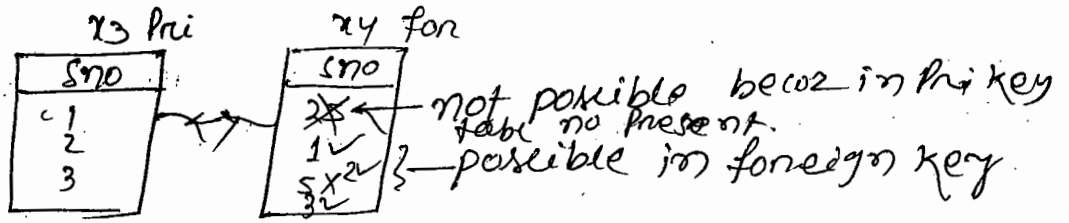
→ This is also called as composit Primary key i.e. It is a combination of cols as a single Primary key.

Foreign key :-

→ If we want to establishes relationship between tables then we must use referential integrity constrant foreign key.

→ one table foreign key must belongs to another table Primary key and also these two cols must belongs same datatype.

→ Always foreign key values based on Primary key values only.



→ generally Pri key doesn't accepts duplicate, null values where as foreign key accects duplicate, null values.

col level :- (References)

→ syntax :-

```
create table tablename (col1 datatype (size)
references mastertablename (Primary key(colname),
col2 datatype (size), ....);
```

sql> create table z4 (sno number(10) references x4);
Table created.

(OR)

sql> create table y4 (a number(10) references x4 (sno));

Table level :- (foreign key, references)

syntax :-

```
create table tablename (col1 datatype(size),
col2 datatype (size), ...., foreign key(col1,
col2) references mastertablename(Primarykey-
colnames));
```

sql> create table z5 (sno number(10), name varchar2(10), col3 number(10), foreignkey (sno, name) references x5)

/

Mar 7

15/02/14

when ever we are establishing relationship between table oracle violates following two rules:

1) Deletion in Master.
2) insertion in child.

1) Deletion in Master :-

If we are try to delete a record in Master table then oracle server returns an error ora-02292.

→ To overcome this Problem 1st we are deleting child table records in child table then only we are deleting corresponding in Master table record in M.T

Otherwise use an <u>on delete</u> cascade class.

On delete cascade :-

when ever we are useing this clause in child table if we are deleting a M.T record in M.T then automatically Mastertable record deleted in M.T and also corresponding C.T record also deleted in child table.

syntax :

```
create table tablename (col1 datatype(size)
references mastertablename (Primarykey column)
on delete cascade, col2 datatype (size)...);
```

eg:- sql> create table Master (sno number(10) Primary key);

sql> insert into Master Values (1);

sql> select * from mas;

SNO
1
2
3

sql> create table child (sno number(10) referenties Master(sno) on delete cascade);

sql> insert into child values (...);

sql> select * from child;

SNO
1
1
1
2
2
3

Testing :-

1) Deletion in Master

sql> delete from master where sno = 1;
1 row deleted.

sql> select * from master; } ← row automatically
sql> select * from child; } deleted.

→ Oracle also supports another clause ~~on delete~~ "on delete set null".

On delete set Null:-

whenever we are using this clause in child table, if we are deleting M.T record then automatically corresponding foreign key value set to null in child table.

Syntax:-

```
create table tablename (col datatype(size)
references Mastertablename (Primarykeycol name)
on delete set null);
```

2) Insertion in child:-

If we are try to insert other then P.K values into foreign key then oracle server returns an error ora-02291, becoz in DB systems always foreign key values based on P.K values only.

Check :-

This constraint is used to defined logical conditions according to our business rules.

column level :-

syntax :-

```
create table tablename(colname datatype(size)
check(logical condition), ....);
```

eg:- sql> create table test(sal number(10) check(sal > 5000));

Table created.

eg:- sql> create table test1(name varchar2(10) check (name = upper(name)));

sql> insert into test1 values('soma');

Error: check constraint (SCOTT.SYS_C005489) violated.

→ Predefined constraint name.

Table level :-

sal> create table test1(name varchar2(10), sal number(10), check (name = upper(name) and sal > 5000));

Assigns user defined names to constraints :-

When ever we are creating a constraints DB servers automatically generates an unique identification no. for identifying constraint.

→ In oracle, oracle server automatically generates an unique identification no. in the format of SYS_Cn this is called predefined constraint name.

→ we can also create our own name in place of SYS_Cn useing constraint keyword.

→ This is called <u>user defined constraint name</u>

syntax:

constraint	userdefinedname	constraintype

↳ constraint name

↳ notnull, Primary k, foreign k, unique, check

eg1:- say create table tests (sno number (10), Primary key);

say insert into tests values (1);

say insert into tests values (1);

error: unique constraint (SCOTT.SYS_C005491) violated.

eg2:- say create table ~~table6~~ test (sno number (10) ~~size~~ <u>constraint P_abc Primary key</u>);

↳ user defined constraint name

say insert into test6 values (1);

say insert into test6 values (1);

error: unique constraint (SCOTT.P_ABC) violated.

Mgd ?

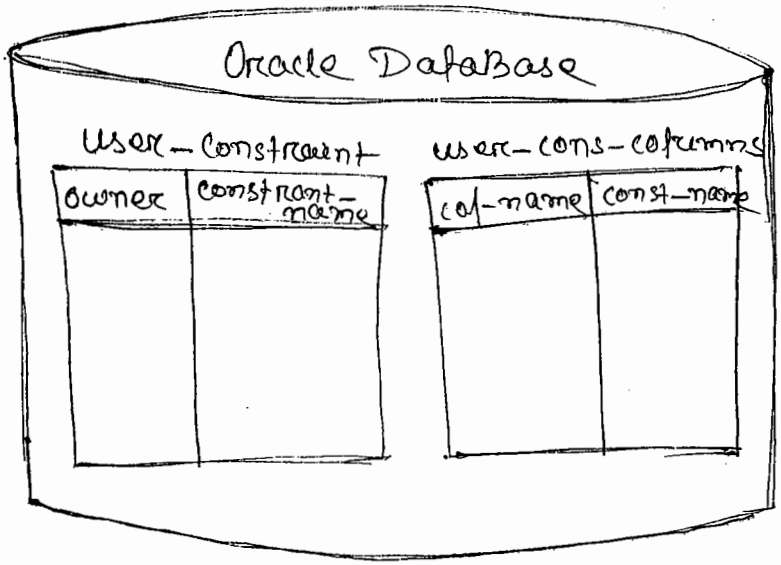
14/02/14

when ever we are installing oracle server automatically readonly tables are created in DB. These tables are also called as Data Dictionaries.

→ (After diagram) diagram (1.1)

→ All constraint information stored under <u>user_constraint</u> Data Dictionary

say desc user_constraints;



<diagram 1.1>

SQL> Select constraint_name, constraint_type from user_constraints where table_name='EMP';

(capital letter only)

O/P :-

constraint_name	constraint_type
PK_EMP	P
FK_DEPTNO	R

NB:- If we want to view col names along with constraints name then we are useing ~~user~~ user_cons_columns Data dictionary.

eg:- SQL> Desc user_cons_columns;

SQL> Select constraint_name, column_name from user_cons_columns where Table_name='EMP';

O/P :-

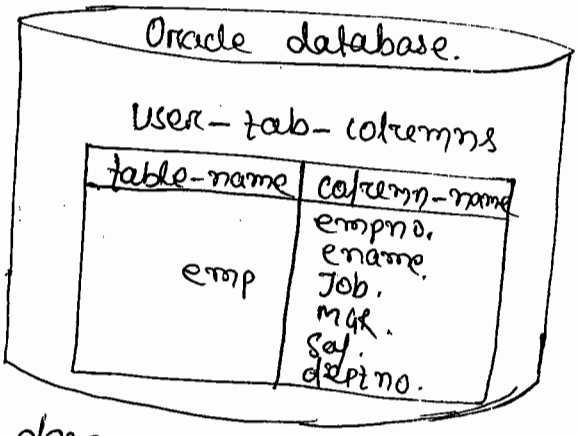
constraint_name	column_Name
FK_DEPTNO	DEPT NO
PK_EMP	EMPNO

NB:- If we want to view logical condition of the check constraint then we are useing Search_condition Properties from user_constraint Data dictionary.

sqly select search_condition from user_constraints where table_name = 'TEST'.

o/p:- Search_condition
name = upper(name)

⇒ In oracle all columns information stored under user_tab_columns data dictionary.



eg:- sqly desc user_tab_columns;
sqly select column_name from user_tab_columns where table_name = 'EMP';

~~eg: sqly select count(*) from~~

* W.A.Q to display no. of columns number from emp table.
sqly select count(*) from user_tab_columns where table_name = 'EMP'

Mar 7

15/02/14

Default constraint:-

Oracle also support default constraint to save default values on a column.

syntax:-

colname datatype (size) default actualvalue;

sqly create table test(name varchar2(10), sal number (10) default 2000);

sqly insert into test(name) values('abc');

sqly select * from test;

name	sal

NB:- if we want to view default values for a col then we are useing, data_default Properties from user_tab_cols data dictionary.

sqly desc user_tab_columns;

eg:- sqly select column_name, data_default from user_tab_columns where table_name = 'TEST';

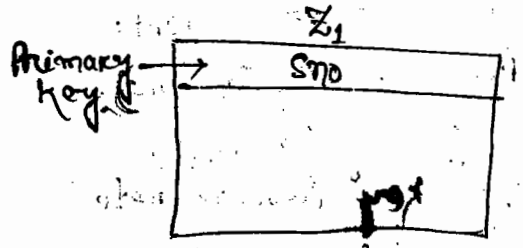
o/p:-

column_name	data_default
sal	2000

Adding, droping constraints on existing tables:-

useing alter we can also add and drop constraint on existing table and existing column.

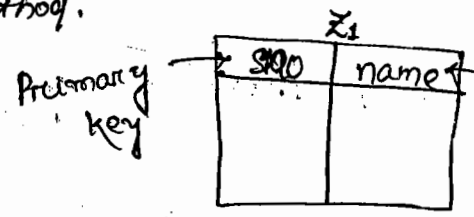
NB if we want to add constraint of exisisting table existing column then we are useing table level syntax method.



sqly create table Z1 (sno number(10));

sqly alter table Z1 add Primary key (sno);

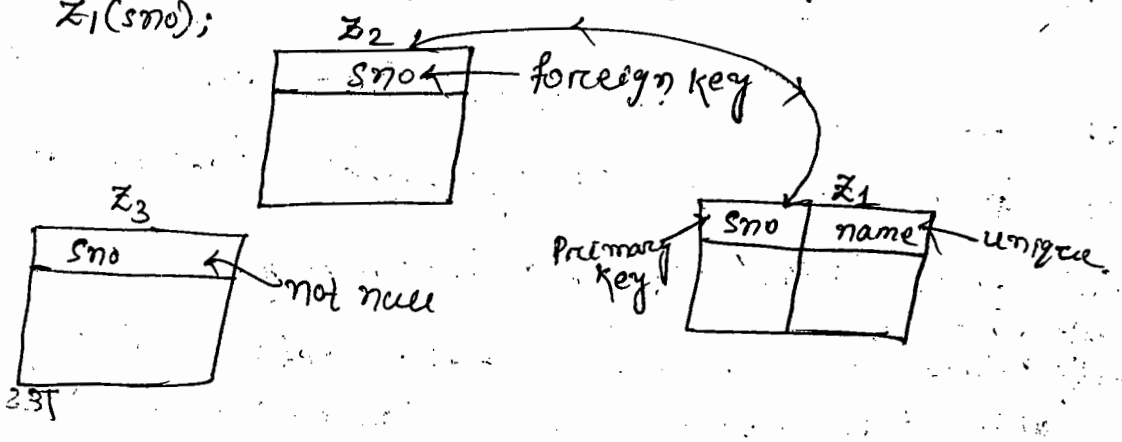
NB:- if we want to add a new column along with constraint then we are useing column level syntax method.



sql> alter table z_1 add name varchar$_2$(10) unique;

sql> create table z_2 (sno number(10));

sql> alter table z_2 add foreign key (sno) references z_1(sno);



NB:- if we want to add not null constraint of existing table and existing column then we are using alter with modify.

sql> alter table tablename modify column not null;

eg:- alter table z_3 (sno number(10));

sql> alter table z_3 modify sno not null;

sql> alter table z_3 add new varchar$_2$(10) not null;

sql> create table emp1 as select * from emp;

sql> create table dept1 as select * from dept;

sql> alter table dept1 add Primary key (deptno);

sql> alter table emp1 add Primary key (empno);

sql> alter table emp1 add foreign key (deptno) references dept1 (deptno) on delete cascade;

Dropping constraints :-

syntax :-

method-1 alter table tablename drop constraint constraintname;

(OR)

method-2 alter table tablename drop Primary key

sql> create table z10 (sno number(10) Primary key)

sql> alter table z10 drop Primary key;

NB:- if we want to drop Primary key along with referential foreign key then we are useing cascade class alonge with alter, drop.

syntax:-

alter table tablename drop Primary key cascade

sql> alter table z1 drop Primary key;

Error: this unique/Primary key is referenced by some foreign key.

sql> alter table z1 drop Primary key cascade;

mar 7 sql> create table w1(sno number(10) constraint ab not null; 17/02/14

sql> alter table w1 drop constraint ab;

Subqueries:-

Query with a another query is called nested query or subquery.

→ generally child queryes are also called sub queries.

→ These queries are used to retrive data from single or multiple table based on more then one step Process.

→ All DB System having two types of sub queries

1) Non-correlated sub queries.

2) co-related subqueries.

→ in non-correlated subquery child queryes is executed 1st, then only Parent queryes executed.

→ where as a co-related subqueryes Parent queryes is executed 1st then only child query is executed.

Child query :-

A query which provide values to another query is called child query.

Parent query

A query which received values from another query is called parent query.

1) Non-correlated subqueries :-

Non-correlated subqueries
- Single row subquery
- Multiple row subquery
- Multiple column subquery.
- In Line view (OR) subquery used in (from clause).

Q. W.A.Q to display the employee who are getting more then the average salary from emp table.

A. select empname, empsal from emp where sal > (avg(sal)); wrong.

Blunder Mistake

Q. select * from emp where sal > (select avg(sal) from emp);

This is an single row sub queries because there child query return single value

→ in single row subqueries we are useing =, ≠, <, >, ≤, ≥, operators.

execution

step1: sql> select avg(sal) from emp;

step2: sql> select * from emp where sal > 2216.07743;

Q: W.A.Q to display the employees who are working with sales dept from emp, dept tables using Non-correlated subqueries. (without Join condition).

A: Select * from ~~dept~~ emp where deptno = (select deptno from dept where dname = 'SALES');

Q: W.A.Q to display the employees who are working as same as SMITH dept no.

A: Select * from emp where deptno = (select deptno from dept where dnam = 'SMITH');

Q: ~~QAW~~ W.A.Q to display eirearmost employee details from emp table.

A: Select * emp where Hiredate = (select min(Hiredate) from emp);

Mq: W.A.Q to display the employee who are working under BLAKE from emp table using empno, mgr columns. 18/02/14

A: select * from emp where mgr = (select empno from emp where ename = 'BLAKE');

Q: W.A.Q to display lowest avg. sal job from emp table using group by.

A: Select job, avg(sal) from emp group by job having avg(sal) = (select min(avg(sal)) from emp);

Error: Nested group by function without GroupBy.

Rm:
when a child query contains nested group functions then we must use group by caluse.

Sol: Select job, avg(sal) from emp group by job having avg(sal) = (select min(avg(sal)) from emp group by job);

say select Job, avg(sal) from emp group by Job having avg(sal) > (select avg(sal) from emp where Job = 'CLERK');

JOB	AVG(SAL)
President	5300
Manager	2825
Analyst	3050

say select deptno, min(sal) from emp group by deptno having min(sal) > (select min(sal) from emp where deptno = 30);

Deptno	Min(sal)
20	1100
10	2650

Q. to display employee details who are getting max salary in each department from emp table.

say Select * from emp where sal = (select max(sal) from emp group by deptno);

note: Single-row subquery returns more than one row.

* This is a multiple row subquery bcoz here child query returns more than one value. In multiple row subqueries we are using in, all, any operators.

* we can also use in operator in single row subqueries.

say Select * from emp where sal in (select max(sal) from emp group by deptno);

(OR)

say Select deptno, sal, ename from emp where sal in(select max(sal) from emp group by deptno);

DEPTNO	SAL	ENAME
30	2850	BLAKE
20	3100	FORD
10	5300	KING

Q: W.A.Q to display the employee who are working on either sales or research deptment from emp, dept table.

A: Select * from emp where deptno In (select deptno from dept where dname = 'SALES' OR dname='RESEARCH');

TOP-N analysis :-

19/02/14

1) inline views
2) rownum.

inline views :-

Oracle 7.2 introduced inline views.

→ Generally we are not allowed to use order by class in child query. to overcome this Problem oracle 7.2 introduced some query is from clause, this type query are called as inline view.

syntax:-

```
select * from (select statement);
```

Rownum :-

Rownum is a pseudo column it behavious like a table column.

→ Generally pseudo col are used in all tables.

→ Rownum automatically assigns nos. to each row in a table at the time of selections.

eg:- say select rownum, ename from emp;

Rownum	ENAME
1	SMITH.
2	ALLEN.
3	WARD.
4	JONES.

These are Temporary values

→ Rownum heaving temporary values.

example:-

say Select rownum, ename from emp where deptno = 10;

o/p:-

Ronum	ENAME
1	CLARK
2	MILLER.

Q: W.A.Q to display 1st row from emp table useing rownum.

A: say Select * from emp where rownum = 1;

work

say Select * from emp where rownum = 2;

not work o/p: no row selected.

Q: W.A.Q to display 2nd row from emp table useing rownum.

A:

NB: Generally rownum does not work with more then one +ve integers i.e it works with <, ≤ operators.

Q: W.A.Q to display 1st 5 rows from emp table useing rownum.

A: say select * from emp where rownum < = 2

Q: W.A.Q to display ~~5th~~ 1st 5 higest sal ~~empty~~ from emp employee table useing rownum.

A: say select * from (select * from emp order by sal desc) where rownum<=5; emp.

Q: W.A.Q to display 5th higest salary employee from emp table useing rownum.

A: say select * from (select * from emp order by sal Desc) where rownum<=5
MINUS
select * from (select * from emp order by sal desc) where rownum<=4.

Q: W.A.Q to display second row from emp table useing rownum.

A: Select * from emp where rownum<=2
Minus
Select * from emp where rownum<=1.

say select * from emp where rownum between 1 and 5;
working.

say select * from emp where rownum between 2 and 5;
notworking.

say select * from emp where Rownum>1:
not working.

say select * from emp where rownum>=1;
working.

Q: W.A.Q to display the row betn 1to5.

A:

24/02/14

Q: WAQ to display the rows between 3 and 9 from emp table useing rownum.

A: Select * from emp where rownum <=9
Minus
select * from emp where rownum <=3.

Q: WAQ to display last two rows from emp table useing row-num.

A:
```
select * from emp ~~where rownum~~
Minus
select * from emp where rownum <=(select
count(*) -2 from emp);
```

NB:- when we are creating a reference name for rownum in inline use then that reference name works with all sql operators.

Q: WAQ to display 5th row from emp table useing rownum alias name.

A: select * from (select rownum r, ename, sal from emp) where r=5;

O/P :-

R	ENAME	SAL
5	MARTIN	1150

(OR)

sol> Select * from (select rownum r, e. * from emp e) where r=5;

Q: WAQ to display 1st row, last row from emp table useing rownum alias name.

A: sol> Select * from (select rownum r, ename, sal from emp) where r=1 or r (select (*) from emp);

O/P :-

R	ENAME	SAL
~~SMITH~~		
1	SMITH	1100
12	MILLER	2700.

Q: W.A.Q to display even no. of rows from emp table useing rownum aliash name.

Select * from (select rownum r, ename, sal from emp) where mod (r, 2) = 0;

Q: W.A.Q to display 5th higest salary employee from emp useing rownum aliash name.

Select * from (select rownum r, ename, sal from (select * from emp order by sal desc)) where r = 5; (sn)

use when it need 1, 2, 3 ....

O/P :-

R	EName	sal
5	MILLER	2700

analytical functions used in inline views :-

Oracle 8i introduce analytical function these are :-

1) row-number()
2) rank()
3) dense-rank()

→ These 3 analytical functions automatically assignes ranks (1, 2, ... n) either groupwise or rowwise in a table.

Syntax :- Analytical function name() over (Partation by colname order by colname [asc/dsc])

Optional Part

→ row-number() analytical function automatically assign different rank numbers when values are same. where as rank, desc-rank() analytical fun automatically assigns same rank nos when values are same. But rank skips next consecutive to rank number. where as dense-rank() does not skips next consecutive to next rank number.

eg:- row-number():-

Deptno	Ename	sal	R
30	BLAKE	1150	1
30	TURNER	1150	2
30	WARD	1000	3
30	MARTIN	950	4

Rank():-

Dept	Ename	sal	R
30	BLAKE	1150	1
30	TURNER	1150	1
30	WARD	1000	3

good.

Dense-rank():-

Dept	Ename	sal	R
30	BLAKE	1150	1
30	TURNER	1150	1
30	WARD	1000	2

Mag ?

```
> select * from (select deptno, ename, sal, row-number()
  over (Partition by deptno order by sal desc)
  r from emp) where r<=10;
```

(alias name)

Q: W.A.Q to display second higest Salary employee in each dept from emp table useing analytical functions.

A) > select * from (select deptno, ename, sal, ~~row_number()~~ dense_rank() over(Partition by deptno order by sal desc) r from emp) where r<=2;

o/p :-

Deptno	ename	sal	R
10	MILLER	2700	2
20	JONES	2975	2
30	TURNER	1400	2

Q: W.A.Q to display 5th higest salary employee from emp table useing analytical function (dense_rank()).

A) select * from (select deptno, ename, sal, dense_rank() over(~~Partition by emp~~ deptno order by sal desc) r from emp) where r<=5;

o/p :-

Deptno	ename	sal	R
10	MILLER	2700	2

Mgr Nb:-

Same like a normal function.

25/02/14

Analytical functions also accepts Parameters.

eg:- lead():-

This functions computes an expressions on next rows and return values to the current row.

Syntax:-

lead(expression , offset , default) over(Partitions by colname order by colname [asc/desc])

(expression ↑ means col name; offset, default = numbers)

→ Here offset returns next immediate row. And also default returns default values, if next immediate row not available with in that group.

ex:-

```
> Select deptno, ename, sal, lead(sal, 1, 0)
  over(Partition by deptno order by ename) r
  from emp.
```

lag:-

This function comprises an expressions on the Previous row and returns values in to current row.

```
Select deptno, ename, sal, lag(sal, 1, 0)
over (Partition by deptno order by ename) r
from emp.
```

o/p:-

Q: W.A.Q to display the year and no. of employees per year in which more than one employee was hiole from emp table.

A)
```
> Select to_char(hiredate, 'yyyy') "year", count(*)
  from emp group by to_char(hiredate, 'yyyy')
  heaving count(*) > 1
```

o/p:-

year ~~To_X~~	count(*)
1987	2
~~1980~~	
1981	8
~~1982~~	

Q: W.A.Q to <u>display duplicate data</u> from a table.

> create table test(sno number(10));

> Select sno, count(*) from test group by sno. having count(*) >1.

o/p:-

SNo	count(*)
30	3
20	2
40	1
50	1
10	4

← Duplicate data.

o/p:-

SNo	count(*)
30	3
20	2
10	4

> select sno, count(*) from test group by sno.

o/p:-

~~Q: W.A.Q~~ **Rowid:-** Rowid is pseudo col it behavios like a table col when we are inserting data into table oracle server automatically generates and unique identification number for identifying record uniquely this no. is also called row add or row id.

→ Generally rownum having temporary values where as rowid having fixed value.

eg:- > Select rownum, rowid, rowname from emp;

sql> Select rownum, rowid, ename from emp where deptno =10;

NB:- We can also use Max, Min functions in rowid.

sql> select max(rowid) from emp;

o/p:- rowid

AAAMAZAAAGAAA.

sql> select min(rowid) from emp;

> Generally rowid are used to delete duplice row except one row in each group.

* Q: co. A. Q to delete duplicate rows except one row in each group from a table using rowid.

> ~~Select~~ delete from test where rowid not in (Select Max(rowid) from test group by sno);

> Select * from test;

SNO
10
20
30
40
50

Multiple column subqueries:-

We can also compare multiple col values of the child query with the multiple col of the Parent query.

→ These type of query are also called as Multiple col ~~of~~ ~~to~~ subqueries.

→ in this type of queries we must specified Parent query where conditional col with in Parenthesis.

syntax:-

```
Select * from table col name ~~where~~
where (colname1, colname2 ....) in (select colname1,
colname2 .... from tablename where condition);
```

Q: WA.Q to display the employees whose Job, MGR match with in Job, MGR of of employee SCOTT from emp tables.

> Select * from emp where (Job, MGR) in (select Job, MGR from emp where ENAME='SCOTTE');

O/P:-

EMPNO	ENAME
7788	

Q: W.A.Q to display the employee who are getting max sal ~~from emp~~ in each dept from emp table ~~in~~ ~~wit~~ ~~each~~ ~~dept~~ useing multiple row subquery.

> Select Deptno, sal, ename from emp where sal in (select max(sal) from emp group by deptno;

O/P:-

deptno	sal	ename
30	2750	BLAKE
20	3000	SCOTT
20	2750	FORD
10	5000	KING.

Q: W.A.Q to display the employees who are getting max sal in each dept from emp table useing multiple col subquery.

> Select deptno, sal, ename from emp where (deptno, sal) in (select deptno, max(sal) from emp group by deptno)

deptno	sal	ename
30	2750	BLAKE
20	3000	SCOTT
10	5000	KING.

Mag 7.

28/02/14

Q: W.A.Q to display e name, dname salary of the employee who salary, commition match with the sal, commition with the employees working in the location "DALLAS" useing emp, dept table.

A: Select ename, dname, sal from emp e, dept d where e.deptno = d.deptno and (sal, nvl(comm,0)) in (select sal, nvl(comm,0) from emp e, dept d where e.deptno = d.deptno and d.loc = 'DALLAS')

Q: W.A.Q to display the employees who are getting more then the higest paid employee in 20th dept from emp table.

A: Select * from emp where sal > (select max(sal) from emp where deptno = 20);

O/P:-

Empno	ename	JOB	MGR	sal	com	deptno
7838	King	Clerk	25-Feb	5000		20

Q: W.A.Q to display the employees who are getting more then the lowest paid employee in 10th dept from emp table.

A: Select * from emp where sal > (select min(sal) from emp where deptno = 20);

when ever Resource table large amount of data & also when we are comparing no. of values & also if child query contains max or min functions then those type of subqueries d grates performance of the application. To overcome this proble all data bases Provided sumqueries special operators to improve performance of the query these sub query

special operators are 'all', 'any'.
These operators are used along with relational operators.

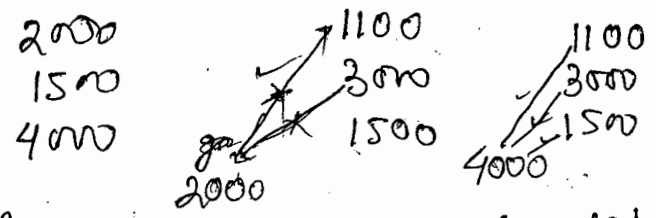
1) Select * from emp where sal > all (select sal from emp where deptno = 20);

2) Select * from emp where sal > any (select sal from emp where deptno = 10);

'all', 'any' operators used in multiple row subquery

IN → it returns same values in the list → child query.

All → it satisfy all values in the list.



ANY → it satisfy any value in the list.

IN → = ANY.

↳ when every we are using subquery special operator 'any', optimizer internally uses logical operator 'AND' also ~~we are using subqueries~~ special operator 'all' optimizer internally uses logical operator 'OR'.

1) select * from EMP where Deptno > All (10, 20);
) select * from emp where Deptno > any (10, 20);

↳ In operator is also same as '= Any' but not > ANY.

Sal) select * from emp where deptno = any (10, 20);

Co-Related subqueries :-

generally in noncorrelated subqueries childquery is executed first then only parent query executed.

where as a correlated subquery parent query is executed 1st then only child query executed.

↳ in correlated subqueries we must create reference or alias name for the parent query table in parent query and pass this alias name into child query where class.

syntax:-

```
select * from tablename aliasname
where colname = (select * from tablename
where colname = aliasname.colname);
```

eg:- select * from emp e where sal = (select * from emp where sal = e.sal.

when we are submitting correlated subqueries database servers get a candidate row from parent query table and then control passed into child query where clause is based on evaluation value it compares values in parent query.

↳ in non correlated subqueries child query executed only once path parent table. where as in correlated subqueries child query is executed for each and every row in parent query table.

27/02/14

Q? Generally correlated subqueries are used in denormalization process.

↳ in this process we are useing correlated update generally if we want to modify one table col value based on another related table then we must use correlated update.

syntax :—

```
update tablename1 aliashname1
set colname =(select colname from tablename2
              aliashname2 where aliashname1.
              common colname = aliashname2.
              commonconame.
```

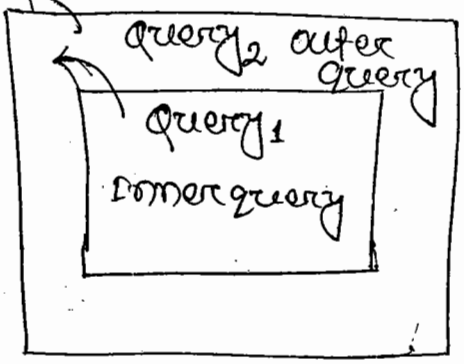
> alter table emp add dname varchar2(10);

> update emp e set dname =(select dname from dept d where e.deptno = d.deptno);

> select * from emp;

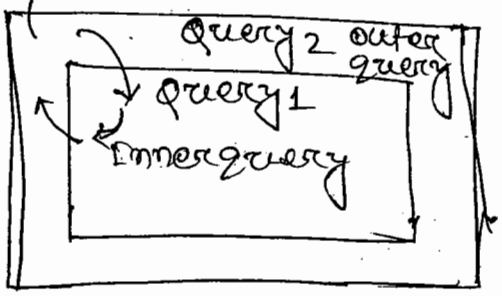
noncorelated subquery

Final output.



corelated subquery.

Final output.



> Q: w.A.Q to display 1st higest salary emplaye from emp table useing corelated subquery.

> Select * from emp e1 where 1 =(select count(*) from emp e2 where e2.sal >= e1.sal);

o/p:- King 50,000.

Phase1:

~~step~~ step: ~~get~~ a ~~candidate~~ row ~~(first row)~~ →

Q:> w.a.q to display 2nd hegest sal employee from the following table useing correlated subquery.

> Create table emp1(Ename varchar2 (10), sal Number (10));

> Select * from emp1;

Ename	sal
abc	100
xyz	150
zzz	200
bbb	300

> Select * from emp1 e1 where 2=(select count(*) from emp1 e2 where e2.sal >= e1.sal);

Ename	sal
zzz	200

execution :-

Sol> Select * from emp1 e1 where 2=(select (*) from emp2. e2 where e2.sal >= e1.sal);

Phase1 :-

step1 :- get a candidate row (first row)(abc 100)

step2 :- Select count (*) from emp1 e2 where e2.sal >= 100

step3 :- select * from emp1 e1 where 2=4 (false).

Phase2 :-

step5 :- get a candidate row(xyz 150)

stepII :- Select count(*) from emp1 e2 where e2.sal >= 150

stepIII :- Select * from emp1 e1 where 2=3 (false)

phase3 :-

step1 :- get a candidate row (zzz 200)

stepII :- Select count(*) from emp1 e2 where e2.sal >= 2

stepIII :- select * from emp1 e1 where 2=2 (true).

zzz	200

← O/P.

> ~~insert~~ insert into emp1 values ('soma', 2000);

> Select * from emp1 e1 where 2 = (select count(*) from emp1 e2 where e2.sal > e1.sal);

o/p :- no row selected.

soln :-

> Select * from emp1 e1 where 2 = (select count (distinct(sal)) from emp1 e2 where e2.sal > = e1.sal)

o/p :-

Ename	sal
ZZZ	200
soma	200

n-1 method :-

> Select * from emp, e1 where (1-1) = select count (distinct (sal)) from emp, e2 where e2.sal > e1.sal);

IN:B

In case of n-1 method we can't use '>=' or '<=', only we can use '<' & '>' depending on required query.

Q: W.A.Q to display nth higest sal employee from emp table useing correlated subquery.

A > select * from emp e1 where sn = (select count (dinstinct (sal)) from emp e2 where e2.sal >= e1.sal);

o/p :- Enter value for n : 1

King 5000

Q: W.A.Q to display the employees who are geting more then the avg salary of their JOB from emp table useing correlated subquery.

A > Select * from emp e1 where sal > (select avg (sal) from emp where job = e.job);

28/02/14

Exists operator :-

"Exists" operator used in co-related sub-queries only. This operator performance is very high compare to "in" operator.

This operator always returns boolean values either true or false.

↳ This operator is used to test weather a given set is ~~used to to~~ empty or non empty. if set is empty then it returns false otherwise it returns true.

Ex:- {1, 2, 3} = true.
{ } = false.

This operator used in 'where' clause. In all database system when we are useing exists operator, operator that we are not allow to used colname in where condition.

Syntax:-

Select * from tablename aliashname where exists (select * from tablename where colname = aliashname colname).

Q: WAQ to display departments from dept table having employee in emp table useing correlated subqueries.
- select * from dept d where (select * from emp where deptno = d. deptno);

Q: W.A.Q to display those dept from dept table not having employee in emp table using correlated subqueries
- select * from dept d where not exists (select * from emp where deptno = d. deptno);

DEPTNO	DNAME	LOC
40	operations	BOSTON

Q: W.A.Q to display those dept from dept table doesnot have employee is emp table using non correlated subqueries.

-select * from dept where deptno not in (select deptno from emp);

DeptNo	DName	LOC
40	operations	BOSTON

→ Generally "not in" operator doesnot work with null values. To overcome this Problem we must use not exits operator in correlated subqueries.

→ insert into emp(empno, deptno) values (1,null);

→ Select * from dept where deptno not in (select deptno from emp);

// no row selected.

→ select * from dept d where deptno not exits in (select * from emp where deptno = d. deptno);

VIEWS

→ views is a database object which is used to Provide authority level of security.

→ Generally views doesn't store data, That's why views are also called as virtual tables.

→ view is also called as window of a table.

→ Generally views are created by database administrator.

→ In all dbs if you want to retrict table columns from one user into another user then must use views.

→ views are created from base tables. Based in the base table all dbs heaving 2 types of views.

1. Simple view
2. complex view (or) join view.

→ Simple view is a view which is created from only one base table where as complex view is a view

which is created from no. of base tables.

Mag 7

01/03/24

→ Conn sys as sysdba;
Enter Password : sys
- grant create any view to scott
- conn scott/tiger.

Simple view :-

sql> create or replace view v1
as
select * from emp where deptno=10;

sql> select * from v1;

DML operations through simple view to base table :-

we can also perform dml operations through simple view to base table based on following restrictions.

1) If a simple view contains group functions, group by clause, distinct, rownum, set operators, joins, then we can't perform dml operation through simple view to base table.

2) we must include base table not null column into the view only we can perform insertion operation through simple view to base table.

Ex:-

1) create or replace view v1
as
select * from v1;
- insert into v1 (empno, ename, deptno) values (1, 'soma', 30);
- select * from emp;

2) create or replace view v2
as

select ename, sal, deptno. from emp where deptno=30;
-select * from v2;
- insert into v2 (ename, sal, deptno) values ('abc', 2000, 30);
o/p:- ora-400: can't insert null into empno column.

with check option:-

If we want to create constraint on views then we are using with check option clause in views.

syntax:

create or replace view viewname
as
select * from tablename where condition with check option;

Ex:- create or replace view v3
as
select * from emp where deptno=10 with check option;
-insert into v3(empno, ename, deptno) values(2, 'xyz', 30);
o/p:- Error: view with check option where clause violation.
- insert into v3(empno, ename, deptno) values (2, 'xyz', 10);
o/p:- 1 row created.

→ In all databases system whenever we are creating a view automatically view difinitions are permanently stored in db.

→ In oracle, if we want to view these definitions then we must use user_views data dictionary.

Ex:- create table base (sno number(10), name varchar2(10));
create or replace view v1
as
select * from base;
-desc user_views;

- select text from user-views where VIEW-NAME = 'V1';

Text

select 'SNo', "NAME" from base

COMPLEX VIEWS (OR) JOIN VIEWS :-

- complex views is a view which is created from multiple base tables.

Ex:- create or replace view vs
as
select ename, sal, dname, loc from emp, dept where emp. deptno = dept. deptno;

- select * from vs;

Ex:- update vs set ename = 'abc' where ename = 'SMITH';
1 row updated.

- update vs set dname = 'xyz' where dname = 'SALES';
Error : cannot modify a column which maps to a non key-preserved table.

- Generally in all database system we can't perform dml operations through complex views to base tables.
- when we are trying to perform dml operation through complex views to base tables some table col are affected and same other table column are not affected.
- To overcome this problem, oracle provided 'instead of trigger" Prydefault "instead of triggers" are row level triggers and also instead of triggers are created on views.

↳ in oracle if we want to view modifiable or non modifyable col then we are useing
user_updatable_column.
data dictionary.

TRIGGERS (PL/SQL)

- Trigger is also same as stored procedure but it will automatically invoked whenever dml operations performed on table or view.
- All db's having 2 types of triggers.
  1) statement level
  2) Row level.
- In statement level triggers, trigger body is executed only once per dml statements, where as in row level triggers, triggers body is executed for each row per dml statements.

syntax:- 03/03/15

```
create or replace trigger triggername.
before/after insert/update/delete on tablename
[for each row]
begin
  ≡
end
```

trigger timings — trigger events — trigger specifications — trigger body

Diff betn statement level trigger and row level trigger.

- create table test(col1, Date);

Statement level trigger :-

```
create or replace trigger tr1
    after update on emp
    begin
    insert into test values (sysDATE);
    end;
```

Testing :-

```
update emp set sal= sal +100 where deptno =10;
// 3 rows updated
-select * from test;
 col
 03-MAR-14
```

Row level trigger

```
-create or replace trigger tr2
 after update on emp
 for each row
  begin
  insert into test values (SYSDATE);
  end;
  /
```

Testing :-

```
-update emp set sal = sal +100 where deptno =10;
// 3 rows updated.
-select * from test
 o/p :- col1
        03-MAR-14
        03-MAR-14
        03-MAR-14
-drop trigger tr2
```

Maa

05/03/14

Materialized views :-

fast refresh materialized views :-

↳ These type of Materialized view are also called as increamental fast refresh

↳ These Materialized ~~used~~ view performace is very high compare to complet refresh materialize views becoz in past refresh materialize views row ~~addatio~~ ids are not changed when we are refreshing materialize views no. of times also.

syntax:-

```
create Materialized view viewname
Refresh fast as
select statement;
```

↳ Before we are creating past refresh Materialized view database Administrator must create materialized viewed log on base tables.

syntax:-

```
create Materialized view log on basetablename;
```

SQL> create materialized view log on base;

o/p created.

SQL> create materialized view mvs
refresh fast
as select * from base;

o/p> view created;

SQL> select rowid, sno, sname from mvs;

o/p:-

ROWID	SNO	Name
AAAIDAAA&AA	1	A
" &AB	2	B
&AC	3	C
&AD	4	D

> update base set name = 'xy' where sno=1;

1 row updated

> select * from base;

> exec dbms_mview.refresh ('mvs'); (Refresh the row)

> select rowid, sno, sname from MV5; (Here Rowid not changed).

On demand/on commit :-

↳ in oracle we are refresh materialize view used in two ways.

1) Automatically.
2) manually.

Manually :-

in manual method we are useing DBMS_~~MV Package~~ for Mviews Package refreshing materialized views.

↳ This method is also called as ondemand method.

↳ in oracle bydefault method is ondemand.

Automatically :-

we can also refresh materialized views without useing DbMS_Mview Package. This method is also called as on commit method.

Syntax:-

```
create materialized view viewname
Refresh complit/Refresh fast on demand/ on commit
(by default)
as
select statement;
```

say create materialized view MV6
refresh fast on commit
as
select * from base;

O/P:- materialized view created.

> select * from MV6;

Sno	Name
1	xy
2	B
3	C
4	D

sql> update base set name = 'zzz' where sno = 2;
sql> select * from base;
sql> select * from mv6;

SNO	Name
1	xy
2	B
3	c
4	D

sql> commit;
sql> select * from mv6;

o/p:-

SNO	Name
1	xy
2	zzz
3	c
4	D

force views :-

We can also create views without base table this type of views are called force views.

syntax:-

```
create or replace force view viewname
as select * from anyname;
```

sql> create or replace force view v10 as
select * from welcome;
warning: view created with compilation errors.
sql> create table welcome(sno number(10));
sql> alter view v10 compile;
sql> desc v10;

Data control language (DCL) :-

1) grant (giving permission)
2) revoke (cancle permission)

sql> conn sys as sysdba; ↵
Enter password: sys.

(or)

sql> conn system/manager

Creating a user :~

syntax : create user username identified by password;
syntax: Grant connect, Resource to username;
or
(DBA)
syntax: conn username/password;

```
sql> conn sys as sysdba;
  Enter Password : sys
sql> create user india identified by india;
sql> grant connect, resource to india;
sql> conn india/india;
sql> select * from emp;
  error : table or view does not exits.
sql> conn scott/tiger;
sql> grant all on emp to india;
sql> conn india/india;
sql> select * from scott.emp;
sql> create synonym sleep for scott.emp;
sql> select * from sleep;
```

DCL

↳ Data security point of view all database systems Provided two types of Privileges
1) System Privileges.
2) Object

1) System Privileges :~

↳ system privileges are created by database Administator only.

↳ Before we are creating Database object, Database Administator giving permission to create database object to system Privileges.

oracle heaving more then 80 system Previleges. these are

create table, create session, create any materialized view, create any view, create Procedure, create any index ....;

syntax:-

grant system privileges to usernames, ...;

sql> conn sys as sysdba;
Enter password : sys
sql> grant create table, create Procedure, create any materialized view to scott, Murali, india;

Servlet JSP
Request Req
Response Resp
out J
Printwriter JSPwriter
context Application
out out
Session session

A a = new A(db);
A Application = new A(db);
if(a == a1)
return true
else
return false

Str s =

Instance Of

if (obj instance of String)
return true
if (s instanceof String)

Req App. Page
Session

C@include
JSP:include

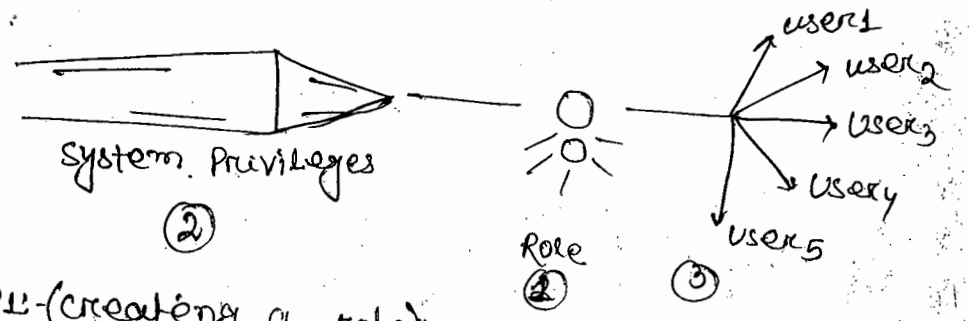
Req Disp
SendRedirect

Oracle (copy-2)

06/03/14

ROLE :-

- In all ~~databases~~ Database administrator only create role.
- Role is nothing but collection of either system or object Privileges
- Generally in multiuser environment no. of ~~us~~ users works on same Project, in this case some set of users required same set of Privileges.
- In this case only dba creates a role and assigns common set of Privileges in to role & then only that role given to the no. of users.



- **Step1:-** (creating a role):-

syntax :-

```
create role rolename;
```

- **Step2:-** assign system Privileges to a role :-

~~role:-~~ syntax :-

```
grant system Privileges to rolename;
```

- **Step3:-** assigng rolename to number of users.

syntax :-

```
grant rolename to username1, username2 ....;
```

Exm :-

```
sqly conn sys as sysdba;
    Enter password : sys
sqly create role r10;
sqly grant create Procedure, create any materialized
    view, create any index to r10;
```

grant r10 to scott, india;

↳ In oracle all system privileges related to role stored under role-sys-privs data dictionary.

ex:-

say desc role-sys-privs;

say select Role, privilege from role-sys-privs where Role='R10';

Predefined Roles :-

when we are installing oracle server automatically 3 Predefined Roles are created, these are :-

1) Connect → end user
2) Resource → developers
3) dba → adminstrator.

*↳ say desc role-sys-privs;

say select ROLE, PRIVILEGE from role-sys-privs where ROLE IN ('CONNECT', 'RESOURCE');

say select ROLE, PRIVILEGE from role-sys-privs where ROLE IN ('DBA');

Object Privileges :-

↳ object privileges are used to perform some operations on the object these privileges are given by either developers or database administrator.

↳ Oracle heaving following object privileges, these are:-

insert, update, delete, select.

called as 'all'.

↳ execute, read, write.

Syntax:-

Grant objectPrivileges on objectname to usernames / Rolename / Public;

SQL> show user;
User is "SCOTT"
SQL> grant all on emp to India;
SQL> grant all on emp to r40;
SQL> grant all on emp to Public;

↳ All object Previliges related to user stored under user_tab_Privs data dictionary.

SQL> desc user_tab_Privs;

with grant option:-

↳ who receives with grant option class those ~~user~~ user give Privileges to another user?

syntax:-

grant object Privileges on objectname to usernames/Public with grant option;

SQL> grant all on emp to india with grant option;
SQL> grant all on emp to r40 with grant option;
~~SQL~~ error: cannot GRANT to a role WITH GRANT OPTION.

NB:- In all database systems Roles does not work with grant option class.

revoke:- This command is used to cancle either system or objectprivileges.

syntax:-

Revoke systemprivileges from username1, username2

syntax:-

Revoke objectprivileges on objectname from usernames/Rolename;

↳ Generally if we want to restricts ~~colu~~ tablecols from one user into another user then we

must create a view with a required columns & that view given to the no. of users.

```
sql> conn scott/tiger;
sql> create or replace view v1 as
     select empno, ename, sal, deptno from emp;
sql> grant all on v1 to soma;
sql> conn soma/soma;
sql> select * from scott.v1;
```

MERGE :-

- Oracle 9i introduced MERGE statement.
- Basically this is an DML command.
- Generally MERGE statement used of data warehousing
- using MERGE statement we are transformingg data from source table into targate table.

syntax:-

```
MERGE into targate tablename
using source tablename
on (join condition)
when matched then
update set targate table colname1 := Source table col1, ...
when not matched then
insert (targattable colname) values (Source table col names);
```

<oracle>

07/03/14

- if Merge statemente are useing insert, update stateme conditionaly.
- Through Merge statement we can't modify on clause columns.

eg:-

```
SQL> Select * from dept; (target table)
SQL> create table depts as select * from dept;
SQL> insert into depts values (1,'a','b');
SQL> Select * from depts; (source table)
SQL> merge into dept d using depts s on (d.dept=s.deptno)
     when matched then
     update set d.name=s.dname, d.loc=s.loc
     when not matched then
     insert (d.deptno, d.name, d.loc) values (s.deptno,
                                              s.dname, s.loc);
SQL> Select * from dept;
```

SEQUENCE

- Sequence is a database object which is used to generate sequence no automatically.
- Generally sequences are used to generates Primary key values automatically.
- Generally sequences are created by database administrator.
- Sequence is a independent database object, once a sequence has been created no. of users simeltanious acces that sequence.
- in all database system sequence are used in automatcament concept.

syntax:- (auto increament)

```
create sequence sequencename
start with n
increament by n
minvalue n
maxvalue n
cycle/nocycle
cache/no cache;
```

eg:- create sequence s_1
start with 5
increament by 2
maxvalue 100;

5
7
9
11
⋮
100

↳ If we want to generate sequence no. then we must use following pseudo col.

1) currval
2) nextval

syntax:-

```
sequencename.currval
```

syntax:-

```
sequencename.nextval
```

↳ Sequences are used in insert, update, delete, select statem.

↳ In Oracle if we want to generate Sequence values using select statement then we must use dual table.

syntax:-

```
Select sequencename.currval from dual;
```

syntax:-

```
Select sequencename.nextval from dual;
```

↳ In all database system If we want to generate 1st sequence no. then we must use nextval pseudo column. Becoz currval pseudo column always returns

current value of the sequence, if sequence session heaving a value.

eg:- select s1.currval from dual;
error: sequence s1.CURRVAL is NOT yet defined in this session.

sql> select s1.nextval from dual;

NEXTVAL
5

sql> select s1.nextval from dual;

NEXTVAL
7

↳ If we want to change sequence Property value then we are useing alter sequence.

syntax:-

```
Alter sequence sequencename
Property name newvalue;
```

sql> alter sequence s1 increment by -1;

NB:- In all database system we can not change start with value useing alter sequence.

eg:- alter sequence s1 start with 4;
error: cannot alter starting sequence number.

↳ In sequences start with can not be less then minvalue.

eg:- create sequence s1
start with 3
increament by 1.
minvalue 5
maxvalue 10;

o/p:- error: START WITH cannot be less then MINVALUE.

cycle/nocycle :-

```
SQL> create sequence s1
     start with 5
     increament by 1
     minvalue 3
     Maxvalue 10
     cycle
     // nocache;
SQL> select s1.nextval from dual;
op:-5
   6
   ...
   10
   3 -> cycle
   4
   ...
   10
   ...
   3 -> cycle
```

```
SQL> create sequence s1
     start with 5
     increament by 1
     minvalue 3
     maxvalue 10
     nocycle
     nocache;
SQL> select s1.nextval from dual;
op:-5
   6
   ...
   10
error:- sequence s1.NEXTVAL exceeds MAXVALUE and cannot be instantiated.
SQL> create table test(sno number(10) Primary key,
                       name varchar2(10));
SQL> create sequence s1 start with 1;
SQL> insert into test values (s1.nextval, '&name');
     Enter value for name : abc
     SQL/
```

say select * from test;

SNO	NAME
1	abc
2	xyz
3	pqr

CACHE:-

Methods of servlet context :~

↳ Java. lang. string
getInitParameter (Java. lang. string name).

- Return a string containing the value of the named context-wide initialization parameter, or null if the parameter does not exist.

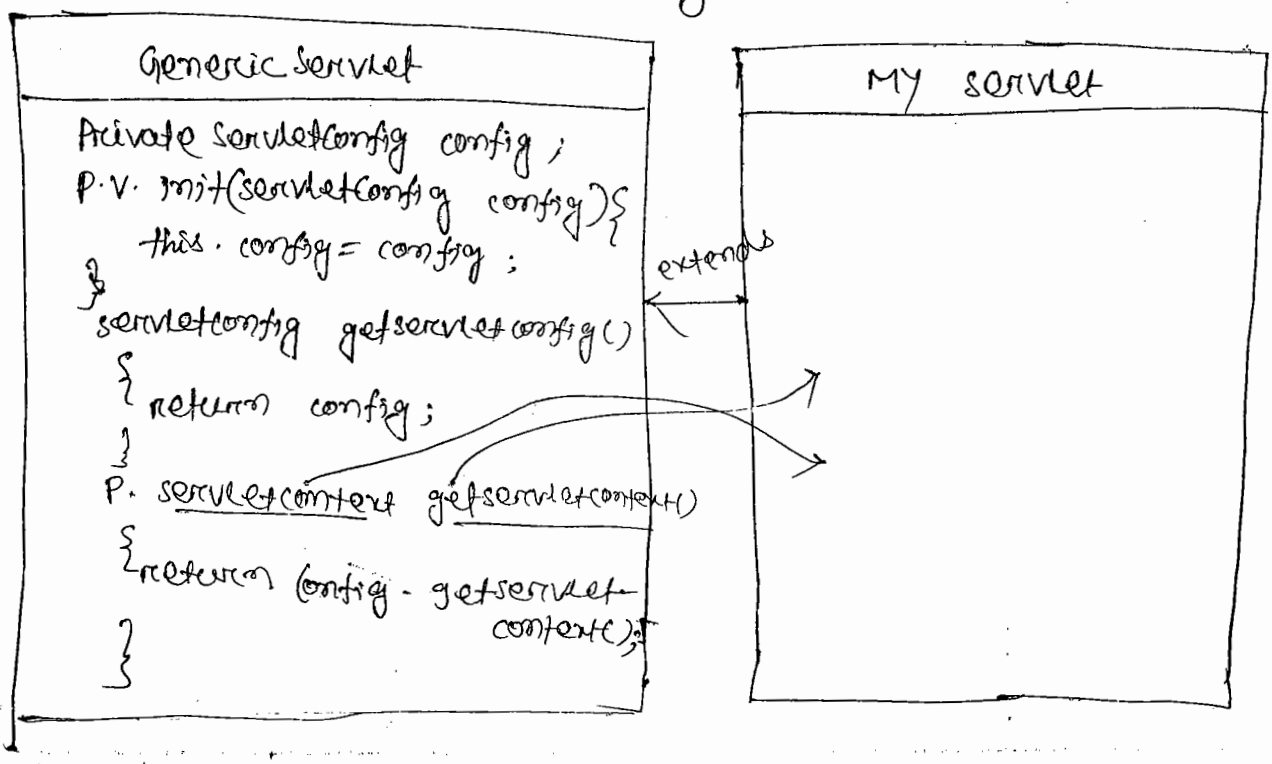
↳ Java. util. enumeration <Java. lang. string>
getInitParameterNames ().

- Returns the name of the context's initilization Parameters as an enumeration of string objects, or an empty enumeration if the context has no initilization Parameters.

How to get servletcontext object ?

1) getServletContext () method servlet config interface.
- servlet context ctx = getservlet config (). getservlet context()

2) getServletcontext () method of GenericServlet.
- servletContext ctx = getservletcontext();



08/03/14

Mar 7

Cache :-

- cache is a memory Area which is used to stored setup sequence numbers.
- cache option used by Database Administrator to retrive sequence values very fastly.
- Generally in all database systems sequence values are generated from disk, if we want to access these sequence values very fastly then database Administrator putting set of sequence no.s into cache memory Area. Then only no. of Application Prog^n access sequence no. from memory Area these Process automatically improves Performance of the applications.
- in Oracle by default cache value is 20 and also min value is 2.
- When ever system crashes automatically cache values are lossed.

eg:- sqly create sequence s1
start with 1.
increament by 1
Maxvalue 100
cache 1;

o/p:- Error: the number of value to cache must be greater then 1.

- All sequences info stored under user_sequences data dictonary.

sqly Desc user_sequences;

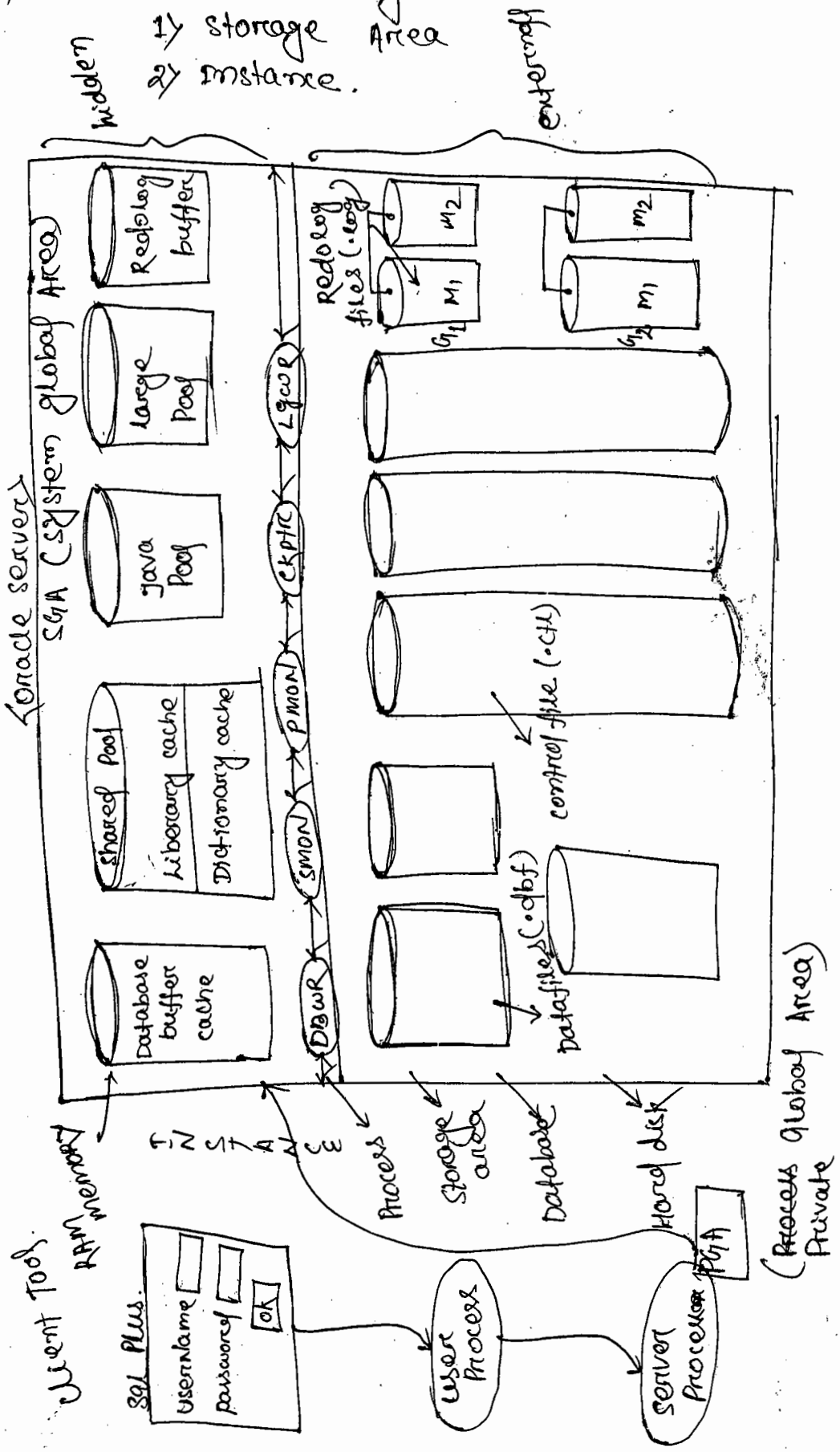
- we can also drop sequence useing drop_sequence sequencename.

sqly drop sequence s1.

ORACLE SERVER ARCHITECTURE

↳ Oracle server having two Parts :-

1) Storage Area
2) Instance.



1) Storage Area :-

whenever we are installing Oracle server automatically 3 files are created in H.D. These are

i) Data files (.dbf)
ii) Control files (.ctl)
iii) Redo log files (.log).

↳ These 3 files are also called as physical database or storage area.

i) Data-files

↳ In oracle whenever we are creating database object those objects are permanently storing datafiles.

↳ All datafiles information stored under dba-data-files data dictionary.

eg:-
sql> Conn sys as sysdba;
Enter Password: sy
connected.
sql> desc dba-data-files;
sql> select file-name fre dba-data-files;

ii) control files :- 10/03/14

↳ control files stored database information.

↳ This files extent is .ctl.

↳ Control files are used by dba in recovery Process.

↳ All control files information stored under V$controlfile data dictionary.

eg:- sql> conn sys as sysdba;
Enter password: sys
sql> desc V$controlfile;
sql> select name from V$controlfile;

iii) Redolog files :-

it stores committed information from Redolog buffer.

↳ Redolog files also used by dba in recovery Process.

eg:- desc V$logfile;

SQL> Select Member from V$logfile;

2) <u>Instance</u> :-

↳ whenever we are connecting to the database server, using a tool automatically an instance is created in RAM Memory area.

↳ This instance is consist of 2 Parts :-

i) SGA

ii) Process.

i) <u>SGA</u> :- (system global Area).

↳ SGA is also called as system/ shared global area.

↳ This global area consist of set of buffer. These are :-

- database buffer cache.
- Shared pool.
- Java pool.
- large Pool.
- Redolog buffer.

↳ <u>Process</u>

when ever we are ~~submitting~~ submitting SQL, PL/SQL code that code automatically stored in liberary cache within shared pool.

↳ This liberary cache reduces parsing.

↳ Liberary cache also stored cache values defined in sequece objects.

↳ Shared pool also contains dictonary cache which execute DCL related data dictonary information.

- whenever we are requisting a table using a tool that oracle server Process checks requested table available in database buffer cache.
- if it is not available in database buffer cache then DBWR Process checks requested table available in datafiles. if it is available then copy of the table created in database buffer cache.

Java pool execute java related objects and also large pool used by database administrator in recovery process.

- Redolog buffer stores new values for the transaction..
- Server Process also contains PGA (Private global area) which identify a clients uniquely.
- PGA also stores PL/SQL collections.
- Oracle server mainly consist of 3 structure.
  1) Physical structure
  2) Logical structure.
  3) Logical storage structure.

1) Physical structure :-

- Physical structure contains 3 files visible in O.S.
- Physical ~~files~~ structure are handle by database Admin. only
- control files, log files.

2) Logical structure :-

Logical structure not visible in O.S.

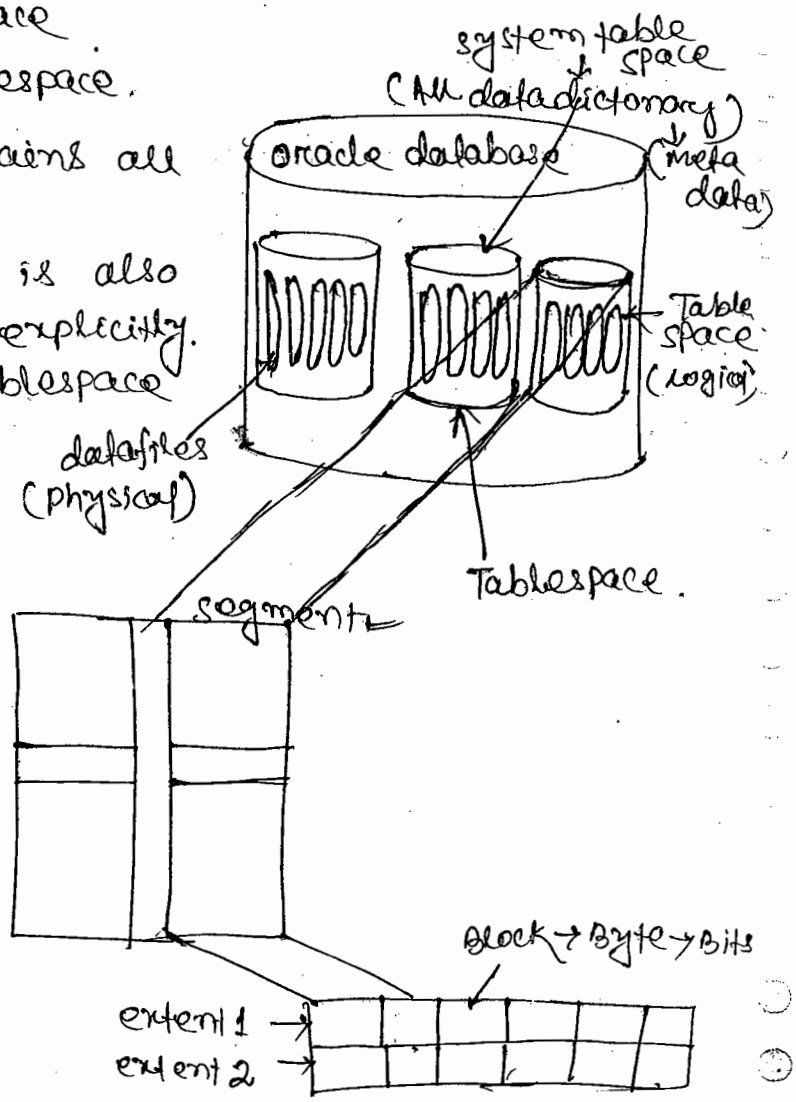
- This structure contains database objects like. table, views, Procedures, functions, indexes.
- This structure handles by either developer or dba.

iii) Logical storage structure

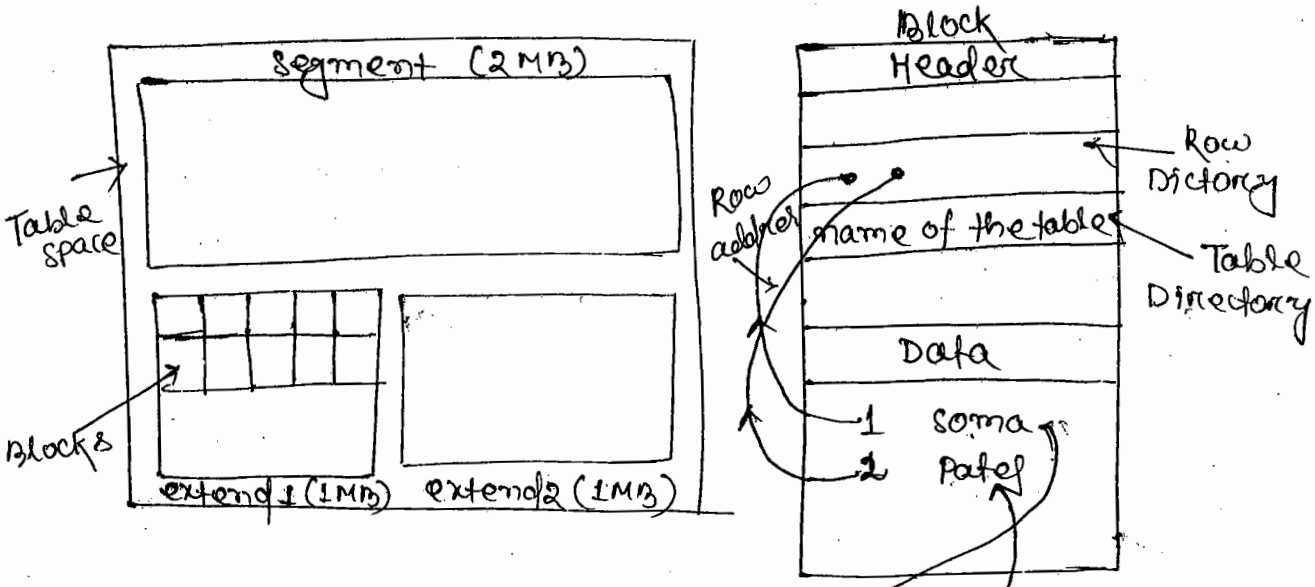
- Logical storage structure handle by dba only.
- This structure contains :-
  - Tablespace.
  - segment.
  - Extent.
  - Block.

• Tablespace :-

- Tablespace is nothing but collection of datafiles. one data files belongs to one tablespace only.
- whenever we are installing oracle server automatically 6 tablespaces are created.
- If we want to run oracle minimum 2 tablespaces are required, these are :-
  - system tablespace
  - SYSAUX tablespace.
- system tablespace contains all data dictonaries.
- database Administator is also create tablespaces explicitly. by useing create tablespace Privileges.



sql> create table T_1 (sno number(10), name varchar$_2$(10));

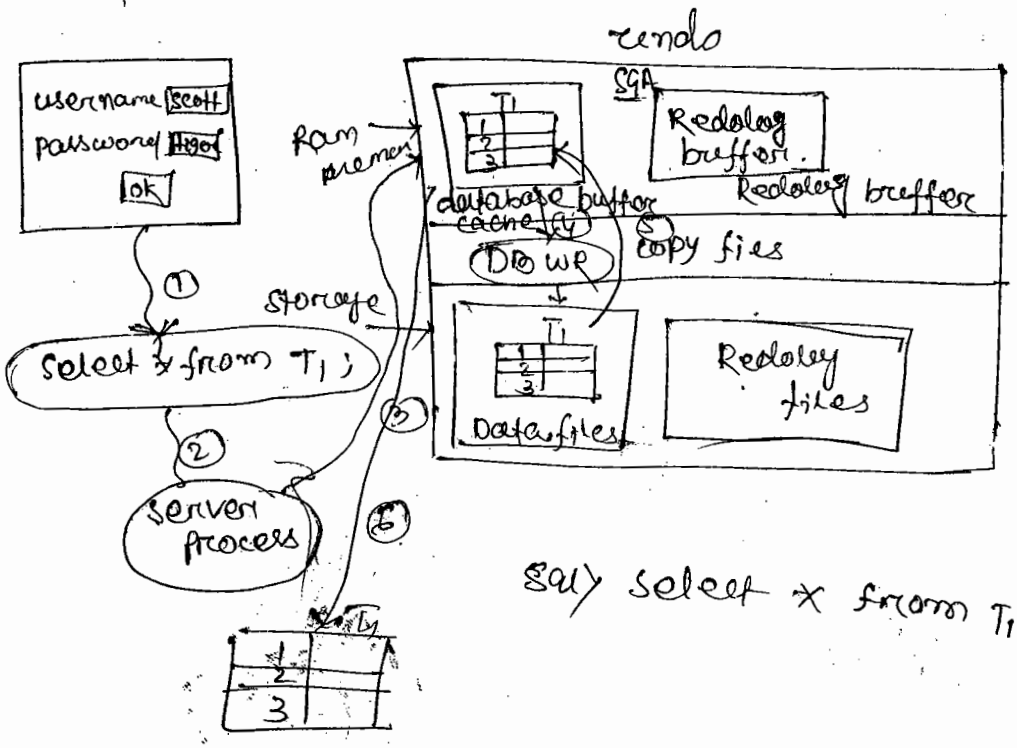


sql> insert into T_1 values (1, 'soma');
sql> insert into T_1 values (2, 'Patel');

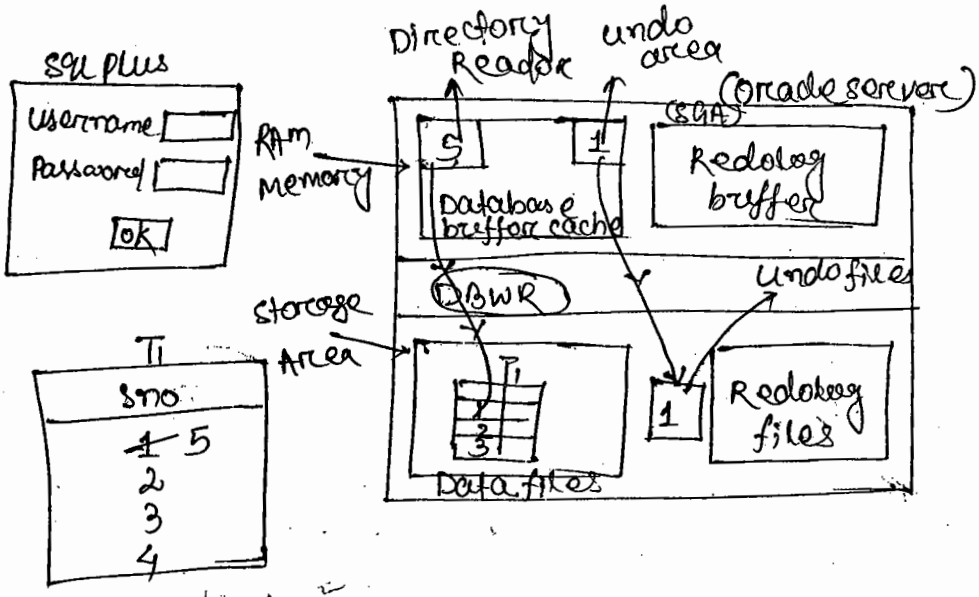
<u>Segment</u> :-

↳ Segment is a collection of extends that forms a database object like, tables, indexes, views;

<u>NB:</u> in all database systems list storage element is Block.



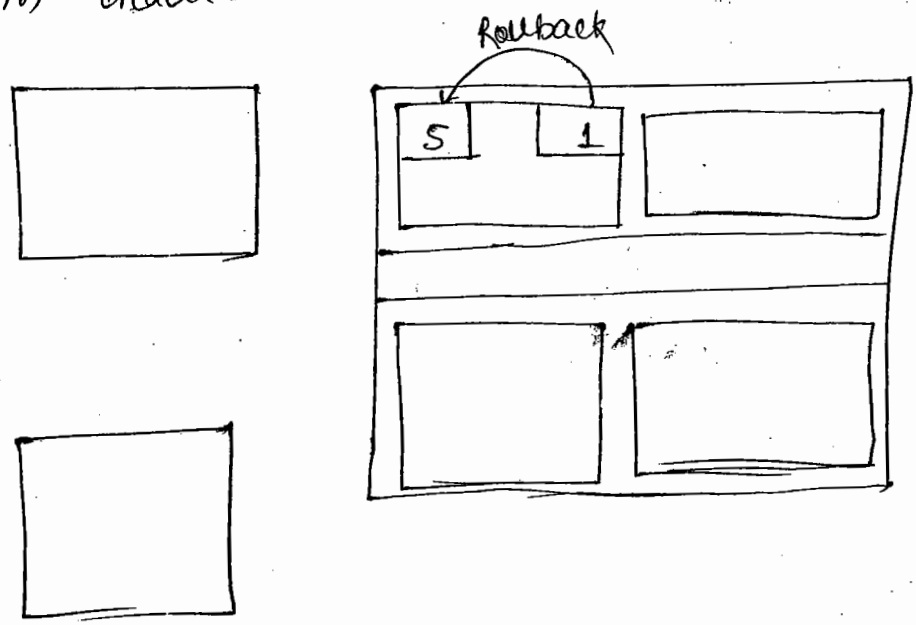
sql> select * from T_1;



sql> update T1 set sno=5 where sno=1;
sql> commit;

NB:- If oracle database whenever we are performing transaction new value for the transaction automatically stored in DB ~~not only~~ we are using commit not only this one, old values of the transaction also automatically stored in undo files with in hard disk.

→ These undo files are used by flashback queries in oracle.

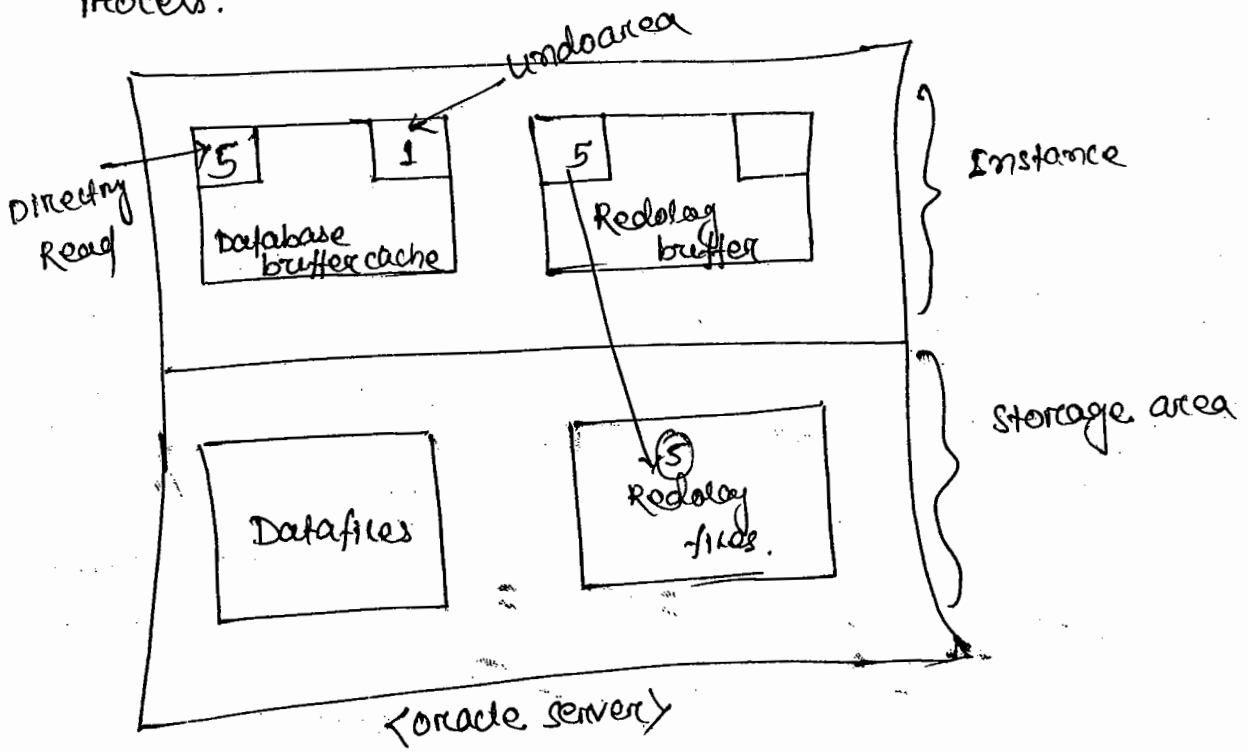


Mga

Dt 11/03/14

? Redolog buffer :~

↳ when we are performing transaction, new values for the transaction automatically stored in Redogy buffer. whenever user going commit / 1/3 fill on redolog buffer then automatically redolog buffer data transfer into redolog files. These Redolog files are used by DBA in recovery process.



NORMALIZATION

- Normalization is a scientific Process which is used to decompose a table into no. of tables.
- This Process automatically Reduces redundancy.
- Normalization Process automatically avoids insertion, updation, deletion Problems.
- In design phase of the SDLC, database designers design logical model of the database. in this logical model only database designers uses normalization Process through normal forms.
- In 1970. E.F codd written a paper "relational Model of data for large shared data banks".
- In this paper only. EFcodd introduced 1st 3 normal forms.

Normal forms :-

1) first normal form (1NF)
2) Second " " (2NF)
3) Third " " (3NF)
4) BCNF (Boyd codd NF)
5) Fourth normal form (4NF)
6) Fifth " " (5NF)

First normal form (1NF) :-

In a table is in 1st normal form, then every cell having a single value and also identifying record uniquely using a key.

Item table

Item Name	color	Price	tax
Marker	black, Red	20	0.2
pen	blue, green	30	0.3

(NOT IN 1NF)

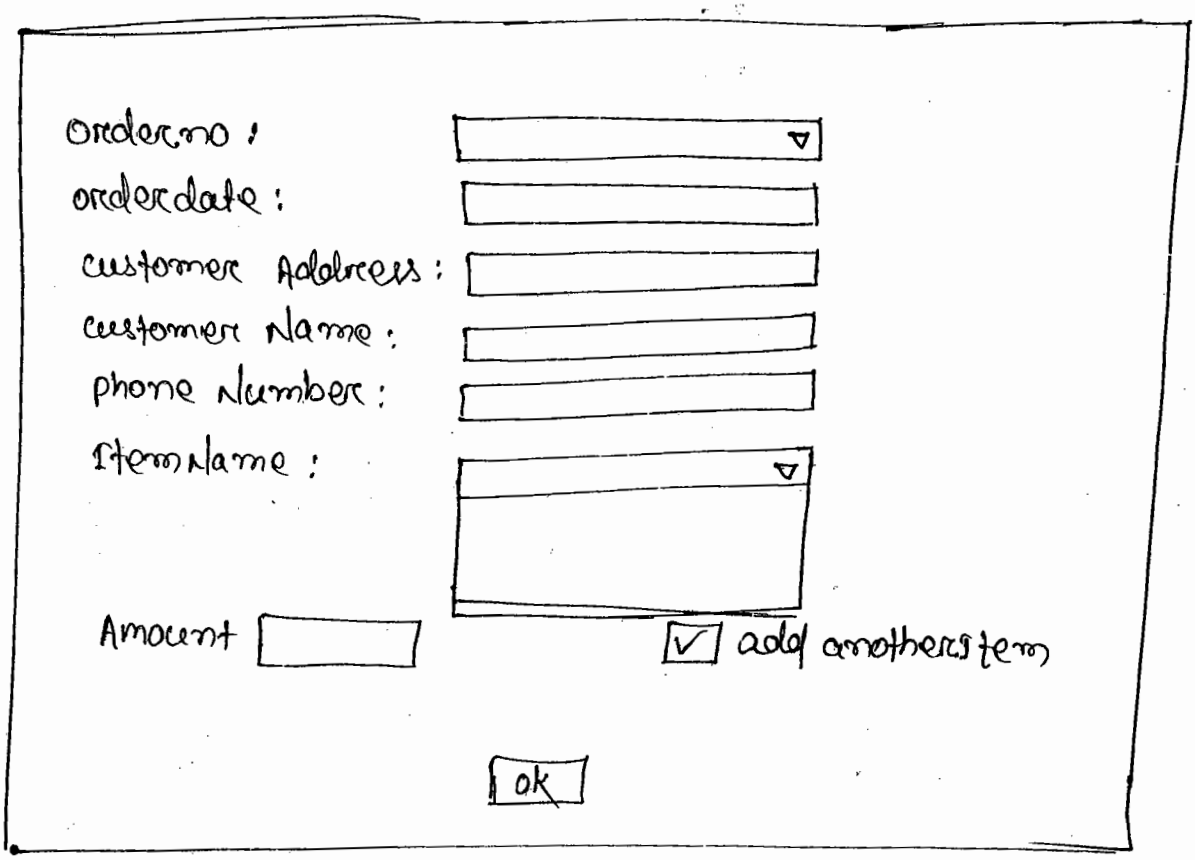
candidate key

1NF →

Item table

Item Name	color,	Price	tax
Marker	black	20	0.2
Marker	Red	20	0.2
pen	blue	30	0.3
pen	green	30	0.3

(IN 1NF)



* In this example Itemname amount are repeating group. so draw it in another table and for more atomicity add order no to that table.

Primary key

ordermo	order date	customer Address	customer Name	ph No

<order - Master table>

order no	Itemname	Amount
1	Bluestar A/c	40.000
1	LED TV	30.000
1	Refrigerator	45.000

→ order details of table or child table.

(1NF)

4) Identify Repeating Groups and put it into separate table in more atomic form.

in all database system by default 1st normal form Process table is a child table. in this table one column having duplicate data. in this table one column not helping to uniquely identifying a record.

Second Normal form

A table is in 1st normal form and also all non-key attributes fully functionally dependent on total candidate key.

Generally, 1st NF deals with Atomicity where as 2NF deals with Relationship betn key, non-key attributes.

when a table contains particular non-key attributes then the table not in 2nd normal form.

Process :-

1) Identify the partial non-key attributes which depends on particular key attributes and those all attributes put into separate table. This table is called 2NF table. By default, 2NF table is a Mastertable. in this table only all nonkey

attributes fully functionally dependent on total candidate key.

key attribute

candidate key

Item Table

Item Name	color	Price	Tax
Marker	black	30	0.3
Marker	Red	30	0.3
pen	blue	20	0.2
pen	green	20	0.2

P

P

non-key attribute.

Item Name	color

Foreign key

Item Name	Price	Tax

Primary Key

Master Table

12/03/14

<student sports activity>

Sno	Sname	Activity1	cost1	Activity 2	cost 2
101	abc	cricket	$10	Golf	$20
102	xyz	Hokey	$30	cricket	$10
103	zzz	swimming	$40	Golf	$20
104	aaa	cricket	$10		

Sno	Sname
101	abc
102	xyz
103	zzz
104	aaa

Sno	activity 1	cost1	activity 2	cost2
101	cricket	$10	Golf	$20
102	Hokey	$30	cricket	$10
103	Swim	$40	golf	$20
104	cricket	$10		

(Not in 2nd N.F).

key attribute

Candidate key

Partial P

Sno	Sname
101	abc
102	xyz
103	zzz
104	aaa

Primary key

Sno	activity	cost
101	cricket	$10
101	Golf	$20
102	Hokey	$30
102	cricket	$10
103	swim	$40
103	Golf	$20

Foreign Key

Activity	cost
cricket	$10
Golf	$20
Hokey	$30
Swim	$40

Primary Key

Master Table

Sno	Sname
101	abc
102	xyz
103	zzz
104	aaa

2nd N.F Table

Sno	ACTIVITY	cost
101	cricket	$10
101	Golf	$20
102	Hokey	$30
102	cricket	$10
103	swim	$40
103	Golf	$20

Non-key attribu

key attribute, candidate key

< Project Table >

P, P, Non-key Attribute, F

Ecode	Projectcode	dept	depthead	Hours
E01	P_1	system	D_1	5
E02	P_1	sales	D_2	7
E103	P_2	research	D_3	9
E101	P_2	system	D_1	12
E104	P_3	s/w	D_4	8
E102	P_3	sales	D_2	15

not in 2nd NF

Phase 1 :- (uniqueness check called as FD) (functional dependency)

① Ecode $\xrightarrow{FD}$ Hours (X)

for every value for Ecode more then one value in hours.

E101 → 5, 12

② Projcode $\xrightarrow{FD}$ Hours (X)

P_1 → 5, 7

③ Ecode + Projcode $\xrightarrow{FD}$ Hour (✓)

Phase 2 :-

① Ecode $\xrightarrow{FD}$ Dept (✓)

E101 → System

② Projcode $\xrightarrow{FD}$ Dept (X)

P_1 → System, Sales

Phase 3 :-

① Ecode $\xrightarrow{FD}$ Depthead (✓)

② Projcode $\xrightarrow{FD}$ Depthead (X)

Primary Key <Master Table>

ecode	dept	deptheaf
E101	system	D_1
E102	sales	D_2
E103	research	D_3
E104	s/w	D_4

Foreign Key <child table>

ecode	Projcode	hours
E101	P_1	5
E101	P_2	12
E102	P_1	7
E102	P_3	15
E103	P_2	9
E104	P_3	8

Mar 13/03/14

↳ Before Normalization above resource table having insertion, updation, deletion Problems.

↳ Insertion Problem :—

↳ In the above resource table when we are inserting a Particular dept employee we must assign Project code. This is called Insertion Problem.

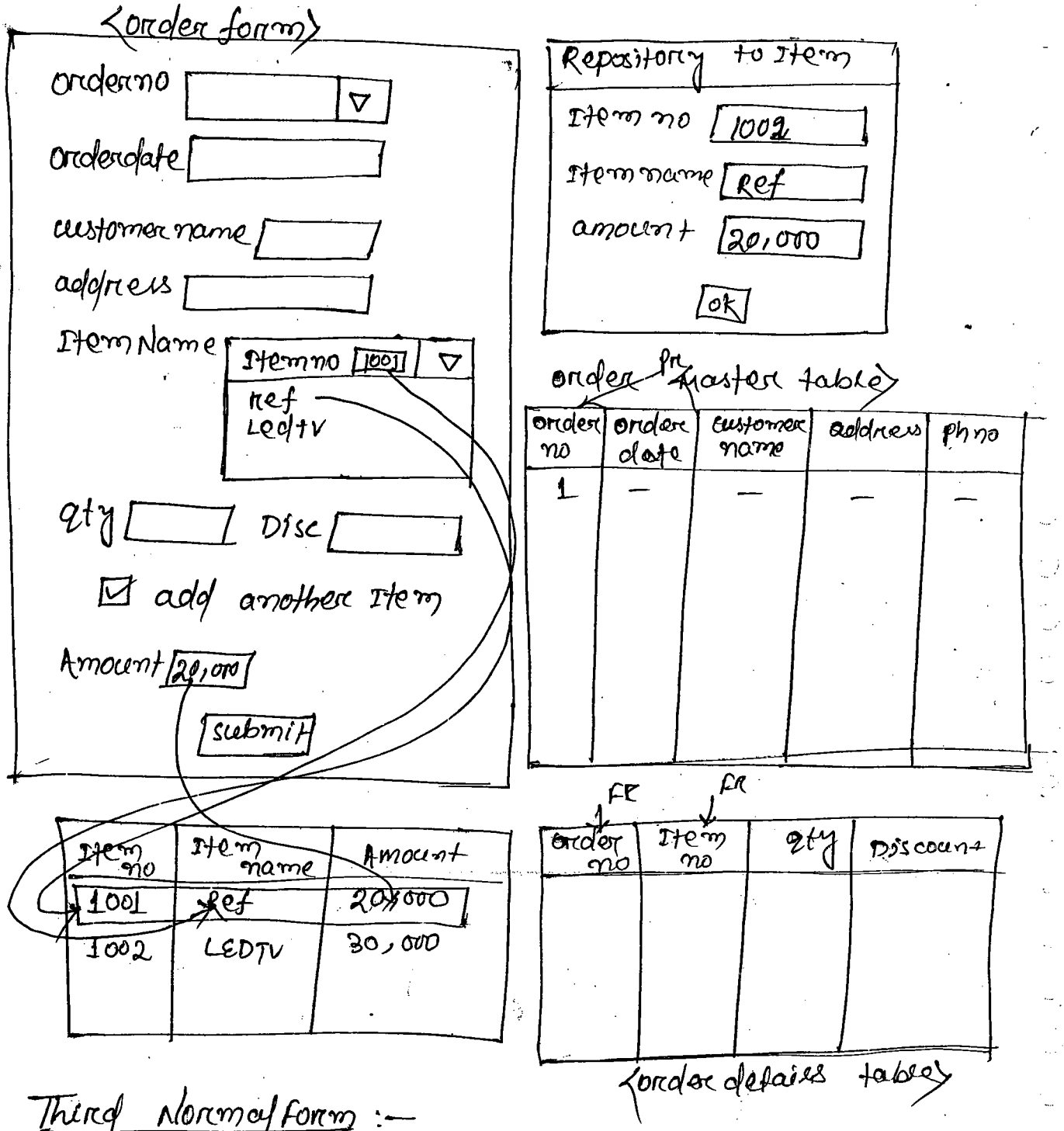
↳ Updation Problem :—

↳ In the above resource table ecode, dept, deptheaf values are repeated. whenever a particular employee transfered one dept to another dept then we must Modified on thes 3 attribute values. correctly other-wise inconsistency Problem occured. This is called updation Problem.

↳ Deletion Problem :—

↳ In the above resource table when we are deleting a employee record then automatically dept details also deleted. this is called as deletion Problem.

↳ when ever we are useing normalization Process automatically insertion, updation, deletion Problems are avoided.



Third Normal form :-

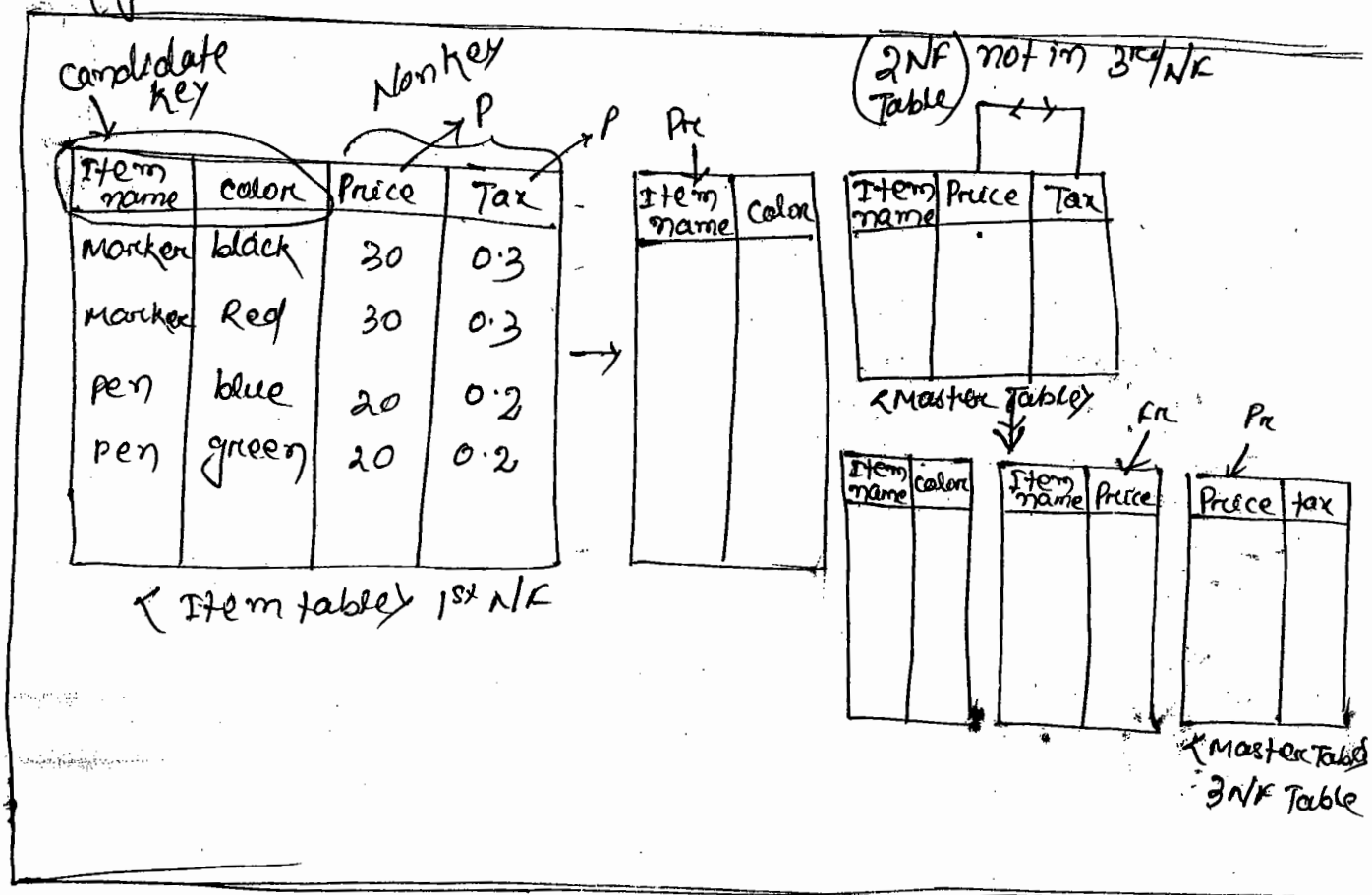
- A table is in 2nd NF and also all nonkey attributes only dependent on only Primary key.
- whenever Non-key attributes which are depends on another Non-key attributes then that table not in 3rd normal form.

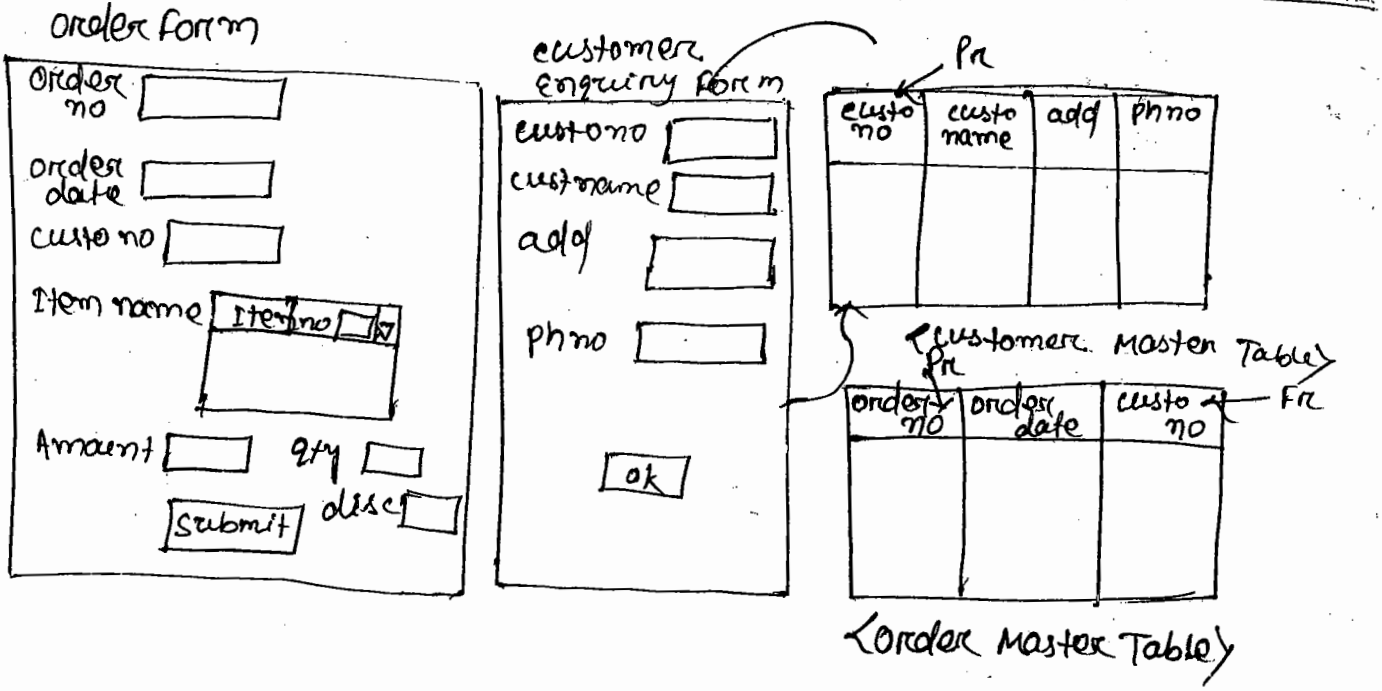
Process :-

- Identifying non-key attributes which depends on another non-key attributes those all attribute

put into seperate table. These table is called 3rd NF Table.

↳ By default this is an Mastertable. in this table only all Non-key attributes are only depends on Primary key.





2NF

Item no (Pr)	Item name	Amount

(Item Master)

Order no (FR)	Item no (FR)	Qty	Disc

(Order details Table)

Mar 8

14/03/14

Candidate key

Non key value

Sname	Customer Id	Grade	Grade value
abc	CS111	A	4.00
xyz	CS111	B	3.00
zzz	CS222	C	2.00
aaa	CS222	A	4.00

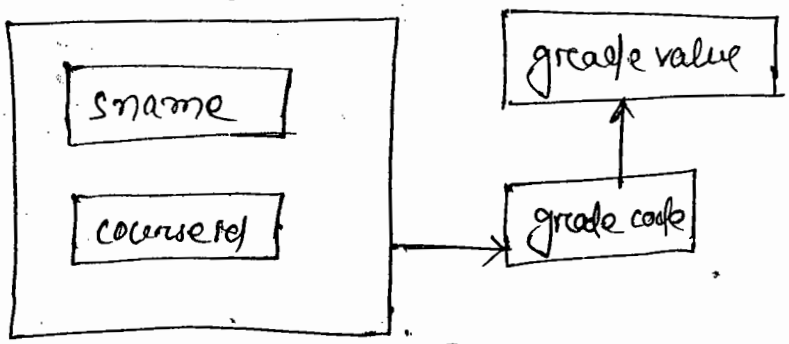
(Not in 3rd NF)

Sname	cust id	Grade (FR)
abc	CS111	A
xyz	CS111	B
zzz	CS222	C
aaa	CS222	A

Grade (Pr)	grade value
A	4.00
B	3.00
C	2.00

(Master table) (3rd NF)

Logical Diagram :—



Functional Dependency :—

If given two tuples in a relation 'r' then 'x' (an attribute set) agrees then 'y' (another attribute set) can not disagree then $x \rightarrow y$ is called functional dependency.

$$x \rightarrow y$$

here y is functionally dependent on x.

(OR)

x functionally determines y.

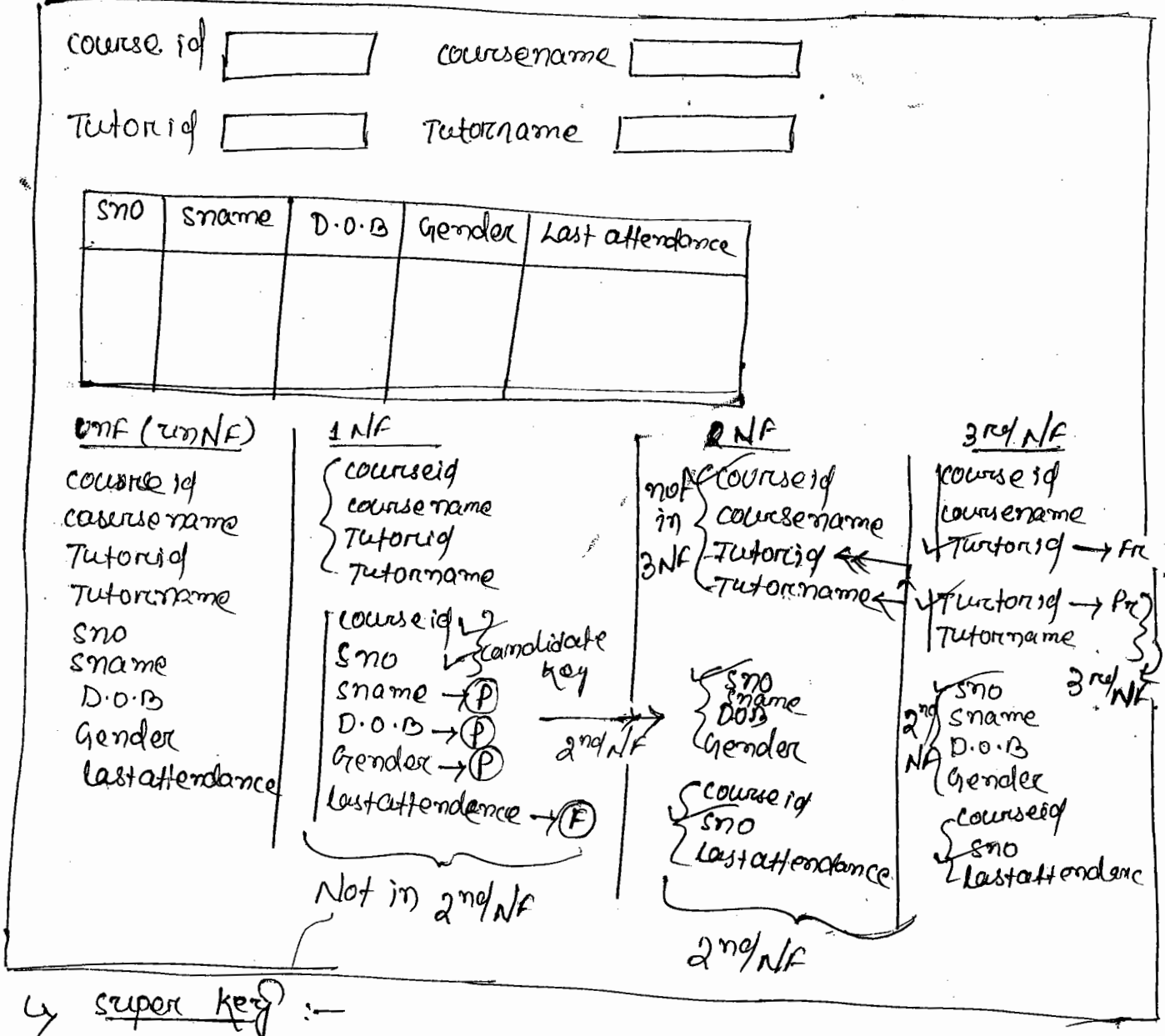
1NF → remove repeating groups.

2nd NF → Remove Partial attributes.

3rd NF → Remove attributes those are not dependent on key

(OR)

Remove attributes which are dependent on another non-key attributes.



↳ <u>super key</u> :-

A column or operational column which uniquely identifies a record in a table.

↳ <u>candidate key</u> :-

Minimal superkey which uniquely identify record in a table is called candidate key.

↳ i.e a superkey which is subset of another key is not a candidate key.

↳ **Primary key:-**

Any one of the candidate which is uniquely identify record in a table is called Primary key.

↳ **Alternative key:-**

Otherthen Primary key candidate keys are called as alternative key.

Empno	Ename	Skillid	Skill	voterid
1	soma	1	oracle	v_1
1	soma	2	sybase	v_1
1	soma	3	Ingress	v_1
2	abc	4	DB_2	v_2
2	abc	1	oracle	v_2
3	xyz	5	Informix	v_3
4	zzz			v_4
5	aaa	6	MSSQL server	v_5
5	aaa	4	DB_2	v_5

superkey

1) Empno + skill → candidate key
2) Empno + Ename + skill (x)
3) Empno + skill + voterid (x)
4) Ename + skill → candidate key
5) Ename + skill + voterid (x)
6) voterid + skill → candidate key
7) Empno + Ename + voterid + skill (x)

candidate key

1) Empno + skill → Primary key
2) Ename + skill
3) voterid + skill } → Alternative key.

Mar 7 15/03/14

CLUSTER

- cluster is a database object which is used to improve Performace of the join query.
- cluster contains group of tables together and shares same datablock.
- cluster tables must have a common column name this common col is also called as cluster key.
- whenever we are submitting inner join & outer Join then database servers checks request tables are available in cluster or not.
- if from clus tables available in cluster then database servers very fastly retrive data from cluster table.

Generally database adminstrator only creates cluster at the time of table creation.

- Based on common col name

Creating a cluster:-

step:-1

cluster creating. or creating a cluster.

- DBA create a cluster object based on common col name.

syntax:-

```
create table cluster clustername(common colname
    datatype (size));
```

step:-2 create an index on cluster:-

syntax:-

```
create index indexname on cluster clustername;
```

step:3:- create actual tables

syntax:-

```
create table tablename (col1 datatype(size),
    col2. datatype(size) ....) cluster clustername
    (columnname);
```

eg:-

say create cluster emp-dept (deptno number(10));

say create index z1 on cluster emp-dept;

say create table e1(deptno number(10), empno number(10),
    ename varchar2(10), sal number(10))
    cluster emp-dept (deptno);

say create table d1(deptno number(10), dname varchar2(10),
    loc varchar2(10)) cluster emp-dept (deptno);

↳ we can not drop cluster, if cluster having tables. to overcom this Problem oracle 8.0 introduced to drop cluster along with tables useing including tables clause.

syntax:-

```
drop cluster clustername including tables;
```

eg:-

say drop cluster emp-dept;
    error: cluster not empty.

say drop cluster emp-dept including tables;
    cluster dropped.

say desc user-clusters;

↳ all clusters information stored under user-clusters data dictionary.

BCNF :-

A Table is a BCNF, then every determinent is a candidate key.

- whenever tables haveing multiple candidate keys and also these candidate keys are overlaped and also one candidate key non-key attribute which depends on ~~other~~ another candidate nonkey then only we use BCNF Process
- if a table contains single candidate key then BCNF is same as 3NF.
- when tables contain multiple candidate key then deletion Problem occured, to overcome this Problem we are useing BCNF Process.

Process :-

identifing one candidate key Non-key attributes which depends on another candidate key non-key attributes put in to seperate tables this tables is called BCNF.

- in this table only every determinement behaves like a candidate key. and also this table doesn't contain non-key attributes.

Ecode	name	Projcode	hours
E_1	abc	P_1	8
E_2	xyz	P_2	5
E_3	zzz	P_4	6
E_4	soma	P_1	7
E_4	soma	P_2	9
E_5	aaa	P_3	10

(Project-table)

Non Key attribute

FD

1. Ecode + Projectcode → hours
2. name + Projectcode → hours

1. Ecode + Projecode → candidate key
2. name + Projcode → candidate key

① ecode → name
↓
determinant
↓
candidate key

Ecode	name
E_1	abc
E_2	xyz
E_3	zxz
E_4	soma
E_5	aaa

(BCNF Table)

② name → ecode (FD)
↓
determinant
↓
candidate key.

Profid	Dept	H.O.D	hours
P_1	chemestry	H_1	5
P_1	comp	H_2	7
P_2	physic	H_3	6
P_3	zoology	H_4	10
P_3	Botany	H_5	12

① Profid + Dept → Hours (FD)

② Profid + H.O.D → hours (FD)

① Profid + dept → candidate key ①

② Profid + H.O.D → candidate key ②

Dept	H.O.D
chem	H_1
comp	H_2
physics	H_3
zoology	H_4
Botany	H_5

(BCNF Table)

② H.O.D → Dept (FD)
↓
determinent
↓
candidate key

① Dept → H.O.D (FD)
↓
determinent
↓
candidate key

Profid	Dept	hours
P_1	chem	5
P_1	comp	7
P_2	physic	6
P_3	zoology	10
P_3	Botany	12

Forth Normal Table (4NF) :—

Mag

17/03/14

Fourth Normal Form :—

- A Table is a fourth normal form. Then There is no multivalue dependency. i.e
- If a table is a 4th NF then that table does't contain more then one independent multivalued attribute
- Multivalued dependency accured, when a table contain more then 2 bits and also identifies record using combination all bits & also one attribut set of values which depends on another ~~valu~~ attribute set of values and also same attribute set of values which depends on another attribute set of values & also some attributes are not logically related then only we are using 4th NF.

Process :—

- identifying idenpendent attribute which dependents on dependent attribute put it in to seperate table.

Movie	Star	Procedure
M_1	S_1	P_1
M_1	S_2	P_1
M_1	S_3	P_1
M_1	S_4	P_1
M_1	S_5	P_2
M_1	S_6	P_2
M_2	S_7	P_5
M_2	S_1	P_5
M_2	S_2	P_5
M_2	S_3	P_5

Movie	Star

Movie	Procedure

NB:- when a table contains only one independent multivalued attribute then that table automatically reduces repreating information.

Project Table ← Candidate key

empid	Project id	skills
1111	Railway Res	JSP
1111	Railway Res	Oracle
1111	Railway Res	.Net
1111	Library Pr	.php
1111	Library Pr	SQL Server
1111		HA.

→

empid	Project id
1111	Railway
1111	Library

empid	skills
1111	JSP
1111	Oracle
1111	.net
1111	Php
1111	SQLserver

↳ in the above resource table an employee having new skills then we must set Project id to null & also when we are try insert new Project then we must set Employee skills to null. To overcome this Problem DB designers uses 4th NF Process.

Fifth Normal Form :—

↳ A Table is in 5th NF then that table does't contains cyclic dependency. cyclic dependency occource when a table contains more than two fiels & also indentifing a record through combination of all fields and also all fields are related to one another then only database designers uses 5NF.

↳ 5th NF is also called as Projection joined NF.

↳ In 4th NF some attributes are not logically related.

but in 5th NF all attributes are logically related.

Buyer	Product	company
B_1	Shirt	Levies
B_1	Jeans	Arrow
B_1	shirt	Pepe's

Buyer	Product

Product	company

Buyer	comp

Indexes:-

- it is a Database object which is used to improves performace of the application, becoz indexes ~~are~~ on column retrive data very fastly form the DB.
- Indexes are created tables columns.
- Generally we are creating indexes two ways
  1) Automatically.
  2) Manually.
- **Automatically:-**
  whenever we are creatating P.K or unique key then Oracle Server automatically creates a B-tree index of those columns.
- **Manually:-**
  we can also create indexes explicitly by useing create index ~~Previlieges~~ Privleges.

  syntax:-

  create index indexname on tablename (colname);

- In all Database whenever select 'select query' contain where or order by clause then only DB server checks those columns having indexes in DB.

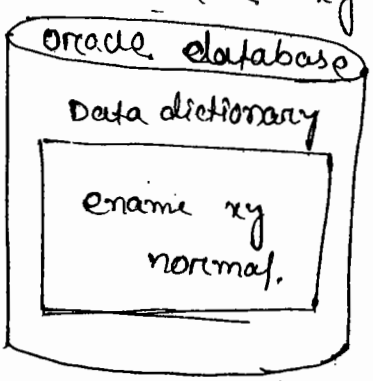
whenever indexes are available then DB servers very fastly retrive data from DB.

NB:- In oracle when a where class contain not equal to (!=), is null, is not null operators, then oracle server does not search for indexes, if those cols already heaving indexes also.

- Generally indexes are created by DBA only.
- Oracle heaving two types of indexes:-
  1) Btree indexes.
  2) bitmap indexes.

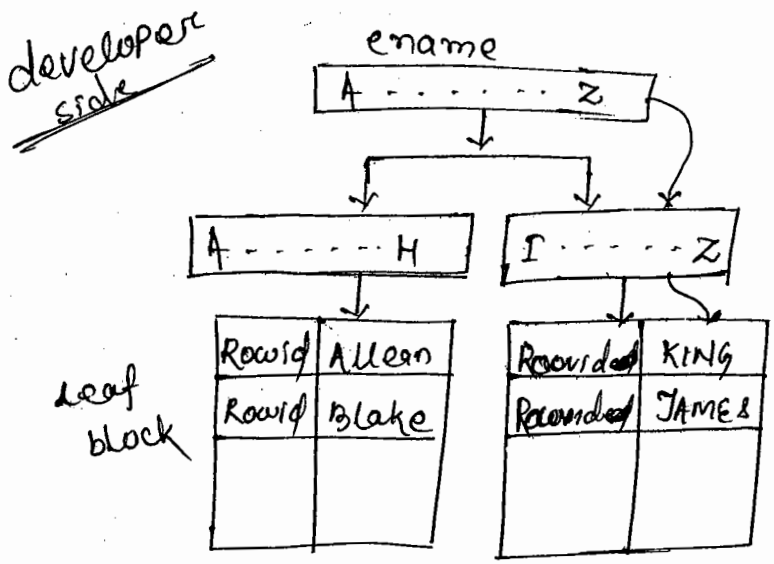
- b-tree indexes:-
  in oracle by default indexes are b-tree indexes.
- whenever we are creating b-tree indexes then Oracle server automatically creates a b-tree structure.
- In this b-tree str always leaf blocks stores actual data along with rowids.
- whenever we are submitting index column in where clauses then oracle server searches index col available in oracle database, if it is available then oracle server uses index scan method to search/retrive requested data for b-tree structure.

sql> create index xy on emp(ename);



DBA side

say select * from emp where ename='KING';



↳ all indexes information stored under user_indexes data dictionary.

↳ say create index xy on emp(ename);

say desc user_indexes;

say select index_name, index_type from user_indexes where table_name = 'EMP';

o/p:-

INDEX_NAME	INDEX_TYPE
PK_EMP	NORMAL
XY	NORMAL

NB :- If we want to view col names along with index name then we are useing USER_IND_COLUMENS Data dictionary.

eg:- say desc user_ind_columns;

say select index_name, column_name from user_ind_columns where Table_name = 'EMP';

o/p:-

Index_name	column_name
PK_EMP	EMPNO.
XY	ENAME.

Mag 7 18/03/14

↳ Oracle haveing two types of b-tree indexes.
1) non-unique btree indexes.
2) unique btree indexes.

↳ By default automatic indexes are unique btree indexes, if we want to create unique btree indexes manually then we are useing following syntax.
syntax:-

```
create unique index indexname on tablename
(colname);
```

NB:- we are not allowed to create unique indexes on duplicate value columns.

say create unique index ab on emp (Job);
error: cannot create unique indexes; duplicate keys found.

bitmap indexes :—

↳ Oracle 7.3 introduced bitmap indexes.
↳ Generally bit map indexes are used in data warehousing application.
↳ Generally bitmap indexes are created on low cardinality columns.
↳ These cols heaving more duplicate data.
syntax:-

```
create bitmap index indexname on table name
(colname);
```

↳ say create bitmap index ab on emp (Job);
↳ when ever we are submiting bitmap index then oracle server automatically create a bitmap table.

↳ whenever user requesting a data using logical operators, then Oracle server directly operates bits with in bitmap tables. then result bitmap automatically converted into rowid using an internal bitmap functions.

↳ create bitmap index ab on emp(Job).

JOB	1	2	3	4	5	6	7	8	9	10	11	12	13	14
CLERK	1	0	0	0	0	0	0	0	0	0	1	1	0	1
SALESMAN	0	1	1	0	1	0	0	0	0	1	0	0	0	0
MANAGER	0	0	0	1	0	1	1	0	0	0	0	0	0	0
ANALYST	0	0	0	0	0	0	0	1	0	0	0	0	1	0
PRESIDENT	0	0	0	0	0	0	0	0	1	0	0	0	0	0

function based indexes:—

Oracle 8i introduced function based indexes.

↳ By default function based indexes are btree indexes.

↳ In oracle whenever 'where' class contains functions or expressions then oracle server doesn't search for indexes if those cols already having indexes also, to overcome this Problem oracle 8i introduced function based indexes.

syntax:—

```
create index indexname on tablename(funname
                                    (colname));
```

eg:- create index in1 on emp(upper(ename));
- index created.

say Select index_name, index_type from user_indexes where table_name = 'EMP';

o/p:-

Index_name	index_Type
IN1	function-based Normal.

Virtual columns:-

- Oracle 11g introduced virtual cols in a table.
- In Oracle DB to store stored expressions then we are useing views are function based indexes.
- These two methods indirectly stores stored expressions in DB.
- Oracle 11g introduced to store stored expressions directly in a table. useing virtual cols through generated always as clause.
- syntax:-

```
colname datatype(size) generated always as (function
    name or expression (colname));
                                  ↳ [virtual] ← optional
```

- say create table test (a number(10), b number(10), c number(10) generated always as (a+b) virtual);

say insert into test(a,b) values(10, 20);

say select * from test;

a	b	c
10	20	30

- N.B if we want to view virtual col express, then we are useing data_default property from user_tab_columns data dictionary.

say desc user_tab_columns;

- we can also drop index using drop index indexname.

Set operators :-

- set of operators are used to retrive data from single and multiple tables.
- These operators are also called as <u>vertical joins</u>.

These are:
- union → (Return values one time only)
- union all → (unique + duplicate)
- ~~union~~
- intersect → (Common values)
- minus → (values are in first query those values are not in second query)

<u>Exm</u>:- select Job from emp where deptno = 10 Union
select Job from emp where deptno = 20;

- when we are useing set of operation corresponding Expression must belongs to same datatype otherwise current server returns an Error.

<u>ex</u>:- select <u>deptno</u> from emp union
select <u>dname</u> from dept;

error: expression must have same datatypes as corresponding expressions.

say select <u>dname</u> from emp union
select dname from dept;
o/p:- <u>ename</u>

- always set of operators returns first query colnames and colheadings.

say select <u>dname</u> from dept union
select dname from emp;
o/p:- <u>dname</u>

<u>N:B</u> we can also retrive data useing set of operators If corresponding expression not belongs to same datatype also, in this case we must use corresponding type conversion functions.

say select deptno "deptno", to-char(null) "deptnames"
from emp union
select to-number(null), dname from dept.

deptno	deptnames
10	Accounting
20	operations
30	Research.
	sales.

Conversions:—

converting one datatype into another datatype is called conversions.

→ oracle heaving two types of conversions.

1) implicit conversions.
2) explicit conversions.

Mar 7 implicit conversions (or) automatic conversion 19/03/14

→ Note:—

1) In oracle when an expression contain strings and also that string representing few number then oracle server automatically converts that the string in to number datatype.

say select sal + '100' from emp;

2) when ever we are passing number type in to character fun then oracle server automatically converts number type into character type.

eg:- say select length (1111) from dual.

o/p:- length (1111)
4

3) In oracle when ever we are passing date string in to date functions then oracle server automatically converts date-string in to date-type. But in this case date string format must be

default date format.

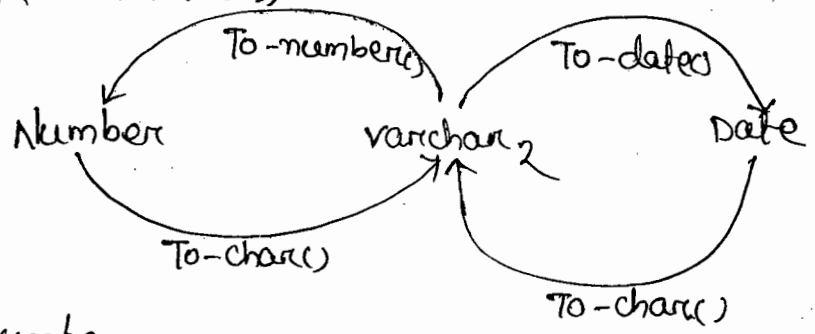
sql> select last_day('17-Aug-05') from dual.

o/p :- 31-Aug-05.

1) Implicit conversion Table :-

from	To	Assignment	Evaluation of expression
varchar2	num.	Y	Y
varchar2	date	Y	Y
Number	varchar2	Y	N
Date	varchar2	Y	N

2) Explicit conversions:-



↳ To-number() :-

converting a string representing numeric value with format into numeric value without format.

sql> select to-number('$45.6', '$99.9') +3 from dual.

o/p :- 48.6

↳ To-char() :-

This is a overloading fun^n which is used to convert number type into char type and also used to convert date type into date string.

sql> select to-char (123456.78, '$999,999.99') from dual;

$123,456.78

say select nvl(to-char(mgr), 'no manager') from emp;

~~To~~ say select to-char (sysdate, 'dd/mm/yy') from dual;

19/03/14.

To-date() :—

It is used to convert datestrings into datetype.

decode() :—

decode is a conversion function which is used to decodeing the values i.e it converts one type into another type.

↳ it is also same as if-then-else construct in PL/SQL.

↳ decode() internal uses equality (=) operator.

syntax :—

```
decode (colname, value1, stmt1, value2, stmt2
                                ---- stmt);
```

say select ename, sal, ~~other~~ deptno, decode (deptno, 10, 'ten', 20, 'twenty', 'other') from emp;

o/p:—

10	ten
20	twnty
30	other

NB:— if we want to count diff data values for a sing column then we must use 'decode function' with in 'group by' clause.

say select Job sum(decode (deptno, 10, sal)) "deptno10", sum(decode (deptno, 20, sal)) "deptno20", sum (decode(deptno, 30, sal)) "deptno 30" from emp group by job;

o/p:—

JOB	deptno10	deptno20	deptno 30
clerk	1300	2100	950
salesman	—	—	5600
President	5000	—	—
Manager	2450	2975	2850
Analyst	—	6000	—

Maa 20/03/14

```
sqly Select dname, sum(decode (Job,'CLERK',1,0)) "CLERKS",
     sum (decode (Job, 'SALESMAN',1,0)) "salesmans",
     sum(decode(Job,'MANAGER',1,0)) "Managers",
     sum(decode(Job, 'ANALYST',1,0)) "Analysts",
     sum(decode(Job,'PRESIDENT',1,0)) "Presidents"
     from emp e, dept d where e.deptno = d.deptno
     group by dname;
```

o/p:-

Dname	clerks	salesmans	Manager	Analyst	Presiden.
Accounting	1	0	1	0	1
Research	2	0	1	2	0
sales	1	4	1	0	0

NB:- if we want to modify data conditionally then we can also used 'decode' function in update statement

ex:-

```
sqly update emp set common = decode (Job, 'CLERK',
     sal*0.1, 'SALESMAN', sal*0.2, 'MANAGER', sal*0.3,
     'ANALYST', SAL*0.5, SAL*0.6);
```

Case statement:-

Oracle 8.0 introduced case statement & also Oracle 8i introduced case conditional statements.

↳ Case statement also used to decoding the values & it also same as if-then-else if construct in PL/SQL.

↳ Case statement performance is very high compare to decode function.

NB:-

Decode internally uses equality operator, where as in Case statement explicitly we are using all SQL

operators.

Method 1 :- (syntax)

```
case colname
when value1 then statement1
when value2 then statement2
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
else statement end.
```

ex:- select ename, sal, deptno, case deptno
when 10 then 'ten'
when 20 then 'twenty'
else 'others' end from emp;

Method 2 :- (case conditional statement)

syntax:

```
case
when column condition then statement1
when column condition then statement2
- - - - - - - - - - - - - - -
- - - - - - - - - - - - - - -
else statements end
```

ex:- select ename, sal
case
when sal < 1000 then 'low salary'
when sal between 1000 and 3000 then 'Medium sal'
when sal in(3100, 4000, 4500) then 'special sal'
else 'other sal' end from EMP;

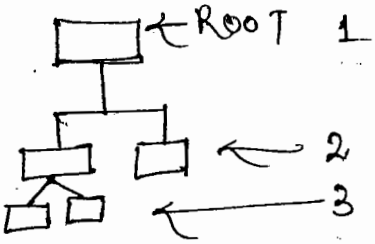
Hierarchy Querys :-

- In Relational DB, we can also store hirearchical data.
- if we want to store hirarchial data Relational table must contain min 2 cols and also

- this cols must belongs to same datatype.
- if we want to retrive hirarchial data then we are useing <u>Level pseudo col</u>.
- Along with Level pseudo col oracle provided following clauses.
  1) start with
  2) connect by

<u>Level</u>:-

Level is a ~~super~~ pseudo col which assign numbers to each level in a tree structure.



1) <u>Start with</u> :-

- it specified searching condition in tree structure.

syntax:-

```
startwith condition
```

2) <u>connect by</u> :-

- Connect by clause ~~by~~ specified ~~by~~ Relationship betn Parent, child col useing a <u>Prior operator</u>.

<u>syntax</u>:-

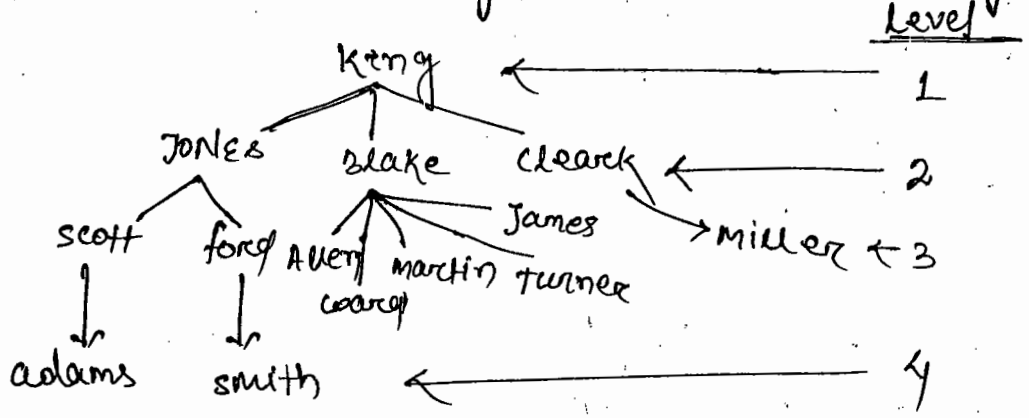
```
Connect by Prior Parentcol = childcolname
```

syntax:-

```
select colname1, level from tablename where
condition start with condition connect by prior
Parentcol = childcolname;
```

Q: write a hirarchial query to display hirarchy from emp table using empno, mgr cols.

A: sol: Select level, ename from emp start with ename='KING' connect by prior empno=mgr;



NB1 :- Oracle 9i introduced sys_connect_by_Path() which returns Path of the hirarchy in tree structure.

syntax :-

(special symbol)

sys_connect_by_Path(colname, 'delimitermame')

eg:- select level, sys_connect_by_Path(ename, '/') from emp start with mgr is null connect by Prior empno=mgr;

Q: write a hirarchy query to display the emp who are working under blake from emp table.

A: select level, sys_connect_by_Path(ename, '→') from emp start with ename='BLAKE' connect by prior empno=mgr;

→ select level, sys_connect_by_Path(ename, '→') from emp start with ename='MILLER' connect by empno=PRIOR mgr;

o/p :- 1 → Miller
2 → Miller → Cleark
3 → Miller → Cleark → King

PRIOR:- PRIOR is a unary operator which specified parent column.

NB:- Oracle 10g introduced connect_by_isleaf pseudo column which returns '1' to the leaf nodes and zero to the all other nodes.

eg:- select leaf, ename, connect_by_isleaf from emp start with mgr is null connect by Prior empno = mgr;

NB Oracle 10g introduced connect_by_root operator which returns Root node in hierarchy.

syntax:-

connect_by_root colname

↳ select level, ename, connect_by_root ename from emp start with mgr is null connect by Prior empno = mgr;

Mar 7 Lock:- 21/03/14

↳ Locking is a mechanism to prevents another user excess ~~when ever~~ for resources.

↳ All DB systems hearing 2 types locks:-

1) Row level Locks.
2) Table level locks.

↳ Row level Locks:-

↳ in this Method we locking set of rows using for update clause.

↳ This clause used in select statement only.

syntax:-

select * from tablename where condition for update [nowait];

↳ when we are performing locks another user query the data but can't perform DML operation and also when we are useing commit & Rollback automatically locks are ~~real~~ released.

Session1	Session2
scott	soma
sql> conn scott/ Tiger ↵	sql> conn soma/ soma ↵
sql> Select * from emp where deptno=10 for update;	sql> update scott.emp set sal=sal+100 where deptno=10;
sql>commit; [for Releasing locks]	[we can not perform DMLS]

no wait :-

↳ This is an optional clause, used along with for update clause.

↳ when we are useing this clause oracle server automatically returns control to the current session, if another user not realearing clause also.

↳ in this case oracle server returns an Error :-

ORA-00054 : resource busy

Session1 scott	Session2 soma
sql> conn scott/ Tiger ↵	sql> conn soma/soma ↵
sql> select * from emp where deptno=10 for update nowait; ORA-00054: Resource busy	sql> select * from scott emp where deptno=10 for update;

Table level Locks :-

It is used by dba only.

↳ in this Method we are locking a table, where are two types of table level locks used by user

1) Shared lock;
2) exclusive lock;

Shared locks :-

when we are useing this lock another user Query the data, & that can't perform DML data & also no. of users locks the resources.

syntax :-

```
Lock table tablename in share. Mode;
```

Session1 scott	Session2 soma
sql> conn sys scott/tiger ↵	sql> conn sys ~~soma~~ soma/soma ↵
sql> lock table emp in share mode;	sql> select * from scott.emp; ✓
	sql> lock table scott.emp in shared mode; ✓
	sql> update scott.emp set sal = sal +100; [we cannot perform DMLs] ×

2) exclusive lock :-

when we are using this lock another user Query Data & they can not perform DML operations. but at a time only one user locked the resources;

syntax :-

```
lock table tablename in exclusive mode;
```

Session1 scott	Session2 soma
sql> conn sys scott/tiger ↵	sql> conn sys soma/soma ↵
sql> lock table emp in share mode;	sql> select * from scott.emp; ✓
	sql> lock table scott.emp in shared mode; ×

NB:- in all DB systems, when we are useing cursor locking in mechanis all database servers internaly uses exclusive locks.

nested table :-

Table within another table is called nested table.

- Oracle 8.0 introduced nested tables.
- nested table is a user defined type which is used to store no. of data items in a single unit.
- in oracle we can also create our own datatypes useing type keyword.

Step 1 :- (creating an object)

object is a user defined type which is used to represent diff data types in to single unit.

- it is also same as structures in C language.

syntax :-

```
create or replace type typename as object
(attribute1 datatype (size), attribute2 datatype (size),..);
```

Step 2 :- (creating an nested table type)

syntax :-

```
create or replace type typename as table of object
type;
```

Step 3 :- (create a original / actual table)

syntax :-

```
create table tablename (col1 datatype (size),
                        col2 datatype (size),....,
 col nestedtabletype) nested table colname store
 as anyname;
```

eg:- say create or replace type g1 as object (bookid number(10), bookname varchar2(10), Price number(10));
/

say create or replace type fan as table of g1;
/

say create table student (stno number(10), name varchar2(10) col3 fan) nested table col3 store as abcf;

o/p:- desc student ;

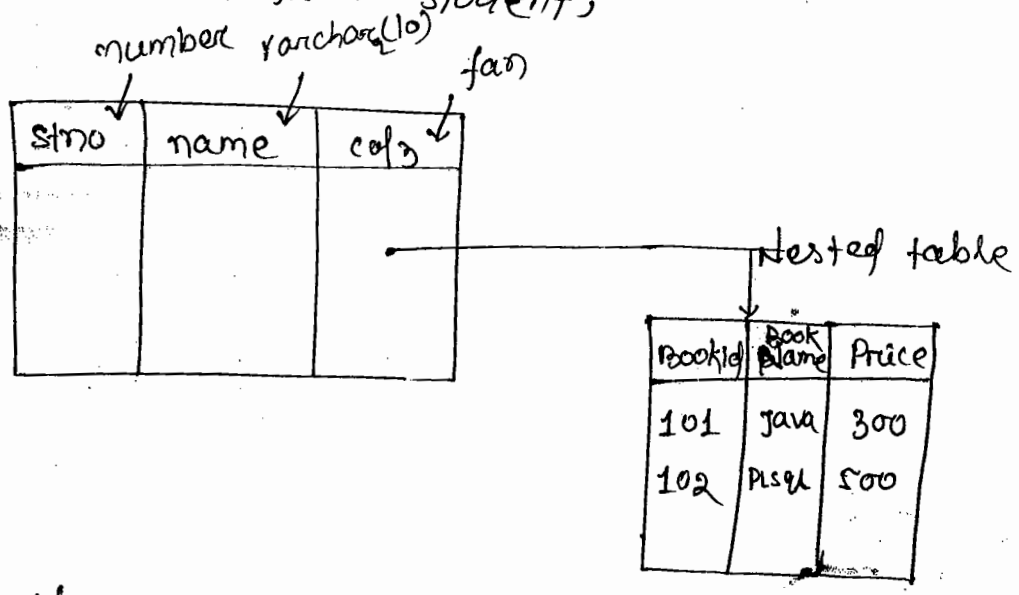
name	type
STNO	NUMBER(10)
NAME	VARCHAR2(10)
COL3	FAN

↳ If u want to store data into nested table type then we must use constructor.

↳ Here constructor name is same as type name.

say insert into student values(1, 'murali', fan(g1(101, 'java', 300), g1(102, 'plsql', 500)));

say select * from student;



NB:- if we want to view nested table data then we are useing table operator.

Syntax:-

Select * from ~~Select~~ table (select. nestedtable colname from actual table);

say select * from table(select col3 from student);

Bookid	Bookname	Price
101	java	300
102	plsql	500

21/03/14

Mqa ? **synonyms** :-

synonym is a db object which is used to provide security of the table becoz synonyms hides another user object name

↳ synonym is a referenced name or alias name for the original names.

↳ All database systems having two types of ~~symnos~~ synonyms :-

1) Public synonyms.

2) Private ,,

↳ in all dbs by default synonyms are private synonyms, if we want to create public synonyms, then we are useing following syntax.

syntax :-

```
create public synonym synonymname for ~~username~~
username.objectname@database link;
```

scott/Tiger	Murali/Murali	soma/soma
sql> Grant all on emp to murali, soma;	sql> select * from scott.emp;	sql> select * from scott.emp.
	sql> create synonym cooler for scott.emp;	sql> select * from cooler; error: Table or view does not exits.
	sql> select * from cooler; (✓)	sql> select * from ab; (✓)
	sql> create public synonym ab for scott.emp; error: insufficent Privilege	
	sql> conn sys as sysdba; Password: ***.	
	sql> grant create public synonym to murali;	

say connect murali/murali
say create public synonyms
ab for scott.emp;
say select * from ab;

TCL (Transaction Control Language) :-

Transaction :-

It is an logical unit of work betn two points.

↳ All db system having two transactional commands

1) commit
2) Save Point

Commit :- This commit is used to save the transaction permenently to disk (DB).

syntax :-

```
commit;
```

Save point :- It is an logical mark betn transactions.

syntax :-

```
savepoint savepointname;
```

Rollback :- This commit is used to undo with transact. from memory.

```
syntax :- Rollback;
```

```
say update ....
    Delete ...
    insert ...
    savepoint a1;
    insert
    update
    savepoint a2;
    update
    Delete
    Rollback to a1;
    Comit;
```

Partitions :-

Partitions tables are created by dba on recovery and backprocess.

↳ This Process automatically improves performance of of the application.

↳ partitions tables are used in datawarehouseing application.

↳ oracle heaving 3 types of partitions.

1) range Partitions.
2) List "
3) hash " . (never used)

1) range Partitions :-

in this method partitions tables are created based on range of values.

syntax :-

```
create table tablename (col1 datatype (size),
col2 datatype (size), ...) Partition by range
(keycolname) (Partition Partitionname values
less then (value), - - - - );
```

↳ to view Particular Partitions :-

syntax :-

```
select * from tablename Partition (Partitionsnames);
```

say create table test (empno number(10), name varchar(10)) Partition by range (sal) (Partion P_1 values less then (1000), Partition P_2 values less then (2000), Partition P_3 values (3000));

say insert into. . . .

say select * from test Partition (P_1);

2) List Partitions :-

↳ Oracle 9i introduce list Partitions, using this Partitions we can also create Partition table based on char^n datatype column, in this Partitions Partition tables are created by useing list of values.

syntax :-

```
create table tablename (col1 datatype(size), col2
datatype(size) . . .) Partition by list(keycolname)
(Partition partitionname values (value1, value2, ...),
- - - - - - - Partition Partitionname values (default))
```

eg:-

sql> create table test (sno number(10), name varchar2(10))
Partition by list (name) (Partition P1 values ('india', 'Pakistan'), Partition P2 values ('us', 'uk', 'canada'),
Partition others values (default));

sql> insert into . . .

sql> select * from test;

SNo	SNAME
1	Japan
2	aus.

3) hash Partitions :-

in this Method Partitions are created by db servlere based on hash algorithm.

syntax:-

```
create table tablename (col1 datatype (size), col2 datatype
(size), - - ..) Partition by hash (key col) Partition no;
```

eg:- create table test (name varchar2(10), sal number(10))
Partition by hash (sal) Partition 5;

all Partition are stored under user_tab_Partition data dictonary.

eg:- desc user_tab_Partitions;

say select Partition_name from user_tab_Partition where table_name = 'test';

nvl2():-

↳ Oracle 9i introduced nvl2(), it is also used to replace user defined values inplace of null.

syntax:-

$\boxed{nvl_2(exp_1, exp_2, exp_3);}$

↳ if expression ~~work~~ is null, then it returns exp_3 otherwise it returns $expression_2$.

Ex:- select nvl_2(null, 10, 20) from dual;

o/p:- 20

say select nvl_2(50, 10, 20) from dual;

o/p:- 10

Q: update emp commition based on following condition.

1) if commition is null then update to 500.

2) comm is not null then update.

say ~~select emp comm(~~

say update ~~table~~ emp set comm = nvl_2(comm, comm+500, 500);

coalesce():-

This is an ANSI() which returns 1st normal expression.

eg:- Select coalesce (null, 20, null, 20, null) from dual;

o/p:- 20.

↳ nvl function internaly uses implicit conversion where as coalesce() not useing automatic conversion.

say select nvl('a', sysdate) from dual;

a

say select coalesce ('a', sysdate) from dual;

error: inconsistent datatypes: expected CHAR got Date

Max

24/03/14

—o—

PL/SQL

24/03/14

→ PL/SQL introduction :-

1) → PL/SQL introduction

* select ---- into clause
* anchor notation
* bind variables
* % type, % type.
* Conditional, control stmts.

2) → CURSORS

* implicit, explicit cursors
* explicit cursors life cycle.
* cursor for loops.
* Parameterised cursor.
* for update, where current of clauses
* implicit cursor attributes.

3) EXCEPTIONS

→ * Predefined, user defined, unnamed exception
* sql code, sqlerror.

→ raise_application_error()

4) SUBPROGRAMS

⇒ * Stored Procedures
* in, out, in out modes {Parameter Modes}
* no copy
* autonomous transactions
* authid current_user

⇒ * Stored function.
* dmls used in functions.
* function execution methods.
* Wm_Concat()

5) TRIGGERS

* row level triggers
* applications
* before/after
* statement level triggers
* trigger execution order.
* follows clause
* Mutating error introduction.
* system triggers.

6) PACKAGES

* Introduction
* serially-reusable
* Overloading Procedures.
* forward declarations.

7) TYPES USED IN PACKGED

* PL/SQL record
* index by table
* nested table
* varray
* collection methods.

8) BULK BIND

* BULK collection
* forall statements
* indicates of clause
* SQL % bulk-rowcount

9) Utl-file package

10) Lobs (clob, blob, bfile)

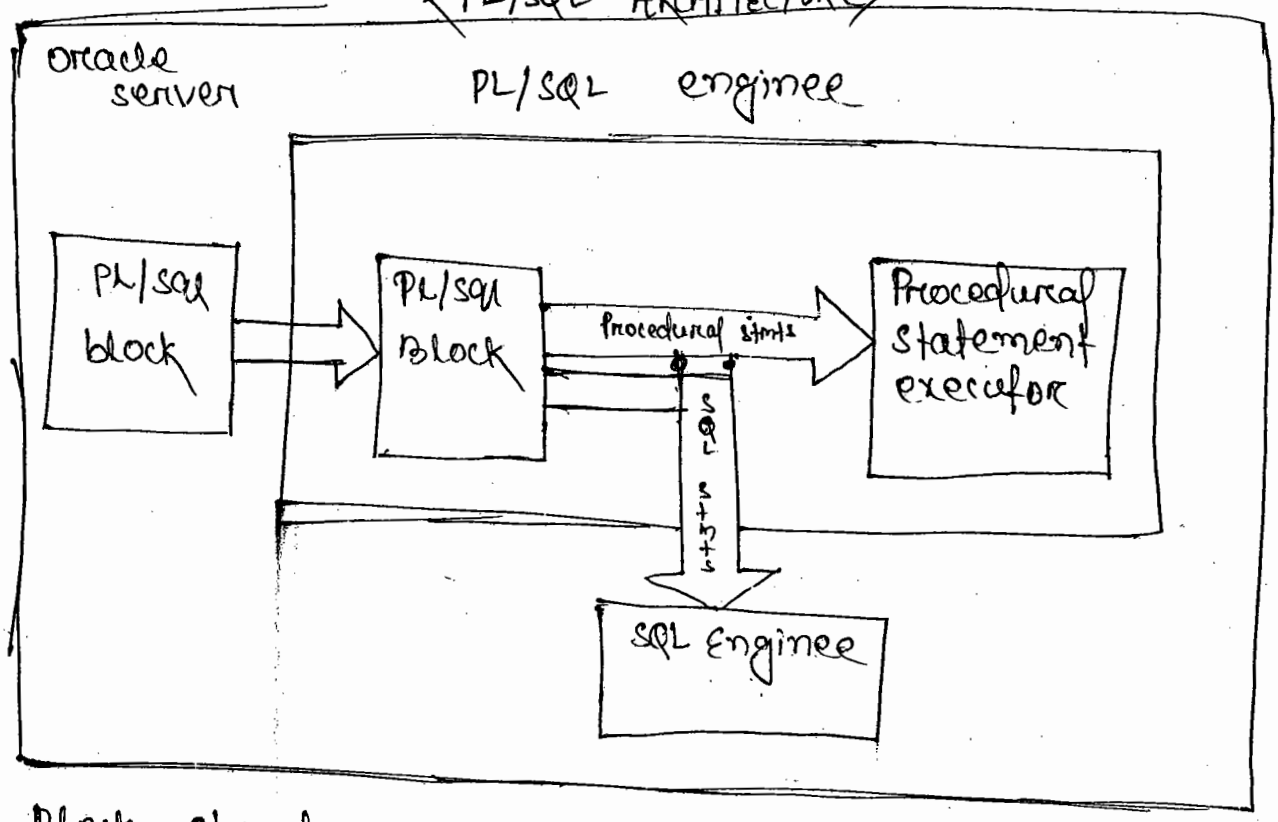
11) SQL * loader

12) nested blocks
   * local Procedures, local functions.
13) Dynamic sql.
14) Mutating error
   * Oracle 11g compound triggers.
15) dbms_pipe package.
16) member procedures, member functions
17) Oracle 11g features.

PL/SQL

- Oracle 6.0 introduced pl/sql.
- PL/sql is a Procedural language extension for sql.
- Oracle 6.0 → PL/sql 1.0
  Oracle 7.0 → PL/sql 2.0
  Oracle 8.0 → PL/sql 8.0
  -
  Oracle 11.2 → PL/sql 11.2
- basically PL/sql is a block structured Programing language.
- when we are submitting PL/sql blocks into the Oracle server, all sql statements are executed with in sql engine. and also all Procedural statements are executed with in PL/sql engine.

<PL/SQL ARCHITECTURE>



Block structure:-

Declare [optional]
→ variable declarations, cursors, user defined exceptions;

Begin [Mandatory]
→ select .... into clause;
→ DML, TCL;
→ conditional, control statement;
→ Exception [optional]
→ handling runtime errors.
end; [Mandatory]

Declaring a variable :-

syntax :- variablename datatype (size);

ex:- SQL> declare
a number (10);

ex:-

Storing a value into variable :-

↳ Useing an assignment operator (:=) we can store a value in to variable

ex:-

in oracle	in c
a number (10);	int a;
a := 50; a [50]	a = 50; a [50]
↓ Assignment operator	↓ assignment operator

syntax :- variable ename := value;

```
say declare
    a number (10);
    begin
    a := 50;
    end;
```

Display a message or variable value :-

syntax :-

```
dbms_output . put_line ('message');
  Package name      Method name
  (or)
dbms_output . put_line (variable name);
```

To view PL/SQL versions :-

↳ **say Select * from v$version;**

eg:- say set serveroutput on;
↳ if we don't write this then o/p doesn't show.

```
say begin
    dbms_output . Put_line ('welcome to PL/SQL');
    end;
    /
```

o/p :- welcome to PL/SQL

eg:- say declare

```
a number(10);
begin
a:=50;
dbms_output.out_line(a);
end;
/
```

O/P:- 50

Select.... into clause:-

This clause is used to retrive data from table and stored in to PL/sql variables.

↳ select... into clause always returns single record or single value at a time.

syntax:-

```
select colname1, col2 .... into var1, var2 .....
from tablename where condition;
```

↳ This clause used in executable section of the PL/sql block.

Q: W.A. PL/sql Program for user entered empno display name of the employee and his salary from emp table.

Ans: A declare

25/03/14

```
v_ename varchar2(10);
v_sal number(10);
begin
select ename, sal, into
v_ename, v_sal from emp
where empno=&no;
```

```
dbms_output.put_line
(v_ename || ' ' || v_sal);
end;
/
```

O/P:- Enter value for no: 7902
FORD 3000

NB: In PL/SQL when we are using 'not null' or 'constant' clauses then we must assign the value at the time of declearing the variable in declear section of PL/SQL block.

eg:-
```
declare
a number(10) not null := 50;
b constant ~~to~~ number(10) := 8;
begin
dbms_output.put_line(a);
dbms_output.put_line(b);
end;
/
```

NB: we can also use default caluse inplace of assignment operator when we are assigning the value at the time of variable declaration in declare section of PL/SQL block.

eg:-
```
declare
a number(10) default 50;
begin
dbms_output.put_line(a);
end;
/
```

O/P:- 50

↳ W.A. PL/SQL Program retrieve max salary from emp table and stored it in to PL/SQL variable and also display that salary.

~~declare~~
~~a number(10)~~
~~v_ename varchar2(10);~~
~~v_sal number(10);~~
~~begin~~
~~Select ename, sal~~

```
declare
  v_sal number(10);
  begin
  Select max(sal) in to v_sal
  from emp;
  dbms_output.put_line(v_sa
  end;
  /
```

o/p:- 5200

```
declare
  a number (10);
  b number (10);
  c number (10);
  begin
  a:= 80;
  b:= 30;
  c := greatest(a,b); → [c:= max (a,b)] X wrong
  dbms_output.put_line(c);
  end;
  /
```

o/p:- 80

NB:- in PL/SQL expressions, we are not allowed to use group functions, decode, conversion function ~~with in PL/SQL~~ ~~expression~~ but we are allowed to use number functions, character functions, date functions in PL/SQL expressions.

eg:-
```
declare
  a varchar2 (10);
  begin
  a:= upper('soma');
  dbms_output.put_line(a);
  end;
  /
```

o/p:- SOMA

Variable attributes :-

variable attributes are used inplace of datatypes in variable declaration.

- These variable attributes are also called as anchor notations.
- when we are useing these attribute oracle server automatically allocates memory for the variables based on the column datatypes in a table.
- Oracle heaving two types of variable attributes There are :-
  1) column level attributes.
  2) Row level attributes.
- Column level attributes :-
  In this Method we are defining attributes for individual columns.
- Column level attributes are represented by useing percentage (%) type.

syntax :-

```
variable name tablename.colname % type ;
```

- when we are useing this attribute oracle server automatically allocates Memory for the variables based on corresponding col datatype from a table.

eg:-
```
declare
v_ename    emp.ename%type;
v_sal      emp.sal%type;
begin
select ename, sal into
v_ename, v_sal from emp
where empno=&no;
dbms_output.put_line(v_ename || ' ' || v_sal);
end;
```

2) Row level attributes :-

In this Method a single variable can represent all diff data types in a ~~record~~ row within a table. This variable is also called as <u>record type variable</u>.

↳ it is also same as structure of in C-language.

↳ These attributes are represented by <u>useing % rowtype</u>.

syntax :-

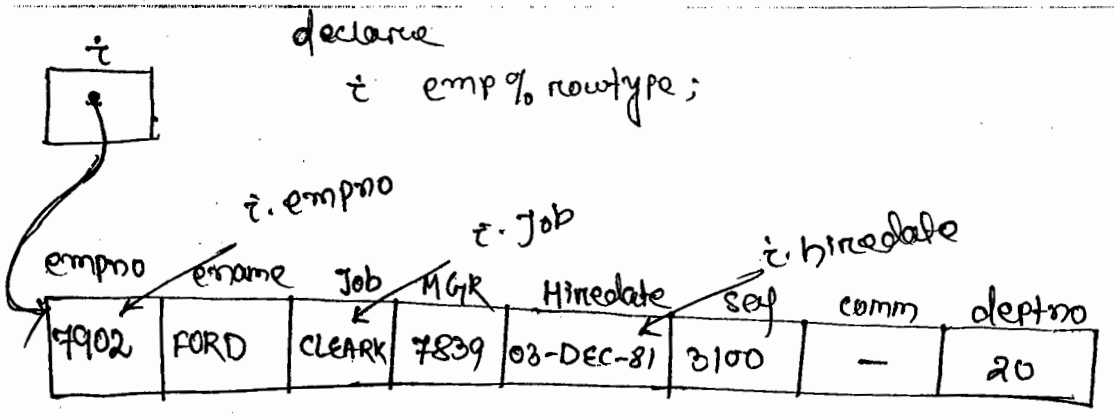
variablename tablename%rowtype ;

eg:-
```
declare
i emp%rowtype ;
begin
select ename , sal , hiredate , deptno
into i.ename , i. sal , i. hiredate,
i. deptno from emp where empno=&no;
dbms_output . Put_Line (i. ename || ' ' || i.sal || ' ' ||
i.hiredate || ' ' || i.deptno);
end;
/
```

(alternative way)

```
declare
i emp%rowtype ;
begin
select * into i from emp where empno =&no;
dbms_output . out_Line ( i.ename || ' ' || i.sal || ' ' ||
i.hiredate || ' ' || i.deptno);
end;
/
```

O/P:- Enter value for. no : 7902
FORD 3100 03-DEC-81 20



Mag 7 28/03/14

datatypes and variables :-

1) it supports all sql datatypes (also called as scalar datatypes). + boolean datatypes.
2) Composit datatypes
3) Ref obj.
4) Lobs (large objects) → clob, blob, bite.
5) non PL/sql variables (or) bind variables.

Conditional statements :-

1) if
2) if-else
3) elsif

if :- syntax :- if condition then
stmts;
end if;

if-else :- syntax :- if condition then
stmts;
else
stmts;
end if;

elsif :- To check more no. of conditions then we are using elsif.

syntax :- if condition1 then
stmts;
elsif condition2 then

```
    stmts;
elsif condition3 then
    stmts;
  - - - - -
  - - - - - - -
else
    stmts;
end if;
```

eg:- declare

```
v_deptno number(10);
begin
select deptno into v_deptno
from dept where deptno=&no;
if v_deptno=10 then
dbms_output.put_line('ten');
elseif v_deptno=20 then
dbms_output.put_line('twenty');
elsif v_deptno=30 then
dbms_output.put_line('thirty');
else
dbms_output.put_line('others');
end if;
end;
/
```

o/p:- Enter value for no: 40
others

sqlY/ Enter value for no: 90
error.

NB:- when PL/SQL block Package select into clause is also if requested data not available in a table then oracle server returns an error

ORA-1403 : no data found.

NB2 :- when PL/SQL block contains DML statements and also if requested data not available in a table then oracle server does't return any error messages to handled these types of blocks we are useing implicit. cursor attributes.

eg :- begin
delete from emp where
ename = 'welcome';
end;

o/p :- PL/SQL Procedure successfully complited.

NB3 :- when ever select into clause try to return multiple values at a time then oracle server returns an error:

ORA - 1422 : Exact fetch returns more then requested number of rows.

Control statements (loops) :-

1) Simple loop
2) while loop
3) for loop

Simple loop :-

This loop is also called as infanit loop. Here ~~what~~ body of the loops statements are executed repeatetly. ~~syntax~~

syntax :- loop
" stmts;
end loop;

↳ To exit from infanit loop oracle Provided two methods.

1) method 1 :- exit where True condition;

eg:-
```
declare
n number (10) := 1;
begin
Loop
dbms_output.Put_Line(n);
exit when n>=10;
n:=n+1;
end loop;
end;
/
```

O/P :- 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

Method 2 :- useing if

syntax :-
```
if true condition then
    exit;
end if;
```

eg:-
```
declare
n number(10) := 1;
begin
Loop
dbms_output.Put_Line(n);
if n>=10 then
exit;
end if;
n := n+1;
end loop;
end;
/
```

O/P :- 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

while loop :-

Here body of the loop statements are exected repeatetly until stmts is false.

syntax :-
```
while condition
loop
    stmts;
end loop;
```

eg:-
```
declare
n number(10) := 1;
begin
while(n<=10)
loop
dbms_output.put_line(n);
n:=n+1;
end loop;
end;
/
```

o/p:- 1
2
⋮
10

for loop:-

syntax:-
```
for index variablename in lowerbound
.. upperbound loop
loop
  stmts;
end loop;
end
```

eg:-
```
declare
n number(10);
begin
for n in 1..10
loop
dbms_output.put_line(n);
end loop;
end;
/
```

o/p:- 1, 2 ------10

```
declare
n number(10);
begin
for n in reverse 1..10
loop
dbms_output.put_line(n);
end loop;
end;
/
```

o/p:- 10, 9, 8 ------1

```
begin
for i in 1..10
loop
dbms_output.put_line(i);
end loop;
end;
/
```

NB:- ~~in~~ ~~one~~ for loops index variable internally behaves like a integer variable.

Mar 7 26/03/14

bind variables :-

- bind variables are session variables.
- These variables are create a host environment. That's why these variables are also called as ~~bi~~ host variables.
- These variables are used in SQL, PL/SQL, Dynamic SQL.
- in PL/SQL when a subprogram having out (or) ~~inout~~ in out Parameters. then those subprog must be executed through bind variables only.

Step1 :- creating a bind variable :-

syntax :-

```
SQL> variable variablename datatype;
```

Step2 :- useing bind variable :-

syntax :-

```
: variablename.
```

Step3 :- display variable value :-

syntax :-

```
SQL> Print variablename;
```

eg :-

```
SQL> variable g number;
SQL> declare
     a number(10) := 500;
     begin
     :g := a/2;
     end;
     /
SQL> Print g;
       G
     ---
     250
```

↳ PL/SQL having two types of blocks.

1) Annoymous blocks. [These blocks does't
2) named blocks.

- These blocks does't have a name.
- These blocks are not stored permimently into DB
- Not to call in client applications.

Ex:- declare
- - - - -
begin
- - - -
end;

named blocks

- These blocks having a name
- These blocks are automatically permimently stored in DB.
- These blocks are calling in client applications.

ex:- Procedures, functions, Triggers, Packages.

5* Imp

Cursor :-

cursor is an private SQL memory area which is used to process multiple records and also this is an record by record process.

↳ all DB system haveing two types of static cursors:-

1) implicit cursors.
2) explicit cursors.

implicit cursors :-

for SQL statements returns single record is called implicit cursors.

↳ implicit cursors memory area is also called as context area/SQL area.

↳ when an PL/SQL block contain select---into clause or DML statements then oracle server

Automatically creates a memory is also called as context area or SQL area or implicit ~~area~~. cursor.

↳ This memory area returns single record when PL/SQL block contain select ... into clause.

↳ where as this memory area returns multiple record when PL/SQL block contains DML stmts. but these multiple record process at a time by SQL engine.

eg:- sql> declare

```
v_ename varchar2(10);
v_sal number(10);
begin
select ename, sal into v_ename, v_sal from emp
where empno = &no;
dbms_output.out_line(v_ename|| ' ' || v_sal);
end;
/
```

o/p:- Enter value for no : 7902.
FORD 3000

2) Explicit cursor

↳ For SQL stmts returns multiple records is called as Explicit cursor. and also this is a record by record process.

↳ Explicit cursor memory area is also called as active set area.

Explicit cursor lifecycle :-

1) declare
2) open
3) fetch
4) close.

1) **declare :-**

in declare section of the PL/SQL blocks they are defining the cursor useing following syntax.

syntax :-
cursor cursorname is select statement;

ex:- declare
cursor c1 is select * from emp where job = 'CLERK';

2) **open :-**

when over we r open in the cursor then only oracle server transfer data from table into cursor memory area.

↳ becoz in all database systems when we are open in the cursor then only cursor select stmts are executed.

↳ syntax :-
open cursorname;

↳ This stmts used in executable section of the PL/SQL block.

NB ① when ever we are open the cursor implicitly cursor Pointers Point to the 1st record of the cursor.

3) **Fetch :-** (fetching data from cursor)

useing fetch stmts we are featching data from cursor in to PL/SQL variables.

syntax :-
fetch cursorname into variablename1, variablename2, ... ;

4) close :-

when ever we are close the cursor all the resources allocated from cursor memory area automatically realised.

syntax:- close cursorname;

eg:-
```
declare
cursor c1 is select ename, sal from emp;
v_ename varchar2(10);
v_sal number(10);
begin
open c1;
fetch c1 into v_ename, v_sal;
dbms_output.put_line(v_ename||' '||v_sal);
fetch c1 into v_ename, v_sal;
dbms_output.put_line('my second emp name is'||' '
                     v_ename);
fetch c1 into v_ename, v_sal;
dbms_output.put_line(v_ename||' '||v_high salary);
close c1;
end;
/
```

O/P:- SMITH 900
My second emp name is ALLEN
WARD high salary.

Maa

PL/SQL

07/06/14

Two types of blocks :-

Types of blocks

1) Annonymous blocks.

→ These blocks does not have any name.

→ These blocks are not stored in database

ex:-
```
declare
  - - -
  - - -
Begin
  - - -
  - - -
end;
```

→ These blocks are not allowed to call in client applications.

2) Named blocks.

→ These blocks having a name.

→ These blocks are automatically stored in DB.

ex:- Procedures, functions.

Declaring a Variable :-

Syntax :-

Variablename datatype(size) ;

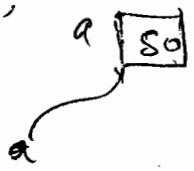
ex:-

PL/SQL

```
sql> declare
     a number(10);
     b varchar2(10);
     a := 50;
     b := a;
```

C

```
int a;
a = 50;
```



Storing a Value into Variable :—

Using assignment operator (:=). we can stored a value into Variable.

syntax :—

```
variablename := value;
```

```
sql> declare
     a number(10);
     begin
     a:=50;
     end;
     /
```

Display a Message (or) variable value :—

syntax :—

```
dbms_output.put_line('Message');
```

(or)

```
dbms_output.put_line(variablename);
```

Package name Method

↳ This Packaged used in either executable section or in exception section of PL/SQL blocks.

To view PL/SQL (or) oracle version :—

```
sql> Select * from v$version;
```

Ex:-

```
sql> Set Serveroutput on;
sql> begin
     dbms-output. Put_line ('welcome');
     end;
     /
```

o/p :- Welcome.

Ex:-

```
sql> set serveroutput on;
sql> declare
        a number(10);
     begin
        a:=50
        dbms_output. Put_line(a);
     end;
     /
```

o/p :- 50

Select.... into clause :-

This clause is used to retrive data from table & stored into PL/SQL variables.

↳ Select.... into clause always returns Single record or single value at a time.

Syntax:-

```
Select col1, col2 .... into var1, var2 ....
from tablename where condition;
```

↳ This claused used in executable ~~block~~ selection of the PL/SQL block.

Q: W.A. PL/SQL Program for user entered a no display name of the employee & his salary from emp table.

```
set serveroutput on;
sal Declare
    v_ename varchar2(10);
    v_sal number(10);
begin
    select ename, sal into v_ename, v_sal from
     emp where empno = &no;
    dbms_output.put_line(v_ename||' '|| v_sal);
end;
/
```

o/p :- Enter value for no: 7369
SMITH 8000

emp

empno	ename	sal
7369	SMITH	8000
7566	JONES	2000
-	-	-
-	-	-

Maa

Mar 7 example :- 12/06/14

```
declare
cursor c1 is select ename, sal from emp;
v_ename varchar2 (10);
v_sal number (10);
begin
  open c1;
fetch c1 into v_ename, v_sal;
dbms_output.put_line(v_ename || ' ' || v_sal);
fetch c1 into v_ename, v_sal;
dbms_output.put_line ('my second emp name
                        is' || v_ename);
fetch c1 into v_ename, v_sal;
dbms_output.put_line (v_ename || ' ' || high sal);
close c1;
end;
/
```

O/P :- SMITH 1500
My second employee name is ALLEN
WARD high salary.

cursor → To Process Multiple record.
→ record by record Process.

explicit cursors attributes :~

Every explicit cursors having 4 attributes

These are :—

* 1) ~~Cursor~~ % notfound.
  2) % found.
  3) % isopen.

  } (Return boolean value True or false)

* 4) % Rowcount. } (Return number ~~datat~~ datatypes).

when we are useing these attributes in PL/SQL block then we must specified a cursor name along with these attributes.

syntax:-

$\boxed{\text{cursorname\% attributename}}$

Except %Rowcount all other cursors attributes returns boolean value either true or false, where as %rowcount always returns number datatype.

1) % notfound :—

This attribute always returns boolean values either true or false.

This attribute always returns true if cursors does not have records after fetching at least one record from the cursor.

Q) W.A PL/SQL cursor Program to display all employee names and their salaries from emp table useing %notfound attribute ?

A)
```
declare
   cursor c1 is select ename, sal from emp;
   v_ename   varchar2(10);
   v_sal     number(10);
begin
   open c1;
   loop
   fetch c1 into v_ename, v_sal;
   exit when c1%notfound;
   dbms_output.put_line(v_ename || ' ' || v_sal);
   end loop;
   close c1;
end;
/
```

Q) W.A. PL/SQL cursor Program to display 1st 5 higest sal emp from emp table using %rowcount attribute.

A)
```
declare
   cursor c1 is select ename, sal from emp
     order by sal desc;
   v_ename   varchar2(10);
   v_sal     number(10);
begin
   open c1;
      loop
         fetch c1 into v_ename, v_sal;
         dbms_output.put_line(v_ename || ' ' ||
                                v_sal);
         exit when c1%rowcount >= 5;
      end loop;
   close c1;
end;
```

Q) W.A. PL/SQL cursor Program to display even no. of records from emp table useing %rowcount attribute.

A)
```
declare
    cursor c1 is select ename, sal from emp;
    v-ename   varchar2(10);
    v-sal     number(10);
begin
   open c1;
     loop
       fetch c1 into v-ename, v-sal;
       exit when c1%rowcount;
        if mod (c1%rowcount, 2) = 0 then
         dbms-output. Put-line (v-ename || ' '
                                  v-sal);
        end if;
      end loop;
     closed c1;
  end;
  /
```

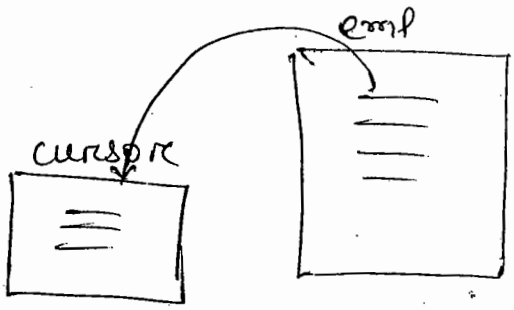
- when ever we are creating a cursor automatically 4 memory locations are created along with cursors memory area.
- These memory locations are viewes like a Variable.
- These memory locations are identify through cursors attribles.
- These variable also stores single value at a time.

Ex:-

```
declare
  a number(10);
  b boolean;
begin
  a:=50;
  b:= True;
```

a: 50

b: true



c1%notfound	c1%found	c1%isopen	c1%rowcount
True	false	true	2

Maa 7. 01/04/14

Q: W.A. PL/SQL cursor Program for Passing Job as a Parameter from emp table, display employee working as clearks or analysist and also display that report statically.

eg:— employee working as clerks
SMITH
ADAMS
JAMES
MILLER
employee working as analysts
SCOTT
FORD.

```
declare
cursor c1(P_Job varchar2) is
Select * from emp where Job = P_Job;
i emp%rowtype;
begin
open c1('CLERK');
dbms_output.put_line('employee coorking as a CLERK');
loop
fetch c1 into i;
exit when c1%notfound;
dbms_output.put_line(i.ename);
end loop;
close c1;
open c1('ANALYST');
dbms_output.put_line('employee working as a
                      Analysts');
loop
fetch c1 into i;
exit when c1%notfound;
dbms_output.put_line(i.ename);
```

```
end loop;
close c1;
end;
/
```

NB:- Before we are reopening cursor then 1st we mus close Properly otherwise oracle server returns an error : ORA-6511 : cursor already open.

NB when we are not opening the cursor ~~and~~ but we are try to perform cursor operation then oracle server returns an error :

ORA-01001 : invalid cursor.

Parameter cursor :-

ex:-
```
declare
cursor c1(P-deptno number) is select * from emp
where deptno = P_deptno;
begin
for i in c1(10)
loop
dbms-output.Put-line(i.ename || ' ' || i.deptno);
end loop;
end;
/
```

W.A. PL/SQL cursor Program to display the emp as clark, analysts of emp table based on ~~Parameter~~ Job is a Parameter and also display that reports Practicly useing cursor for loop.

```
declare
cursor c1(P_Job varchar2) is select * from emp
where Job = P-Job;
begin
dbms-output.Put-line('employees working as clark');
```

```
for i in c1('clerk').
loop
dbms_output.put_line(i.ename);
end loop;
dbms-output.put_line('employees working as analysts');
for i in c1('ANALYST')
 loop
 dbms_output.put_line(i.ename);
end loop;
end;
/
```

NB:- in oracle when we are defining no. of cursors in PL/SQL block and also one cursor values pass in to another cursor then receiving cursor must be an parameterised cursor.

Q: W.A. PL/SQL Program to retrive all dept no from dept table useing a static cursor and also passed these deptnos from this cursor in to another parameterised cursor which returns employee details from emp table based on passed dept no.

A

```
declare
 cursor c1 is select deptno from dept;
 cursor c2(p_deptno number) is select * from
  emp where deptno = p_deptno;
  begin
  for i in c1
  loop
  dbms_output.put_line('My deptno is' || ' ' || i.deptno)
  for j in c2(i.deptno)
  loop
  dbms_output.put_line(j.ename || ' ' || j.sal || ' ' ||
                                        j.deptno);
```

```
end loop;
end loop;
end;
/
```

NB :- we can also Pass default values in Parameterised cursor useing default (or) := . becoz by default cursor Parameter are inparameters.

syntax :-

Parametername datatype default [:=] actualvalue

```
declare
cursor c1 (P_deptno number default 20) is select *
from emp where deptno = P_deptno;
i emp%rowtype;
begin
open c1();
loop
fetch c1 into i;
exit when c1% notfound;
dbms_output.Put_line (i.ename|| ' ' || i.deptno);
end loop;
close c1;
end;
/
```

Maa

Rd-15 13/06/14

%rowcount :-

This attribute always returns no. datatype i.e it returns no. of records number fetched from cursors.

ex:-

```
declare
   cursor c1 is selected ename, sal from emp;
   v-ename varchar2(10);
   v-sal number(10);
begin
   open c1;
   fetch c1 into v-ename, v-sal;
   dbms-output.put-line(v-ename || ' ' || v-sal);
   fetch c1 into v-ename, v-sal;
   dbms-output.put-line(v-ename || ' ' || v-sal);
   dbms-output.put-line('no. of records fetched
                  from the cursor' || ' ' || c1%rowcount);
   close c1;
end;
/
```

o/p:- SMITH 1500
ALLEN 1200
number of records fetched from the cursor 2.

Note:-

Cursors also used to transfer data from one oracle table in to another oracle table.

Q. W.A PL/SQL cursor program to transfer employees who are getting more then 2000 salary in emp table.

Pr ~~declare~~

```
sqly create table target (sno number (10), name
        varchar2 (10), salary number(10));

sqly declare
        cursor c1 is select ename, sal from emp
        where sal > 2000;
        v_ename varchar2 (10);
        v_sal number (10);
        n number(10);
    begin
        open c1;
            loop
                fetch c1 into v_ename, v_sal;
                exit when c1% notfound;
                n := c1% rowcount;
                insert into target( sno, name, salary)
                  values (n, v_ename, v_sal);
            end loop;
        close c1;
    end;
    /
```

O/P :—

sno	name	salary
1	SMITH	3000
2	JONES	3050
3	-	-
4	-	-

% isopen :—

This attribut always return boolean value either true or false.

Ex:-

```
declare
    cursor c1 is select * from emp where
    sal>2000 ;
    i emp%rowtype;
begin
    if not c1%isopen then
      open c1;
    end if;
    loop
      fetch c1 into i;
      exit when c1%notfound;
      dbms_output.put_line (i.ename|| ' '|| i.sal);
    end loop;
    close c1;
  end;
```

O/P:-

%found :-

This attribute also returns boolean value either true or false.

This attribute always returns true if data is available with in a cursor we are featching at least one row from the cursor.

Q) W.A. PL/SQL cursor Program to display all employee names & salary from emp table useing %found attribute.

A)
```
declare
   cursor c1 is select * from emp;
   t emp%rowtype;
begin
   open c1;
   fetch c1 into i;
   while (c1%found)
   loop
     dbms_output.put_line(v_ename || ' ' || v_sal);
   fetch c1 into i;
   end loop;
   close c1;
end;
/
```

O/P :-

~~7237~~	ename	sal
	SMITH	8000
	JOENS	5000
	⋮	⋮

Q) W.A. PL/SQL cursor Program to display total salary from emp table without useing ~~some~~ sum() function

A)
```
declare
    cursor c1 is select sal from emp;
    ~~t emp%rowtype;~~
    v-ename varchar2 (10);
    v-sal number(10):=10;
~~open~~
  begin
     open c1;
     loop
       fetch c1 into v-sal;
       exit when c1%notfound;
       n:=n+v-sal; / n:= n+nvl(v-sal,0);
       end loop;
       dbms-Output. Put-line('total salary is
            '||'||'n);
        close c1;
    end;
    /
```

O/P :- total salary is 33728.

Q) ~~declare~~
~~cursor c1 is select so~~

Maa

Rd-17 14/06/14

Eliminating Explicit cursor Life cycle (or) cursor for loop :-

↳ when we are useing cursors for loop internally Oracle server only open the cursor fetched records from the cursor & closed the cursors automatically.

↳ This is also called <u>shortcut method</u> of the cursor.

cursor for loop syntax :-

```
for indexvariablename in cursorname
loop
statements;
end loop;
```

↳ This loop used in executable section of the PL/SQL blocks.

<u>Note</u> :-

* in cursor for loops index variable internally behavibes like a record type variable (%rowtype).

ex:-

```
declare
   cursor c1 is select * from emp;
begin
   for i in c1
   Loop
    dbms_output.put_line(i.ename || ' ' || i.sal);
   end loop;
end;
```

Note:
we can also eliminate declare section of the using cursor for loop. in this case we must specify cursor select stmts in place of cursor name within cursorforloop.

syntax:

```
for indexvarname in (select statement)
loop
  stmts;
end loop;
```

ex:-

```
begin
  for i in (select * from emp)
  loop
   dbms_output.put_line(i.ename|| ' '|| i.sal);
  end loop;
  end;
  /
```

ex:-

```
declare
   cursor c1 is select * from emp;
begin
  for i in c1
  loop
   if i.sal > 2000 then
   dbms_output.put_line(i.ename || ' '|| i.sal);
   else
   dbms_output.put_line(i.ename || ' '|| 'low salary');
   end if
  end loop;
  end;
  /
```

ex:-

```
declare
   cursor c1 is select * from emp;
   number (10) := 0;
begin
   for i in c1
   loop
     n := n + nvl (i.sal , 0);
   end loop;
   dbms_output.put_line ('total salary is
                                '||' '||n);
   end;
   /
```

o/p :- total salary is 34075

Parameterised cursor ~~constructor~~ :-

we can also pass parameters into cursor same like a subprograms in parameter.

↳ These type of cursors are also called as Parameterised cursors.

↳ In Parameterised cursors we are defining formal parameters when we are defining the cursor & also pass the actual parameters when we are opening cursor.

Note :- In Oracle when ever we are defining formal parameters in cursors, procedures, function then we are not allowed to use datatype size in formal parameter declaration.

Syntax :-

formal Parameter

```
1) cursor cursorname (Parametername datatype)
   is select * from tablename
   where columnname = Parametername

2) open cursorname (actual Parameters);
```

ex:-

```
declare
   cursor c1 (P_deptno number) is select * from emp
    where deptno=P_deptno;
   i emp%rowtype;
 begin
    open c1(10);
    Loop
    fetch c1 into i;
    exit when c1%notfound;
    dbms_output.put_line(i.ename|| ' ' || i.sal || ' ' ||
                                || i. deptno);

    end loop;
    close c1;
 end;
 /
```

Q) W.A. PL/SQL Parameterised cursor Program for passing Job as a Parameter from emp table display the employee working as clerks and analysts and display o/p statically.

ex:- employee working as clerks

SMITH
ADAMS
JAMES
miller.

employee working as analysts
SCOTT
FORD.

```
declare
  cursor c1 (p_job varchar2)
  is select * from emp where job=p_job;
  i emp%rowtype;
begin
  open c1 ('CLERK');
  dbms_output.put_line('employees working as clerks')
  loop
    fetch c1 into i;
    exit when c1%notfound;
    dbms_output.put_line (i.ename);
  end loop;
  close c1;
    open c1('ANALYST');
    dbms_output.put_line('employees working as
                                analysts');
    loop
      fetch c1 into i;
      exit when c1notfound;
      dbms_output.put_line (i.ename);
  end loop;
  close c1;
end;
/
```

Note1:-
Before we are reopen the cursor in must close the cursor properly otherwise oracle server return an error ORA-06511: cursor already open.

Note 2:-

In oracle when we are not open in the cursor but we are try to perform operations on the functions then oracle server returns an error ora-1001: invalid space cursor.

Mar (Rd-19) 16/06/14

↳ In oracle we can also pass default values in parameterised cursor useing default or operator.

syntax:-

```
Parametername datatype default[:=] actual value;
```

ex:-

```
declare
    cursor c1 (p_deptno number default 30) is
    select * from emp where deptno = p_deptno;
    t emp%rowtype;
begin
    open c1();
    loop
      fetch c1 into t;
      exit when c1%notfound;
      dbms_output.put_line(t.ename||' '||t.sal||'
                                t.deptno);
    end loop;
    close c1;
end;
/
```

ex:-

```
declare
    cursor c1(p_deptno number) is select * from
    emp where deptno = p_deptno;
begin
    for t in c1(10);
    loop
      dbms_output.put_line(t.ename||' '||t.sal||
                                '||t.deptno);
    end loop;
end;
/
```

ex:-

```
declare
  cursor c1(P_Job varchar2) is select * from emp
  where Job=P_Job;
begin
  dbms_output.put_line('employees working as clerks')
  for i in c1('CLERK')
  loop
    dbms_output.put_line(i.ename);
  end loop;
  dbms_output.put_line('employees working as
                        analysts');
  for i in c1('ANALYSTS')
  loop
    dbms_output.put_line(i.ename);
  end loop;
end;
/
```

★ Note:-
In all database systems when we are passing values from one cursor into another cursor then receiving cursor must be a parameterised cursor.

Q ex:- W.A PL/SQL Program useing a static cursor retrive all department nos from dept table & pass these department nos in to another parameterised cursors to retrive employee details from emp table based on pass deptno.

A

```
declare
  cursor c1 is select deptno from dept;
  cursor c2(P_deptno number) is select * from
  emp where deptno = P_deptno;
```

```
begin
  for r in c1
  loop
   dbms_output.put_line('my dept table deptno
          is '||' '|| r.deptno);
  for j in c2(r.deptno)
   loop
    dbms_output.put_line (j.ename||' '|| j.sal
                          ||' '|| j.deptno);
  end loop;
 end loop;
end;
/
```

Update, delete statements used in cursor (without useing for update, where cursor current of clauses):-

Q) W.A PL/SQL cursor Program modifying salaries of the employee in employee table based on following condition.

1) if Job = 'CLERK' then increament sal → 100

2) if Job = 'SALESMAN' then decreament sal → 200

A) declare

```
  cursor c1 is select * from emp;
  r emp%rowtype;
 begin
  open c1;
  loop
  fetch c1 into r;
   exit when c1%notfound;
```

```
if i.job='CLERK' then update emp set sal=
        i.sal+100 where empno=i.empno;
elsif i.job='SALESMAN' then update emp set
        sal=i.sal-200 where empno=i.empno;
  end if;
end loop;
~~end c1~~;
close c1;
end;
/
```

Implicit cursor attributes :-

In oracle when PL/SQL blocks contain select...into clause & DML stmts or Pure DML stmts then oracle server automatically creates a memory area, This memory is also called as sqlArea (or) context Area (or) Implicit cursor.

- This memory area returns single record. when PL/SQL block contain select....into clause.
- This memory area also return multiple records when PL/SQL blocks contain pure DML stmts but these multiple records are processed at a time by using SQL engine.
- Alonge with SQL area oracle server automatically creates 4 memory location these memory locations are behaviour like a variable and also these variables are identifies by thorough implic cursor attributes.

These are :—

* sql%notfound.
* sql%found.
* sql%isopen.
* sql%rowcount.

* Always sql%isopen returns false. and also sql%notfound & sql%found attributes returns either true or false boolean values & also sql%-rowcount attribute returns number datatype.

Ex:-

```
begin
  delete from emp where ename='welcome';
  if sql%found then
    dbms_output.put_line('u r record deleted');
  end if;
  if sql%notfound then
    dbms_output.put_line('ur record does not
                          exists');
  end if;
end;
/
```

o/p:- ur record does not exists.

Ex:-

```
begin
  update emp set sal = sal + 100
  where job='MANAGER';
  dbms_output.put_line('affected number of
      managers are '||sql%rowcount);
end;
/
```

affected number of managers are 3

Exceptions :-

17/06/14

Exception is an error at runtime. whenever RE occure use an appropriate exception name is exception handler under Exception section.

Oracle having 3 types of exception.

- ★ Predefined Exception
- ★★ userdefined Exception
- ★★★ unnamed Exception.

★ Predefined Exception :-

- Oracle Provided 20 Predefined exception names for regularly occured runtime errors.
- when ever there runtime error occurs use an appropriate Predefined exception name is exception handler under exception section with in PL/SQL blocks.

syntax:-

```
WHEN PredefinedExceptionName1 then
    stmts;
WHEN PredefinedExceptionName2 then
    stmts;
. . . . . .
. . . . . .
WHEN others then
    stmts;
```

Some Predefined exceptionNames

1) no_data_found
2) Too_many_rows
3) zero_divide
4) invalid_cursor
5) cursor_already_open
** 6) invalid_number
* 7) value_error
: ... ...
: ... ...

1) no_data_found

when ever PL/SQL block carrying select ... into clause and also if requested data not available in a table then oracle server returns an error.

ORA-1403: no data found

To handle this error oracle provided 'no_data_found' exceptionName.

e.g:-

```
declare
    v_ename    varchar2(10);
    v_sal      number(10);
begin
    select ename, sal into v_ename, v_sal
     from emp where empno = &no;
     dbms_output.put_line(v_ename|| ' '||v_sal);
exception
    when no_data_found then
    dbms_output.put_line(' your employee
     does not exist');
end;
/
```

O/P :- Enter value for no : 111
your employee does not exit.

2) too-many-rows :-

when ever select... into clause try to return more than one value at a time or more number of rows, then oracle server returns an error.

ORA-1422: exact fetch returns more than requested number of rows.

for handling this error oracle Provided 'too-many-rows' exceptionName.

ex:-
```
declare
    v-sal Number(10);
begin
    select sal into v-sal from emp;
    dbms_output. Put-line (v-sal);
exception
    when too-many-rows then
    dbms_output. Put-line ('not to return more-rows');
end;
/
```

3) zero-divide :- 20/06/14

In oracle when we are try to Perform division with zero, then oracle server returns an error.

ORA-1476: divisor is equal to zero.

To handle this error oracle Provided zero-divide Exception Name

ex:-
```
begin
    dbms_output. Putline (8/0);
exception
    when zero-divide then
```

```
dbms_output.put_line ('not to perform division with
                        zero');
end;
/
```

4) invalid_cursor :-

when we are not opening the cursor but we are try to perform operations on cursor the oracle return an error, ORA-1001: invalid cursor

To handle this error oracle provided invalid_cursor exception name.

ex:-
```
declare
    cursor c1 is select * from emp
    where sal>2000;
    r emp%rowtype;
begin
    loop
        fetch c1 into r;
        exit when c1%rowfound;
        dbms_output.put_line (r.ename || ' ' || r.sal);
    endloop;
    close c1;
exception
    when invalid_cursor then
    dbms_output.put_line ('first you must open
                        the cursor');
end;
/
```

5) Cursor_already_open :-

Before reopening the cursor we must close the cursor properly otherwise oracle server return an error, ORA-6511: cursor already open

To handle this error oracle provided 'cursor_already_open' exception name.

ex:-
```
declare
    cursor c1 is select * from emp
    where sal>2000;
    i emp%rowtype;
begin
    open c1;
    loop
    fetch c1 into i;
    exit when c1%notfound;
    dbms_output.put_line(i.ename||' '||i.sal);
    end loop;
    open c1;
exception
    when cursor_already_open then dbms_output.
    put_line('we must close the cursor before
                reopen');
end;
/
```

6) invalid_number, value_error :-

in oracle when we are try to convert a string type to number type or data string into data type then oracle server return two types of errors these error,

invalid number, value error.

* invalid_number:-

when a PL/SQL block contains SQL stmts and also those stmts try to convert string type to number type or data string into Date type then oracle server returns an error, ORA-01722 : invalid number.

To handle this error oracle provided invalid_number exception name.

ex:-
```
begin
    insert into emp(empno, ename, sal) values
                    (1, soma, 'abc');
exception
    when invalid_number then
    dbms_output.put_line('insert proper data only')
end;
/
```

value_error :-

when a PL/SQL block contains procedure stmts and also these stmts are try to convert string type to number type then oracle server returns an error,

ORA-6502; numeric or value error : character to number conversion error.

to handle this error we are using value_error exception name.

ex:-
```
declare
    z number(10);
begin
    z:= 'sx' + 'sy';
    dbms_output.put_line(z);
exception
    when value_error then
    dbms_output.put_line('Enter proper data only');
end;
/
```

Note

In oracle whe we are try to store more data then the Datatype size in varchar2 datatype in variable declaration then also oracle server returns an error,

ORA-6502 : Number or value error : character string buffer too small

To handle this error also we are using value-error exceptionName.

```
declare
    z varchar2(3);
begin
  z := 'abcd';
  dbms_output.put_line(z);
exception
    when value_error then
    dbms_output.put_line('invalid string length');
end;
/
```

Note :-

when we are typing to store more data then of data-type size is ~~number~~ NUMBER datatype declared in variable declaration, then oracle server returns an error,

ORA-6502 : numeric or value error : number ~~Precis~~ Precision too large

to handle this error also we using value-error exception Name.

MQ?

Rd-26 Dt-21/06/14

```
declare
   ~~cursor~~
   z varchar2(3) := 'abcd';
begin
   dbms_output.put_line(z);
exception
   when value_error then
   dbms_output.put_line('invalid string length');
end;
/
```

error : ORA-06502 : numeric or value.
error : characters strings buffer too small.

Exception Propagation :-

In PL/SQL exception also ~~based~~ raised in declare section, executable section, exception section.

- when exception are raised in executable section those exception are handle either in inner blocks or outer blocks.
- when exception are raised in declare section or in exception section those exceptions must be handle with outer blocks this is called PL/SQL exception Propagation.

example :-

```
ex:- begin
declare
   z varchar2(3) := 'abcd';
begin
   dbms_output.put_line(z);
end;
```

```
exception
   when value_error then
   dbms_output.put_line('invalid string length
        handled through outer blocks only');
end;
/
```

O/P :-

invalid string length handled through outer blocks only.

Userdefined exception :-

In Oracle we also create our own exception and also raised when ever necessary these type of exception are also called as user defined exception.

Step1 : declare

step 2 : raise

step 3 : handling exception.

Step1 :- declare

In declare section of the PL/SQL blocks we are creating our own exception name using EXCEPTION Predefined type.

syntax :-

```
userdefinedname   exception;
```

Ex

~~Step 1:~~ declare

a exception;

Step 2:- raise

useing raise statement, we can raise user-defined exception explicitly either in executable section or in exception section of the PL/SQL block.

Syntax:-

```
raise  userdefinedexceptionname;
```

Step 3: handeling exception

We can also handle user defined exception as same as predefined exception useing exception handler.

Syntax:-

```
when  userdefinedexceptionname1  then
      stmts;
when  userdefinedexceptionname2  then
      stmts;
- - - - - - - - - - - - - -
- - - - - - - - - - - - - -
when  others  then
      stmts;
```

Q) WA PL/SQL Program raise userdefined exception of SATURDAY?

A)
```
declare
  a exception;
begin
  if to-char(sysdate,'DY')='SAT' then
   raise a;
  end if;
exception
   when a the
dbms-output.put-line('my exeeption raised
        on saturday');
end;
```

O/P:-
My exception raised on saturday.

ex:-
```
declare
  v-sal number(10);
  a exeption;
begin
   select sal into v-sal from emp
   where empno=7839;
if v-sal>3000 then
 raise a;
else
   update emp set sal=sal+100 where
   empno=7839;
end if;
exception
```

```
when a then
dbms_output.put_line ('salary already high')
end;
```

o/p:- salary already high.

ex:-

```
declare
  cursor c1 is select * from emp
  where job ='abcd';
  t emp%rowtype;
  a exception;
begin
  open c1;
  fetch c1 into t;
  if (c1%rowcount =0 then
  raise a;
  end if;
  close c1;
exception
  when a then
  dbms_output.put_line('the requested job
     not available in my table');
end;
/
```

o/p:- the requested job not available in my table;

Nested exception :—

```
declare
   z1 exception;
   z2 exception;
   begin
      begin
         raise z1;
          exception
          when z1 then
          dbms_output.put_line('z1 is handled');
      raise z2;
      end;
      exception
       when z2 then
        dbms_output.put_line ('z2 is handle');
       end;
      end;
       /
```

O/P:- z1 handle
z2 handle
PL/SQL Procedure successfully completed

Mqa

Rd-27 23/06/14

3) Unnamed Exceptions :—

In oracle if we want to handle, other then oracle 20 Predefined exceptions errors then we are useing unnamed method.

↳ In this method we are ~~fix~~ creating our own exception name and associates this exception name with appropriate error no. useing Exception_init()

Syntax :—

```
Pragma Exception_init(userdefined exception name,
                      errornumber);
```

↳ This function used in declared section of the PL/SQL blocks.

Note :—

Here Pragma is a compiler directive.

↳ when ever we are useing this Pragma oracle server associate error no. with Exception name at compile time only.

eg :—

```
declare
   a exception;
   Pragma exception_init(a, -1400);
begin
   insert into emp (empno, ename) values (null, 'soma');
exception
   when a then
   dbms_output.put_line('We cannot insert null
                         values');
end;
/
```

O/P :- we can not insert null values.

ex:-

```
declare
   a exception;
   Pragma exception_init(a, -2292);
begin
   delete from dept where
   deptno = 10;
   exception
   when a then
   dbms_output.put_line ('we cannot delete
                          master records');
end;
/
```

O/P:-

Q) W.A PL/SQL Program to handle -2291 error no. using exception_init() from emp, dept tables.

A)

```
declare
   a exception;
   Pragma exception_init(a, -2291);
begin
   insert into emp(empno, ename, deptno) values
                  (1,'abc', 50);
exception
   when a then
   dbms_output.put_line('not to insert other
                         then Primary key values');
end;
/
```

O/P:- not to insert other then Primary key values.

Error trapping functions :—

in Oracle if we want to catch error no.s; error nos with error messages then we are useing 2 Predefined error trapping functions.

These are :—

* sql code
* sqlerrm

* sql code & * sqlerrm

it returns numbers where as sqlerrm returns error numbers with error messages.

↳ These two functions are used in 'where', 'others then', 'clause' or used in our own 'exception handler'.

Note:— * if we want to catch error numbers at runtime then we are useing sql code function.

Note:—

sql code always returns numbers.

sql code returns values		Meaning
0	→	no oracle errors.
-ve	→	oracle errors
100	→	no-data-found
1	→	user defined exception

Naa

24/06/14

Note:-

In oracle we can also Predefined, userdefined function in INSERT stmts. But we are not allow to Pass SQL code, sqlerrm function directly in INSERT stmts.

To overcome this Problem we must create variable and use these variables in insert stmts.

```
SQL> create table target (Errno number(10), Errmsg
                                  varchar2(200));

SQL> declare
       v_sal number(10);
       v_errno number(10);
       v_errmsg varchar2(200);
       begin
         select sal into v_sal from emp;
       exception
         when others then
         v_errno := sqlcode;
         v_errmsg := sqlerrm;
       insert into largest values(v_errno, v_errmsg);
       end;
       /

SQL> select * from target;
```

Note:-

we can also handle unnamed exception using sql code function.

say begin
delete from dept where deptno = 10;
exception
when others then
if sqlcode = -2292 then
dbms_output.put_line ('not to delete master record');
end if;
/

Raise_application_error() :-

- In oracle, if you want to display undefined exception message in more descriptive from then only we are using raise_application_error().

function:

- This function is used in either in executable section or in exception section of the PL/SQL block.

- when we are using this function oracle server automatically display user defined exception messages in oracle error display format.

- This function accepts two parameters.

Syntax:-

raise_application_error (error number, message);
(error number → (-20,000 to -20,999); message → upto 512 char)

say declare
a exception;
begin

```
if to-char (SYSDATE, 'DY') = 'TUE' then
raise a;
end if;
exception
when a then
raise_application_error(-20123, 'myexception raised
                                  on tuesday');
end;
/
```

O/P :- ORA:- 20123 : My exception raised on tuesday.

<u>Note</u>:-

In oracle this function also used in triggers because this function stop invalid data entry based on the condition when condition is true, where as dbms_output. ~~Put_line~~ package does not stop invalid data entry into table if condition is true also.

SUBPROGRAMS

- Subprograms are named PL/SQL block which is used to solve particular task in a Program.
- All datatypes system having 2 types of subprograms
  1) Procedures (may or may not returns a value)
  2) Functions (Must returns a value)

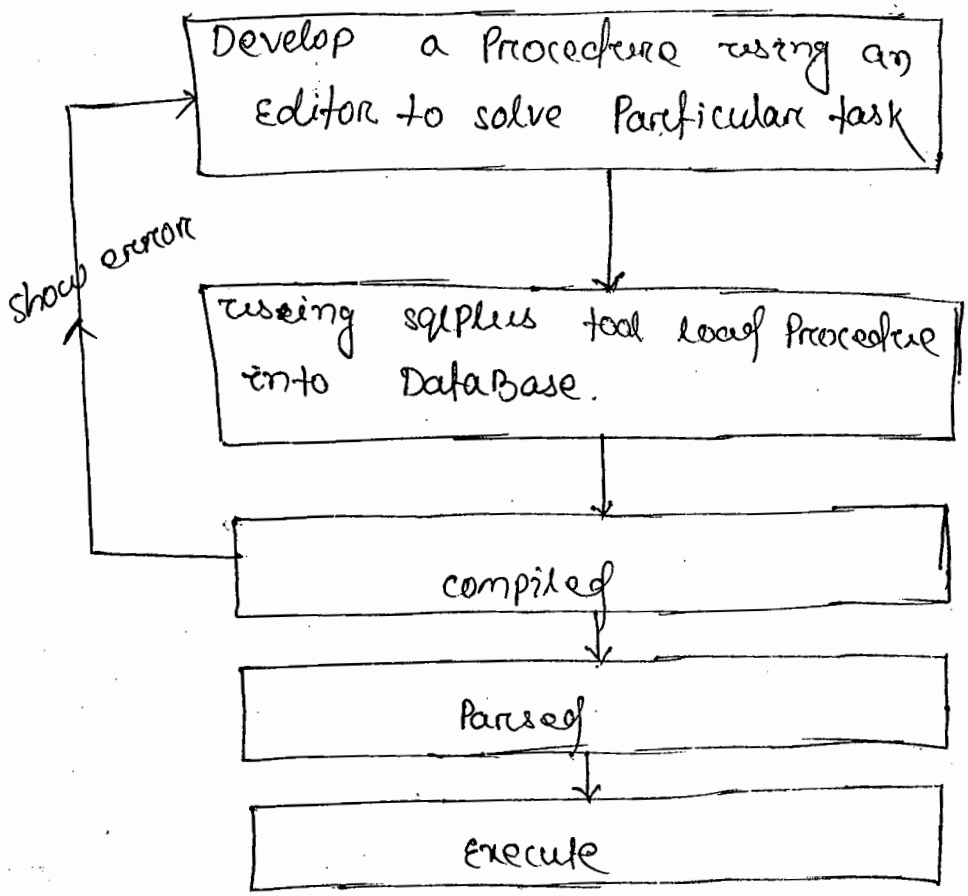
1) Procedures

- Procedure is a named PL/SQL block which is used to solve particular task Procedures may or maynot return values.
- In oracle whenever we are using create or replace keyword infront of the procedure, those Procedures are Parmanently stored in DB that's why that Procedures are also called as Stored Procedure.

Maa ?

Rd-29 25th/Jul/14

- Generally Procedures are used to improves Performance of the application becoz Procedure internally heaving one time compilation.
- In all DB one time compilation Program units improves ~~includes~~ Performance of the application.



Procedure hearing two Parts

1) Procedure specification
2) Procedure module

↳ In Procedure specification we are specifying name of the Procedure and type of the Parameters where as Procedure body we are solving actual task

syntax :-

```
Procedure specification
Create [or replace] Procedure Procedurename(formal Parameter)
IS/AS
Procedure body
Variable declerations, cursors,
   user defined exceptions,
Begin
   - -
[Exception]
[end [Procedurename]];
```

syntax of (formal Parameter) :-

```
Parametername [mode] datatype
                 in
                 out
                 in out
```

to view errors :-

sql> show errors;

executing a Procedure :-

Method 1 :-

syntax :-

```
sql> exec Procedurename (actual Parameters);
```

Method 2 :- (useing annonymous blocks):

syntax :-

```
sql> begin
     Procedurename (actual Parameters);
     end;
     /
```

Method 3 :-

syntax :-

```
sql> call Procedurename (actual Parameters);
```

Q> W.A. PL/SQL stored Procedure for Passing empno as Parameter then display name of the employee & his salary from emp table based on Passed empno.

A>

```
sql> create or replace Procedure
     P1 (P_empno number)
     is
```

```
v_ename varchar2(10);
v_sal number(10);
begin
  select ename, sal into v_ename, v_sal from emp
  where empno=p_empno;
  dbms_output.put_line(v_ename||' '||v_sal);
end;
/
```

execution :-

Method1 :-

```
sqy exec P1(7369);
  SMITH 8000
```

Method2 :- (useing annonymous blocks)

```
sqy begin
      P1(7566);
      end;
   JONES 2000
```

Method3 :-

```
sqy call P1(7902);
   FORD 3000
```

↳ In Oracle all Procedure information stored under user_procedures, user_source data dictionaries.

Note :-

★ if we want to view code of the Procedure then we are useing user_source data dictionary

ex:- desc user_source;

```
sqy select text from user_source
    where name='P1';
```

Q) W.A. PL/SQL stored Procedure for Passing deptno is a Parameter from emp table, then display employee details from emp table based on Passed dept no.

A)
```
create or replace Procedure
P2(P_dept number)
is
cursor c1 is select * from emp
where deptno = p_deptno;
i emp%rowtype;
begin
  open c1;
  loop
  fetch c1 into i;
  exit where c1%notfound;
  dbms_output.put_line(i.ename||' '||i.sal
                       ||' '||i.deptno);
  end loop;
  close c1;
end;
/
```

O/P :- SQL> exec P2(10);

CLARK 5000 10
KING 6600 10
MILLER 4100 10.

Naa 26/06/14

```
sql> CREATE OR REPLACE PROCEDURE P3(P_empno NUMBER, P_incr NUMBER)
     IS
     begin
     UPDATE emp SET
     sal = sal + P_incr
     WHERE empno = P_empno;
     end;
     /
```

sql> exec P3(7702, 200);

Procedure Parameters :—

- Parameter is a name which is used to pass values into Procedure & return values from Procedure.
- Procedure having 2 types of Parameters.
  - * Formal Parameters.
  - * Actual Parameters.

1) Formal Parameters :—

Formal Parameters must be specified in Procedure specification.

- Formal Parameters specifyes name of the Parameter, mode of the Parameter, type of the Parameter.

Note:- In Oracle we are not allow to use data type size in formal Parameter declaration.
when we are declaring cursors, Procedures, functions.

Syntax:-

ParameterName [Mode] datatype

Mode :-

- Mode specifies Purpose of the Parameter.
- Oracle formal Parameter having 3 types of mode.

1) In mode.
2) out mode.
3) In out mode.

1) In mode :-

- In oracle by default Parameter mode is 'in' mode.
- This mode is used to passed the value into Procedure body.
- This Parameter behavibes like a constant in Procedure body.

Note :- we can also pass default values into Procedure using 'in' Parameter in this case we are using default (or) := operator.

syntax :-

```
ParameterName   in datatype default [:=] actualvalue
```

Q) write a PL/SQL stored Procedure to insert a record into dept table using 'in' Parameter?

A)

```
CREATE OR REPLACE PROCEDURE
P1 (P_DeptNo in NUMBER, P_Dname in VARCHAR2, P_LOC in
                                          VARCHAR2)
is
begin
    INSERT INTO DEPT
    VALUES( P_DEPTNO , P_DNAME , P_LOC);
    dbms_output.Put_Line ('your record inserted through
                                  Procedure');
end;
/
```

execution

```
sql> exec P1 (5, 'x', 'y');
```

2) Out Mode :-

'out' mode is used to return values from the Procedure. 'out' Parameter behaves like a reinitialize variable in a Procedure body.

Here explicitly we must specify 'out' keyword.

syntax:-

Parametername out datatype

e.g :-

```
CREATE OR REPLACE PROCEDURE
P1 (a in NUMBER, b OUT NUMBER)
IS
begin
   b:= a * a;
end;
/
```

In oracle when a subprogram having out or in out Parameters, those subprograms are executed using following methods.

Method 1 :- using bind variable.

Method 2 :- using annonymous block.

Method 1

```
sql> variable z NUMBER;
sql> exec P1 (4, :z);
sql> Print z;
```

o/p :- z

16

Method 2

```
sql> declare
       x NUMBER (10);
     begin
       P1 (9, x);
     dbms_output.put_line(x);
     end;
/
```

81

Q) write a PL/SQL stored Procedure for Passing empname as 'in' Parameter from Emp table and return salary of the employee using 'out' Parameter.

A)
```
CREATE OR REPLACE Procedure
  P1 (P-Ename in varchar2, P-sal out NUMBER)
IS
begin
  SELECT sal into P-sal
  FROM Emp
  where ename = TO-UPPER (P-Ename);
end;
```

execution :-

Method-1

say variable a Num.
say exec P1('SMITH' :a);
say Print a;

Method-2

say declare
  x NUMBER(10);
begin
  P1('MILLER' ,x);
  dbms-output-Put-line (x);
end;
/

Mag

27/06/14

Q) write a PL/SQL stored Procedure for Passing dept no as in Parameter and return dname, location using out Parameter from dept no.

A) create or replace Procedure
  P1(P-deptno in number, P-dname out varchar2,
    P-loc out varchar2)
  IS

```
begin
   select dname, loc into
   p-dname, p-loc from Dept.
   where deptno = p-deptno;
end;
/
```

execution

using bind variable :-

```
sql> variable a varchar2(10);
sql> variable b varchar2(10);
sql> exec P1(10,:a,:b);
sql> Print a b;
```

O/P :- A

ACCOUNTING

B

NEW YORK

Q) W.A. PL/SQL stored Procedure using in, OUT. Parameter convert following group by clause.

```
select deptno, count(*) from
Emp
GROUP by deptno;
```

Deptno	count(*)
30	6
20	5
10	3

```
sqy create or replace Procedure
    P1(p_deptno in number, p_count out number)
is
begin
  select count(*) into p_count
  from emp
  where deptno = p_deptno;
end;
/
```

execution

```
sqy variable a Number;
sqy exec (10, :a);
sqy Print a;
```

o/p :- A
3

Pass by value, Pass by reference

when ever we are using modular Programming, all languages supports two types of Passing Parameter mechanism;

1) Pass by value
2) Pass by reference.

These two mechanism specifyes when we are modifying formal Parameter then actual Parameters are affected or not.

→ In Pass by value method actual value does'nt change in main Program, when we are modifying formal Parameter also becacuse internally copy of the values are passed into ~~calling~~ calling Program.

If we want to change actual value according to formal Parameter modification then we are using Passby reference method.

Nocopy :-

Oracle also support these two passing Parameter mechanism when we are using subprogram Parameter In oracle bydefault all in Parameters internally uses pass by reference method. where as bydefault all out Parameter internaly - uses Pass by value method.

↳ In oracle when ever we are returning large amount of data using out Parameter internally copies of the values are created because out Parameter internally uses pass by value method. this Process automatically degrades Performance of the Procedure. To overcome this Problem oracle introduced nocopy hint in out Parameter.

Syntax:-

Parameter out nocopy datatype

Ex:-

```
SQL> create or replace procedure P1( P_name in
     varchar2 , P_sal out nocopy number)
     is
     begin
      select sal into P_sal from emp where ename=
                                          P_ename ;
     end;
     /
```

iii) in out mode :-

↳ This mode is used to pass & return values from the procedure.

↳ This mode behaves like a constant initialized variable in procedure body.

Syntax:-

Parametername in out datatype

Ex- create or replace procedure p1(a in out number)
is
begin
a := a * a;
end;
/

Execution

method1: (bind variable)
SQL> variable z number
SQL> exec p1(:z);
SQL> Print z;

o/p:- Z
81

method2: (using anonymous block)
SQL> declare
x number(10) := &x;
begin
p1(x);
dbms_output.put_line(x);
end;
/

o/p:- enter value for x: 3
9

28.06.14

Q) W.A. PL/SQL Program to store Procedure for picking employee number as Parameter, return salary & returning value from emp table using bind variable.

```
create or replace Procedure P1(Rx in not number)
is
begin
select sal into Rx from emp where empno = Rx;
end;
```

<u>execution</u> :- (using bind variable) :

```
SQL> variable z number;
SQL> exec :z := 7902;
SQL> exec P1(:z);
SQL> Print z;
        z
      3850
```

<u>Automomous concept / transaction</u> :-

Automomous transactions are independent transactions used in Procedures, triggers.

- Generally Automomous transactions are used in child Procedures
- when ever we are calling automomous Procedures in PL/SQL block & also when we are using Rollback in PL/SQL block then automomous Procedures transactions never affected.

↳ If we want to make a Procedure autonomous then we are using:

autonomous_transaction Pragma, commit

Syntax:-

Pragma autonomous_transaction;

↳ This Pragma must be defined in declared section of the Procedure or declare section of the trigger.

Syntax:-

```
create or replace Procedure Procedurename
(formal parameter)
is/as
Pragma autonomous_transaction;
begin
...
commit;
[exception]
...
end [Procedure_name];
```

Ex:-

SQL> create table test(name varchar2(10));

using autonomous transactions:-

```
SQL> create or replace Procedure P,
     is
     Pragma autonomous_transaction;
     begin
```

```
    insert into test values ('india');
    commit;
    end;
    /
SQL> begin
    insert into test values ('hybd');
    insert into test values ('mumbai');
    P1;
    rollback;
    end;
    /

SQL> Select * from test;
    Name
    India.

SQL> delete from test
    // 1 row deleted
SQL> Select * from test
    // no row selected.
```

without using autonomous transactions :-

```
SQL> Create or replace Procedure P1
    is
    begin
    insert into test values ('India');
    commit;
    end;
    /
    Procedure created.
```

```
SQL> begin
     insert into test values ('hyd')
     insert into test values ('mumbai');
     P1;
     rollback;
     end;
     /
SQL> select * from test;

  Name
  ----
  hyd
  mumbai
  India.
```

In oracle, when a Procedure having commit & also when we are calling this Procedure ~~this Procedure~~ in another PL/SQL block then this Procedure commit not only some.

↳ This Procedure transactions but also saves all the above Procedure transactions.

↳ To overcome this Problem oracle size(8.1.6) introduced autonomous transaction is Procedures.

authid current_user :-

↳ Data Procedure having authid current_user clause then that Procedure not executed by another user if you are giving Privilages also.

↳ Generally, when we are reading data from table & Perform some on DML operations then only data security Point of view, we are using this clause

in Procedures.

↳ This clause used in specification of the Procedure.

syntax:-

```
create or replace Procedure Procedure_name
(formal Parameter)
authid current_user
is/as
: :
begin
: :
end;
```

giving Procedure (Privilagation) to another users:-

syntax:-

grant execute on Procedurename to username;

```
SQL> conn scott/tiger;
SQL> Create or replace Procedure P1(P_empno number)
     authid current_user
     is
     V_ename varchar2(10);
     V_sal number(10);
     begin
     select ename, sal into V_ename, V_sal from emp
     where empno = P_empno;
     dbms_output.put_line(V_ename || ' ' ||V_sal);
     end;
SQL> grant execute on P1 to murali;
SQL> con murali/murali
     set server output on;
SQL> exec scott.P1(7902);
     error: table or view does not exists.
```

Mar 7

(35) 30/06/14

Handled (or) un handled exceptions in Procedures:-

when ever we are calling inner Procedure in to outer Procedure, then we must implement inner Procedure exception otherwise Oracle server executes outer Procedure default exception handler.

example of Inner Procedure :-

```
sql> create or replace Procedure
     P1 (a in number, b in number)
     is
     begin
       dbms_output.put_line(a/b);
       exception
       when zero_divide then
       dbms_output.put_line('b can not be zero');
     end;
     /
```

example of outer Procedure :-

```
sql> create or replace Procedure
     P2
     is
     begin
     P1(5,0);
     exception
     when others then
     dbms_output.put_line('any error');
     end;
     /
```

SQL> exec P2;
b can not be zero;

Execution Methods :-

IN Parameter :-

↳ Procedure IN Parameter haveing 3 types of execution method.

1) Positional ~~Method~~ Notations. (ex:- exec P_1(1, 'a', 'b');)
2) Named notations. (ex:- ⇒) SQL> exec P_1(P_dname ⇒ 'x', P_Loc ⇒ 'y', P_deptno ⇒ 2);
3) Mixed notations :-

↳ it is a ~~nomin~~ combination of Positional cum named notations.

Note :-
After Positional their can be all named notations but after named their can not be positional.

SQL> exec
P_1(3, P_Loc ⇒ 'y', P_dname ⇒ 'x');

↳ we can also drop Procedure using drop Procedure ~~name~~ Procedurename.

~~function~~.

FUNCTIONS

- function is a named PL/SQL Blocks which is used to solve Particular tasks & also functions must return a value.
- Function also heaving two Parts.
  1) function specification.
  2) function body.
- In function specification we are specifying name of the function & type of the Parameter where as the function body, we are start solving actual tasks.
- Syntax:-

```
create or replace Function. Functionname(
                                formal Parameters)
return datatype
 is/As
variable declarations, cursors, user defined exceptions;
begin
 ==  } body
 ==
Return expression;
[Exception]
  ==
end [Functionname];
```

Executing a function :-

Method 1 (using select statement) :-

when a function does not have Parameters or when a function heaving IN Parameters then those type of function ~~thei~~ they are executing using select stmts.

↳ syntax:-

```
Select functionname( actual Parameters) from dual;
```

Method 2 (Using annonymous blocks) :-

syntax:-

```
sqlp begin
    variablename := functionname(actual Parameters);
    end;
```

ex:-

```
create or replace function f1(a varchar2)
return varchar2
is
begin
return a;
end;
/
```

execution :-

Method 1 :- (Using select statement) :-

```
sqlp select f1 ('hi') from dual;
hi
```

Method 2 :- (using annonymous block) :-

~~sqlp~~

```
SQL> declare
     x varchar2(10);
     begin
     x := f1('welcome');
     dbms_output.put_line(x);
     end;
/
```

op:- welcome.

Q) W.A. PL/SQL stored function for passing number as a parameter, then return a message either even or odd based on passed number.

A)

```
Create or replace function f1(a number)
return ~~number~~ varchar2
IS

Begin
  If mod (a,2) = 0 then
     return 'even';
  else
     return 'odd';
  end if
end;
/
```

Mea Execution:- 01/07/14

Method 1 (using select stmts):-

```
SQL> select f1(5) from dual;
odd
```

Method 2 (using ennonymous block):-

```
SQL> declare
     z varchar2(10);
```

```
begin
z := f1(4);
dbms_output.Put_line(z);
end;
/
```

o/p :- even.

Method 3 (Using bind variable) :-

```
SQL> variable x varchar2(10);
SQL> begin
     :x := f1(8);
     end;
     /
SQL> Print x;
     x
```

o/p :- even.

Method 4 :-

```
SQL> exec dbms_output.Put_line(f1(7));
```

o/p :- odd

~~Method~~

Method 5 :-

```
SQL> begin
     dbms_output.Put_line(f1(8));
     end;
     /
```

o/p :- even

Note :-

we can also call user-defined functions into insert stmts.

```
sqy create table test (msg varchar2(10));
sqy insert into test values(f1(6));
sqy Select * from test;
```

o/p:- msg
even.

DML Stmts used in functions :~

Generally when a function heaving DML Stmts those functions are not executed using Select stmts, but we are executing those function using annonymous blocks.

↳ if we want to execute those select stmts then we must use ~~at~~ autonomous transactions with in functions.

ex:-

```
sqy create or replace function
    f1(p_empno number)
    return number
    is
    v_count number(10);
    begin
    delete from emp where
    empno = p_empno;
    v_count := sql%rowcount;
    return v_count;
    end;
/
```

execution :- (Select stmts)

```
sqy Select f1(1) from dual;
```

error: cannot perform a DML operation inside a query.

Method2 (using annonymous block(s)) :-

```
sql> declare
        a number(10);
        begin
        a := f1(1);
        dbms_output.put_line(a);
        end;
        /
```

o/p :- 0

Using autonomous transactions :-

Ex :-

```
sql> create or replace function
        f1(p_empno number)
        return number
        is
        v_count number(10);
        Pragma autonomous_transactions;
        begin
        delete from emp where empno=p_empno;
        v_count := sql%rowcount;
        commit;
        return v_count;
        end;
        /
```

execution :-

```
sql> select f1(1) from dual;
```

o/p :- 0

Note :-

In Oracle we can also use Predefined aggregate functions into user defined functions and also use this user_defined functions in same table.

or in diff table. These user-defined functions also used ~~in~~ along with non-aggrigate functions.

ex:-

```
create or replace function f1
return number
is
v_sal number(10);
begin
select max(sal) into v_sal from emp;
return v_sal;
end;
/
```

executions:-

```
Select f1 from dual;
6600
(or)
say select ename, sal, f1 from emp;
```

Q) W.A. PL/SQL stored function for passing empno, date as parameter, return number of years employees working using emp table based on passed parameter.

A)

```
create or replace function f1 (P_empno number,
                               P_date date)
return number
is
z number(10);
begin
select month_between
(P_date, hiredate)/12 into z from emp
where empno = P_empno;
return round(z);
end;
```

execution :-

```
select empno, ename, hiredate, f1(empno, sysdate)
|| ' ' || 'years' "exper" from emp where empno=
7566
/
```

O/P:-

Empno	ename	hiredate	exper
7566	JONES	02-APR-81	33 years.

Note :-

We can also use named, mixed notations in Oracle 11g. when a subprog executed using select stmts.

oracle 11g :-

```
select empno, enama, hiredate, f1 (p_empno => empno,
    p_date => sysdate) || ' ' || 'years' "exper" from
emp where empno = 7566;
```

Mar 7

(37) 02/07/14 :

Q) W.A PL/SQL stored function for passing empno as a parameter then return job of the emp from emp table based on passed parameter.

A)

```
create or replace function f1 (p_ename ~~number~~ varchar2)
return ~~number~~ varchar2
is job
v_empsal ~~number~~ varchar2(10)
begin
  select job into v_job from emp where
    ename = p_ename;
  return v_job;
end;
/
```

execution :-

sql> select f1('SMITH') from dual;

o/p :- CLERK.

Q) W.A. PL/SQL stored function for passing empno as a parameter from emp table, calculate tax of the emp based on the following conditions.

1) if annsal > 10000 then
   tax → 10% of annsal
2) if annsal > 15,000 then
   tax → 20% of annsal
3) if annsal > 20,000 then
   tax → 30% of annsal
   else
   tax → 0

A)
```
create or replace function tax(p_empno number)
return number
is
v_sal number(10);
annsal number(10);
z number(10);
begin
select sal into v_sal from emp where
empno = p_empno;
annsal = v_sal * 12;
if annsal > 10,000 and annsal <= 15,000 then
z := annsal * 0.1;
else if
annsal > 15000 and annsal <= 20,000 then
z := annsal * 0.2;
```

```
        elseif annsal > 25000 then
            z := annsal * 0.3;
        else
            z := 0;
    end if;
    return z;
end;
/
```

execution :-

```
SQL> select tax(7902) from dual;
```

o/p :- 14580

Out Mode :-

This mode is used to return mode value from the functions, but when ever function heaving out parameters in out parameter those functions are not allowed to execute using select stmts.

Q. W.A. PL/SQL stored function for passing deptno as a parameter then return dname, location using out parameters from dept table.

A

```
create or replace function f1(p_deptno number,
        p_dname out varchar2, p_loc out varchar2)
return varchar2
is
begin
  select dname, loc into p_dname, p_loc from
  dept where deptno = p_deptno;
  return p_dname;
end;
/
```

execution :— (using bind variable)

```
sqlp variable a varchar2(10);
sqlp variable b varchar2(10);
sqlp variable c varchar2(10);
sqlp begin
      :a:=f1(10, :b, :c);
    end;
    /
sqlp Print b c;
  B
  ----------
  ACCOUNTING
  C
  ----------
  NEW YORK
```

CURSORS used in functions :-

In Oracle we can also used cursor within user-defined functions.

- Generally in Oracle, if we want to develop user-defined aggrigate functions same like a Predefind aggrigate function and also those aggrigate fun returns multiple values then only we are using cursors with in user-defind functions.
- we can also used these user-defined aggrigate function same like a Pre-defined aggrigate fun in group by clause.

Ex:- 4

```
create or replace function f1(p_deptno number)
return varchar2
is
cursor c1 is select ename from emp where
deptno=p_deptno;
z varchar2(200);
begin
 for i in c1
 loop
  z:=z || ',' || i.ename;
 end loop;
 return z;
 end;
  /
```

execution :-

say select deptno, f1(deptno) from emp group by deptno;

Note :-

Oracle 10g introduced wm_concat() Predefined aggregate function which returns multiple values groupwise.

We can also pass all datatype columns into this function.

Ex SQL> select job, wm_concat(ename) from emp group by job.
/

SQL> select deptno, wm_concat(hiredate) emp
group by deptno.

↳ In oracle, all stored function information are stored under user_procedures, user_source dictionaries.

↳ If you want to view code of the function, then we are using user_source data dictionaries.

Ex

SQL> desc user_source
SQL> select text from user_source where name='F1';
/

↳ We can also drop function by using:

drop function function_name;

TRIGGERS

Mail (39) 03/07/14

↳ Triggers is also same as stored Procedure & it will automatically ~~invoke~~ invoked when ever DML ~~operation~~ Performed on table or view.

↳ All database system heaving two types of triggers:

1) statement level triggers.
2) row level triggers.

↳ In statement level trigger, trigger body executed only once per DML stmts.

↳ where as a row level triggers, triggers body executed for each row for DML stmts.

↳ Triggers also heaving two Parts

1) triggers specification.

2) triggers body.

Syntax:— Trigger Timing → Trigger events

```
create or replace Trigger Triggermame
before / after insert / update / Delete on tablename
[for each row] → for row level Triggers.
[when condition] →
[Declare]
  variable declation, cursors, user defined exceptions
Begin
  ===
end;
```

Trigger specification.

Trigger body

Difference betn stmts level, row level trigger:-

Stmts level trigger:-

say create or replace Trigger th1
after update on emp
begin
dbms_output.Put_Line('stmts level');
end;
/

testing:-

say update emp set sal=sal+100
where deptno=10;

statement level.

3 row updated.

```
sql> drop trigger tl1;
```

row level trigger :-

```
sql> create or replace trigger tl2
     after update on emp
     for each row
     begin
     dbms_output.put_line('row level');
     end;
     /
```

testing :-

```
sql> update emp set sal=sal+100
     where deptno=10;
     row level
     row level
     row level
     3 rows updated.
```

ROW LEVEL TRIGGERS :-

In row level triggers triggers body is executed for each row DML ~~is executed~~ statements thats why we are using for each row clause in triggers specifications. and also DML transaction values are stored in two rollback segment qualifiers.

These are old, new. These qualifiers are used in either trigger specification or in trigger body, when we are using these

qualifier in trigger body then we must use ~~colum~~ column infront of the qualifier name.

syntax:-

```
:old.colname
```

syntax:-

```
:new.columnname
```

operation /	insert	update	Delete
:new	✓	✓	X
:old	X	✓	✓

Q) W.A. PL/SQL row level trigger on emp table when ever user insert in salary, then salary should be more then 5000.

A)
```
create or replace trigger t13
before insert on emp
for each row
begin
if :new.sal <5000 then
raise_application_error(-20345,'salary should
                                be above 5000');
end if;
end;
/
```

testing:-

sql> insert into emp (empno, sal) values (1, 3000);

o/p:- ORA-20345 : salary should be above 5000.

say insert into emp(empno, sal) values (1, 8000);
1 row created.

Q) W.A. PL/SQL row level trigger using emp, dept table impliment on delete cascate concept with out using <u>on delete cascate</u> clause.

A)
```
create or replace trigger t1y
after delete on dept
for each row
begin
delete from emp where deptno = :old.deptno;
end;
/
```

<u>testing</u> :-

say delete from dept where deptno=10;
say select * from dept;
say select * from emp;

Q) W.A. PL/SQL row level trigger, when ever user modifying salary on emp table, display old salary, new salary automatically.

A)
```
create or replace trigger t15
before update on emp
for each row
begin
dbms_output.put_line('old salary is '||' '||:old.sal);
dbms_output.put_line('new salary is '||' '||:new.sal);
end;
/
```

Testing

SQL> update emp set sal = sal + 100
where ename = 'SMITH';
old salary is 1500;
new salary is 1600;

Q> W.A. PL/SQL row level trigger on dept table when ever user modifying deptnos, then automatically those modification are effected in emp table.

A> create or replace trigger t16
after update on dept
for each row
begin
update emp set deptno := new.deptno where
deptno = :old.deptno;
end;
/

testing :-
SQL> update dept set deptno = 1
where deptno = 10;
SQL> select * from dept;
SQL> select * from emp;

Mar 7

Row level trigger applications :- (41) 04/07/14

Q> W.A. PL/SQL Row level trigger on emp table, whenever user entered job as a analyst & also entered comission of the ANAYST, then automatically user entered commision set to null.

A)
```
create or replace trigger tk1
before insert on emp
for each row
when (new.Job='ANALYST')
begin
if :new.comm is not null then
:new.comm:=null;
end if;
end;
/
```

testing :-

```
SQL> insert into emp(empno, Job, comm) values
     (1, 'ANALYST', 2000);
SQL> select * from emp;
```

Q) w.A. PL/SQL trigger on emp table, when ever user inserting data into employee no column, that column doesn't accepts duplicate values. i.e it acts like a Primary key behaviours.

A)
```
Create or replace trigger tk2
before insert on emp
for each row
declare
a number(10);
begin
Select count(*) in to a from emp where
 empno:=new.empno;
```

```
    if a>=1 then
raise_application_error(-20123, 'we cannot insert
                        duplicate values');
else if a=0 then
 dbms_output.put_line('1 record inserted');
end if;
end;
/
```

testing :-

say insert into emp(empno) values(7902);

ORA-20125 : we can not insert duplicate values.

Q) W.A. PL/SQL row level trigger on emp table each department has atmost 4 employees from emp table.

A)
```
Create or replace trigger tk5
before insert or update of deptno on emp
emp for each row
declare
 a number(10);
begin
select count(*) into a from emp where
 deptno = :new.deptno group by deptno;
 if a>=4 then
  raise_application_error(-20567, 'too many
                  employees in my deptno');
 end if;
end;
/
```

testing :-

SQL> insert into emp (empno, deptno) values (1,10);
1 row created.

SQL> insert into emp(empno, deptno) values (2,10);
ORA-20567 : too many employees in my deptno.

↳ auditing a column :-

when ever we have modifying data in a column those transaction values are stored in another table is called auditing a column.

↳ auditing a column app must be implimented using rowlevel triggers.

Ex:-

```
SQL> create table target (oldempno number (10),
                          oldsalary number(10),
                          newsalary number (10),
                          cdy date, ~~user name~~
                          username varchar2(10));
SQL> create or replace trigger tk6 before update
     of sal on emp ~~for each row~~
     for each row
     begin
      insert into target (oldempno, oldsalary,
        newsalary, cdy, username) values (
      :old.empno, :old.sal, :new.sal, sysdate, user);
     end;
     /
```

testing:-

SQL> update emp set sal=sal+100 where deptno=10;

SQL> select * from target

↳ In oracle we are not allowed to use :new, :old qualifiers infront of the sysdate, user pseudo columns.

Maa ? auto increament :- (43) 05/07/14

If we want to generates Primary key values automatically, then all database system uses autoincreament concept.

↳ In Oracle we are implementing autoincreament Concept ~~to~~ using ~~only with~~ row level triggers Sequences. i.e we are creating a sequence in a sql & use that sequence in PL/SQL row level triggers.

Ex:-

~~SQL> create or replace trigger~~

```
SQL> create table test(sno number(10) Primary key,
                       name varchar2(10));

SQL> create sequence s1
     start with 1;
SQL> create or replace trigger tr1
       before insert on test
       for each row
       begin
        select s1.nextval into :new.sno from dual;
       end;
/
```

testing:-

```
SQL> insert into test(name) values('&name');
     Enter value for name: Soma
SQL> /
     Enter value for name: Hippo
SQL> Select * from test;
```

sno	name
1	soma
2	Hippo

* Note:-

Oracle 11g introduce variable assignment Concept when we are using sequences in PL/SQL block, in this case we are not allowed to use dual table & also not allowed to use Select... into clause.

syntax:-

```
begin
variablename := sequencename.nextval;
end;
```

Ex:- Oracle 11g:-

```
SQL> create or replace trigger tr1
     before insert on test
     for each row
     begin
     :new.sno := s1.nextval;
     end;
     /
```

```
SQL> create or replace trigger tr1
     after insert on test
(can't change so error occured) for each row
     begin
        select s1.nextval into :new.sno from dual;
     end;
     /
error: cannot change new value for this
       trigger type.
```

Trigger Timing :— (before/after)

When we are using before timing DML transaction values are not effected directly DB. before this Process these values are effected in either in trigger specification or trigger body then only those value are effected in DB. Thats why in oracle when ever we are modifing :new qualifier values in row level trigger then we must use before timing otherwise Oracle Server returns an

```
error : can not change new values for
        this trigger type
```

↳ In Oracle if we want to restrict invalid data entries also, then we are using before timing.

↳ when we are using after timing DML transaction values are directly effected in DB then only these values are effected trigger.

↳ Generally, if we want to Process data one table column values based on other table or if we want to transfore data one table into another table then only we are using <u>after timing</u>.

Q) W.A. PL/SQL row level trigger on emp table when ever user inserting data into ename column, then automatically user inserted data converted into upper case. emp table...

A)
```
create or replace trigger tr1
before insert on emp
for each row
begin
:new.ename := upper(:new.ename);
end;
/
```

<u>testing</u>:-

```
SQL> insert into emp(empno, ename) values(1, 'soma');
SQL> select * from emp;

     empno  ename
       1    SOMA
```

Max ?

08/07/14

ex:-

```
SQL> create or replace trigger tr1
     before insert or update or delete on emp.
     begin
       if sysdate = last_date (sys date)
     then
       raise_application_error(-20123, 'we can not
                                Perform dmls in last day
                                of the month');
     end if;
   end;
```

testing :-

SQL> delete from emp where empno=7566;

ORA - 20123 : we cannot perform dmls in last day of the month.

Triggering events (OR) trigger Predicate clauses :-

when we are using no. of tables in trigger body & also if we are try to Perform no. of operations in those tables, when we are using triggering events with a trigger body. These are :-

inserting, updating & deleteing clauses.

↳ These events are used in either in stmts levels triggers or in row level triggers.

syntax:—

```
if inserting then
 stmts;
elseif updating then
 stmts;
elseif deleting then
 stmts;
end if.
```

Q) W.A. PL/SQL stmts level trigger on emp table not to perform DMLs operations in any days.

A) create or replace trigger tr2
before insert or update or
delete on emp
begin
if inserting then
raise_application_error(-20123, 'we can not
perform insertion');
elseif updating then
raise_application_error(-20567, 'we can not
perform updation');
elseif deleting then
raise_application_error(-20749, we can
not perform deletation');
end if;
end;

testing :-

```
SQL> delete from emp where empno = 7902;
     ORA-20749: we can not perform deletion.
```

ex:-

```
SQL> create table test (msg varchar2(20));
SQL> create or replace trigger tr3
       after insert or update or delete on emp
       declare
         z varchar2(20);
       begin
         if inserting then
           z := 'rows inserted';
         elseif updating then
           z := 'rows updated';
         elseif deleting then
           z := 'rows deleted';
         end if;
       ~~end;~~ insert into test values (z);
       end;
```

testing :--

```
SQL> insert into emp( empno) values (1);
     // 1 row created.
SQL> insert into emp(empno) values (2);
     // 1 row created.
SQL> update emp set sal = sal + 100 where deptno=20;
     // 5 row updated.
SQL> delete from emp where empno in(1,2);
     // 2 row deleted.
```

SQL> select * from test;

MSG
.....
rows inserted
rows inserted
rows updated
rows deleted

Q) W.A. PL/SQL row level trigger on emp table. whenever user inserting data, on emp table then those values are automatically stored in another table, when ever user modifing data those transactional data automaticaly stored in another table, when ever user deleting data those deleted data also stored in another table.

A) SQL> create table target1 (empno number(10),
name varchar2(10),
salary number(10));

SQL> create table target2 (empno number(10),
name varchar2(10),
salary number(10));

SQL> create table target3 (empno number(10),
name varchar2(10),
salary number(10));

SQL> Create or replace trigger tJ5
after insert or update or delete on emp
for each row
begin
if inserting then

```
insert into target1(empno, name, salary) values
        (:new.empno, :new.ename, :new.sal);
elsif updating then
insert into target2(empno, name, salary) values
        (:new.empno, :old.ename, :old.sal);
elsif deleting then
insert into target3(empno, name, salary) values
        (:old.empno, :old.ename, :old.sal);
end if;
end;
/
```

testing:-

```
SQL> delete from emp where sal > 4000;
SQL> select * from target3;
  // 5 row deleted.
```

Trigger Execution Order:-

1) before statement level.
2) before row level.
3) after row level. ★ imp
4) after statement level.

ex:-

```
SQL> create table test(sno number(10));
SQL> create or replace trigger tru
        after insert on test
        for each row
        begin
```

```
    dbms_output.put_line('after row level');
  end;
  /
SQL> create or replace trigger tu2
       before insert on test
       for each row
       begin
       dbms_output.put_line('before row level');
       end;
       /
SQL> create or replace trigger tu3
       after insert on test
       begin
       dbms_output.put_line('after statement
                                  level');
       end;
       /
SQL> create or replace trigger tu4
       before insert on test
       begin
       dbms_output.put_line ('before statement
                                  level');
       end;
       /
```

testing:-

```
SQL> set serveroutput on;
SQL> insert into test values(8);
       before statement level.
       before row level.
       after row level.
       after statement level.
```

Mar ? follows clause (oracle 11g) :- 09/07/14

↳ In oracle when ever we are using same table on same level when we can not control execution out of the trigger to over come this Problem oracle 11g introduce follows clause.

↳ This clause used in trigger specification.

syntax:-

```
follows create or replace trigger triggername
before/after insert/delete/update on tablename
[for each row]
[when condition]
[follows anothertriggername]
[declare]
- - - -
begin
. . .
end;
```

↳ using follows clause expelicitly we are Providing gurantee execution order of the trigger.

ex:-

```
SQL> create table test(sno number(10));
SQL> create or replace trigger tb1
     after insert on test
     begin
```

```
dbms_output.put_line('trigger1 fired');
end;
/
```

```
SQL> create or replace trigger tb2
     after insert on test
     begin
     dbms_output.put_line('trigger2 fired');
     end;
     /
```

~~SQL>~~ Testing:-

```
SQL> set serveroutput on;
SQL> insert into test values (5);
     trigger2 fired.
     trigger1 fired.
```

using follows clause (oracle 11g):-

```
SQL> create or replace ~~(oracle 11g)~~ trigger tb1
     after insert on test
     begin
     dbms_output.put_line('trigger1 fired');
     end;
     /
```

```
SQL> create or replace trigger tb2
     after insert on test
     follows tb1
     begin
     dbms_output.put_line('trigger2 fired');
     end;
     /
```

Testing :—

SQL> set serveroutput on;

SQL> insert into test values (4);

trigger1 fired.

trigger2 fired.

(rowlevel or stmts level).

Q) w.A. PL/SQL trigger on emp table, when ever user deleting ~~data~~ records on emp table, then automatically display remaining no. of records no.s in bottom of the delete stmts.

A) // using stmts level trigger

```
create or replace trigger th1
    after delete on emp
    ~~for each row~~
    declare
    x number(10);
    begin
    select count(*) into x from emp;
    dbms_output.put_line(x);
    end;
/
```

Testing :-

SQL> delete from emp where empno=7369;

// 13.

// using row level trigger.

```
create or replace trigger th1
    after delete on emp
    for each row
    declare
    x number(10);
```

```
begin
  select count(*) into x from emp; ~~where~~
  dbms_output.put_line(x);
end;
/
```

// Trigger created.

↳ ~~The~~ In this Prog no compile time error, but it error in runtime.

Testing :-

```
SQL> delete from emp where empno = 7902;
ORA-04091 : table SCOTT.EMP is mutating.
```

↳ Mutating error :-

def^n:- If there row level trigger based on a ~~rowlevel~~ table then trigger body ~~come out~~ can not read data from same table ~~but~~ & also it can not perform DML operations on same table, if we are trying this then oracle server returns an error i.e

ORA-04091 : table is mutating. This error is called Mutating error. This table is called Mutating table & also This trigger is called as Mutating trigger.

↳ Mutating error is a run time error. occured in row level triggers.

↳ This error doesn't occured stmts level triggers.

↳ whenever we are using stmts level triggers those transactions are automatically comitted into DB. That's why using trigger body we can ~~use~~ used. commited data without any Problem. ~~whenever~~ that's why stmts level triggers doesn't have mutating error.

↳ whenever we are using row level triggers, DML transaction values are not commited automatically; use trigger body, if we are trying to read without commited data then all DB servers returns mutating errors.

↳ To overcome this Problem, we are using autonomous_transactions with in triggers, but autonomous_transactions always returns Previous result.

ex:-

```
create or replace trigger td2
    after delete on emp
    for each row
    declare
    x number(10);
    Pragma autonomous_transaction;
    begin
    select count(*) into x from emp;
    commit;
    dbms_output.put_line(x);
    end;
    /
```

Trigger Created.

Testing :-

SQL> delete from emp where empno=7902;

Mar 7

10/07/14

↳ when clause :-

we can also specify condition in trigger specification using when clause.

↳ when clause is not allowed to use in stmts level trigger.

↳ In when clause we are not allowed to use colon (:) in front of the qualifier name (new, old).

↳ when condition must be specified with in Parenthesis.

↳ when condition must be an SQL expression.

↳ when condition always returns values either true or false.

↳ when condition returns true then only trigger body is executed.

ex:-

```
SQL> Set serveroutput on;
SQL> create or replace trigger tr1
        after insert on emp
        for each row
        when(new.empno = 1)
        begin
        dbms_output.put_line('display message when
                                empno is 1');
        end;
        /
```

Testing:-

SQL> insert into emp(empno) values(1);

o/p:- display Message when empno is 1 only

// 1 row created.

SQL> insert into emp(empno) values(2);

// 1 row created.

Q) W.A. PL/SQL rowlevel trigger on employee table whenever user inserting a Job as CLERK then automatically stored today into another table & also used when clause.

A) SQL> create table test (col1 date);

SQL> create or replace trigger trg2

```
after insert on emp
for each row
when (new.Job = 'CLERK')
begin
insert into test(col1) values(sysdate);
end;
/
```

Testing:-

SQL> insert into emp(empno, Job) values(5, 'CLERK');

// 1 row created.

SQL> Select * from Test;

```
COL1
---
10-JUL-14
```

Autoincreament using varchar2 datatype :-

```
SQL> create table test(sno varchar2(40) Primary key,
                       name varchar2(40));
SQL> create ~~or replace~~ sequence s1 start with 1;
SQL> create or replace trigger tm1
     before insert on test
     for each row
     begin
     select 'ABC' || Lpad(s1.nextvalue, 10, '0') ~~into~~:
      into :new.sno from dual;
     end;
     /
```

Testing:-

```
SQL> insert into test(name) values('sname');
  Enter value for name: soma
SQL> /
  Enter value for name: Patel
SQL> select * from test;
```

SNO	Name
ABC0000000001	Soma
ABC0000000002	Patel

Calling a Procedure into trigger :-

we can also called Procedure into trigger using call statements.

Syntax:-

```
create or replace trigger triggername
before/after insert/update/delete on tablename
 call Procedurename.
```

ex:-

```
SQL> create table test (totsal number(10));
SQL> create or replace Procedure
     P1
     is
     v_sal number(10);
     begin
     delete from test;
     select sum(sal) into v_sal from emp;
     insert into test values (v_sal);
     end;
     /
// Procedure created.

SQL> create or replace trigger trn2
     after insert or update or delete on emp
     call P1.
```

Testing:-

```
SQL> update emp set sal = sal+100;
SQL> select * from test;
```

DDL Triggers or systems Triggers:-

In all DB we can also create triggers of DDL events. These types of triggers also called as DDL triggers or systems triggers.

↳ In oracle DB Administrator creating this trigger either in DB level or in schema level.

Syntax:-

Syntax:-

```
create or replace trigger Triggername
    before/after create/Alter/drop/truncate/Rename
                    on database/username.schema

    declare
      ...
    begin
      ...
    end;
```

ex:-

```
SQL> conn sys as sysdba;
     Enter Password: sys
SQL> create or replace trigger ty5
     after create on database
     begin
     dbms_output.put_line('somebody create database object');
     end;
SQL> set serveroutput on;
SQL> create table ty(sno number(10));
     Somebody create database object.
```

Q) w.A dd/L trigger on sport schema not to drop emp table.

A) create or replace trigger ty1.
before drop on scott.schema
begin
if ora_dict_obj_name = 'emp' and

```
ora_dict_obj_type = 'TABLE' then
raise_application_error(-20123, 'we cannot drop
                                emp table');
end if;
end;
/
SQL> drop table emp;
ORA-20123: we cannot drop emp table.
```

Mar 7

11/07/14

Syntax:-

```
alter trigger triggername
enable/disable;
```

enable/disable all triggers with in a tab:-

Syntax:-

```
enable/disable all triggers.
```

example:-

```
SQL> alter table emp
     disable all triggers;
```

- Oracle having 12 types of triggers based on stmts level, row level, before, after, insert, update & delete.
- Oracle also having DDL triggers.
- Oracle also supports triggers on views, through instead of triggers.
- All triggers information stored under user_triggers data dictionary.

```
SQL> desc user_triggers;
```

↳ we can also drop triggers.

drop trigger triggername;

—o—

~: PACKAGES :~

↳ Package is a DB objects. which encapsulates global variables, constants, Procedures, functions, types, cursors into single units.

↳ Generally Packages does't accepts Parameters & also can not be nested & also can not be invoked.

↳ Generally Packages are used to improves performance of the applications. becoz whenever we are calling Packaged subProgram 1st time then automatically total Package loaded into Memory area (RAM).

↳ whenever we are calling subsequent, subProgs then oracle server calling those subProgram from memory area.

↳ This Process automatically reduces ~~this~~ disk I/O.

↳ Every Package heaving two Parts

*) Package specification.

**) Package body.

↳ By default Package specification objects are Public. where as Package body objects are Private.

↳ Generally in Package specification we are

declaring objects where as Package body we are implementing ~~Pack~~ those objects.

<u>syntax</u>:—

<u>Package specifaction</u> :—

```
Create or replace Package Packagename.
  is/as
  global variable declarations, constant declarations;
  cursors declarations;
  Types declaration;
  Procedure declaration;
  function declaration;
end;
```

<u>Package Body</u> :—

```
create or replace Package body Packagename
  is/as
  Procedure implementations/definations;
  Functions implementations/definations;
end;
```

<u>Calling Packaged Procedures</u> :—

<u>Method 1</u> :—

SQL> exec Packagename. Procedurename(actual Parameters);

<u>Method 2</u>:— (using annonymous block):—

```
SQL> begin
       Packagename. Procedurename(actual Parameters);
     end;
```

Calling Packaged functions :-

Method 1 :- (using select stmts)

```
SQL> select Packagename.functionname(actual Parameters)
     from dual;
```

Method 2 :- (using annonymous block)

```
SQL> begin
     variablename := Packagename.functionname
                     (actual Parameters);
     end;
```

example :-

```
SQL> create or replace Package Pk1
       is
       Procedure P1;
       Procedure P2;
     end;
     /
SQL> create or replace Package body Pk1
       is
       Procedure P1
       is
       begin
       dbms_output.put_line('first Procedure');
       end P1;
        Procedure P2
          is
          begin
          dbms_output.put_line('second Procedure');
        end P2;
       end;
       /
```

// Package body created.

```
SQL> exec Pk1.P2;
  // Second Procedure.
```

Global Variables :—

In oracle global variable must be specified in Package specification only.

ex:—

```
SQL> create or replace Package Pk2
     is
     g number(10) := 1000;
     Procedure P1;
     function f1(a number) return number;
     end;
      /
SQL> create or replace Package body Pk2
     is
     Procedure P1
     is
     a number(10);
     begin
       a := g/2;
     dbms_output.put_line(a);
     end P1;
     function f1(a number) return number
     is
     begin
     return a * g;
     end f1;
     end;
      /
```

execution :—

```
SQL> exec Pk2.P1;
```

500

SQL> Select Pk2.f1(3) from dual;

300

Mag ? Pragma serially reusable:- 12/07/14

↳ To maintain state of global variable or state or global cursor then we are using serially reusable Pragma in Packages.

↳ Pragma serially-reusable;

* State of the global variable:-

```
SQL> Create or replace Package Pn1
     is
     Pragma serially-reusable;
     g number (10) := 4;
     end;
/
SQL> begin
     Pn1.g := 70;
     end;
/
SQL> begin
     dbms-output.Put-line(Pn1.g);
     end;
/
```

o/p:- 4

~~State~~ ~~of~~

State of the cursor:-

```
SQL> create or replace package Pf1
     cursor c1 is select * from emp;
     Pragma serially_reusable;
     end;
     /
SQL> begin
     open Pf1.c1;
     end;
     /
SQL> begin
     open Pf1.c1;
     end;
     /
```

Overloading Procedures :-

overloading referes to same name can be used for different purpose.

↳ In oracle, we can also implement overloading Procedure using Package these ~~Procedure~~ packages having same name with diff types or diff no. of Parameters.

Ex:-

```
SQL> create or replace package Pf2
     is
     Procedure P1(a number, b number);
     Procedure P1(x number, y number);
     end;
     /
SQL> create or replace Package body Pf2
     is
     Procedure P1(a number, b number)
     is
```

```
c number(10);
begin
c:=a+b;
dbms_output.put_line(c);
end P1;
Procedure P1 (x number, y number)
is
z number(10);
begin
c := a-b;
dbms_output.put_line(z);
end P1;
end;
```

Execution:-

```
SQL> exec Pf2.P1 (a=>9, b=>4);
  O/P:- 13

SQL> exec Pf2.P1 (x=>10, y=>6);
  O/P:- 4
```

forward declaration :-

declaring a Procedure in Package body is called forward declaration.

↳ Generally we can also call Private Procedure into Public Procedure. But in this case we must implement Private Procedure before calling otherwise use a forward declaration.

Ex:-

```
SQL> create or replace Package Pf3
     is
     Procedure P1;
     end;
     /
```

```
SQL> create or replace Package body Pf3
     is
     Procedure P2;
     Procedure P1;
     is
     begin
     P2;
     end P1;
     Procedure P2
      is
     begin
     dbms_output.Put_Line('Private Procedure');
     end P2;
     end;
     /
```

forward declaration.

```
SQL> exec Pf3.P1;
```

o/p :- Private Procedure

Types used in Packages :-

In Oracle we also create our own user-defined Datatype using type keyword.

In PL/SQL user defined ~~Proc~~ Types are created a Procedure/steps :-

1) 1st we are creating type using after/Print Syntax then only we are creating variable of DataType.

2) PL/SQL heaving following types :

- PL/SQL Record.
- ★ Index by table (OR) PL/SQL table (OR) Associative array.

composite datatype.

is also called as collection

is also called collection { • Nested table. • Varray. } → composite Datatypes.

★ • Ref cursor.

1) PL/SQL Records :—

This is a user-defined type which is used to represent diff datatypes in to single unit.

↳ It is also same as structured in 'C' language

↳ This is an user-defined type, so we are creating in 2 step Process.

1) 1st we are creating type then only we are creating a variable of that type.

syntax:- (of step1)

```
Type typename is record (attribute1,
                         datatype(size), --);
```

syntax:- (of step2)

```
variablename    typename;
```

ex:-

sql) declare

```
Type t1 is record (a1 number(10), a2 number(10),
                   a3 number(10));
v_t1 t1;
begin
v_t.a1 := 101;
v_t.a2 := 'soma';
v_t.a3 := 2000;
```

Maa 7 14/07/14

```
dbms_output.put_line(v_t.a1||' '||v_t.a2||' '||v_t.a3);
end;
/
```

O/P:- 101 SOMA 2000

Ex:-2

```
SQL> create or replace package pz1
     is
     type t1 is record (a1 number(10),
                        a2 number(10),
                        a3 number(10));
     procedure p1;
     end;
     /

SQL> create or replace package body pz1
     is
     procedure p1
     is
     v_t t1;
     begin
     select empno, ename, sal into v_t from emp
      where ename = 'SMITH';
     dbms_output.put_line(v_t.a1||' '||v_t.a2
                          ||' '||v_t.a3);
     end p1;
     end;
     /

SQL> set serveroutput on;
SQL> exec pz1.p1;
```

O/P:- 7369 SMITH 2500

*2) Index by table (or) PL/SQL table (or) associative array :-

This is an user-defined type which is used to stored no. of data items in a single unit.

- Basically this is an unbound table.
- Basically index by table heaving key-value Pairs. i.e here value field stored actual data and key field stored indexes.
- These indexes are either no.s or characters and also these indexes are either +ve or -ve no.s.
- This key internally behaviors like a Primary key.
- Generally Index by tables are used to improves performance of the application becoz these tables are stored in RAM memory area. thats why these tables are also called as Memory tables.
- For improveing Performance of the APP oracle Provided binary_integer data for key field.
- If we want to operate index by table then we are using following collection Methods. these are :

1) exists
2) first
3) last
4) Prior
5) next
6) count
7) delete(index).
8) delete(index1, indexn)
9) delete.

↳ This is a user_defined type so we are creat-ing a 2 step Process. i.e 1st we are creating type then only we are creating a variable of datatype.

<u>syntax</u> :— (of step1)

```
Type Typename is Table of datatype(size)
Index by binary_integer ;
```

<u>syntax</u> :— (of step2)

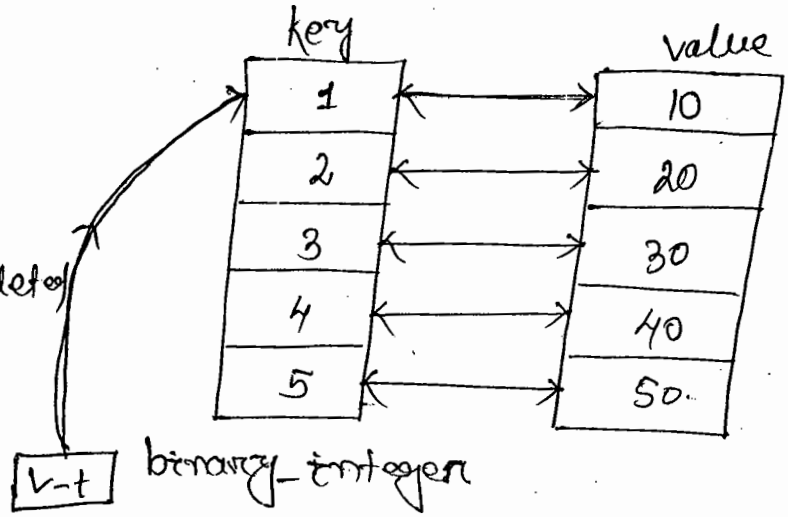
```
variablename Typename;
```

<u>ex</u>:—

```
sql> declare
     type t1 is table of number(10)
     index by binary_Integer;
     v_t t1;
     begin
     v_t (1) := 10 ;
     v_t (2) := 20;
     v_t (3) := 30 ;
     v_t (4) := 40;
     v_t (5) := 50;
     dbms_output.Put_line (v_t(1)) ;
          "        "        (v_t.first);
          "        "        (v_t.last);
          "        "        (v_t.Prior(3)) ;
          "        "        (v_t.next);
          "        "        (v_t.count) ;
          "        "        (v_t.delete(1,3)) ;
```

```
dbms_output.put_line(v_t.count);
end;
/
```

O/P :- 1
1
5
2
4
5
1,2,3 deleted
2



Q) W.A. PL/SQL Prog to ~~read~~ retrive all emp names from emp table & stored these emp names in index by table & also display content from index by table.

A) SQL>
```
declare
    type t1 is table of varchar2(10)
    index by binary_integer;
    v_t t1;
    cursor c1 is select ename from emp;
    n number(10) := 1;
    begin
    open c1;
    loop
    fetch c1 into v_t(n);
    exit when c1%notfound;
    n:= n+1;
    end loop;
    close c1;
```

```
for i in v_t.first..v_t.last
loop
dbms_output.put_line(v_t(i));
end loop;
end;
/
```

Mar 7

15/07/14

↳ whenever resource table having large amount of data & also if we are transferring data using cursors, then automatically oracle server degrades performance of the application. beroz cursors internally uses record by record process.

↳ To overcome this proble if we want to improve performace of the application oracle 8i introduce bulk collect clause.

syntax:-

```
select * bulk collect into collectionvarname
        from tablename where condition;
```

↳ when ever we are using bulk collect clause in oracle server select columns at a time & stored data into collections, This process automatically improves ~~Pre~~ performance of the application.

Ex:-

```
declare
type t1 is table of varchar2(10)
index by binary_integer;
v_t t1;
```

```
begin
select ename bulk collect into v_t from emp;
for i in v_t.first .. v_t.last
loop
dbms_output.put_line (v_t(i));
end loop;
end;
/
```

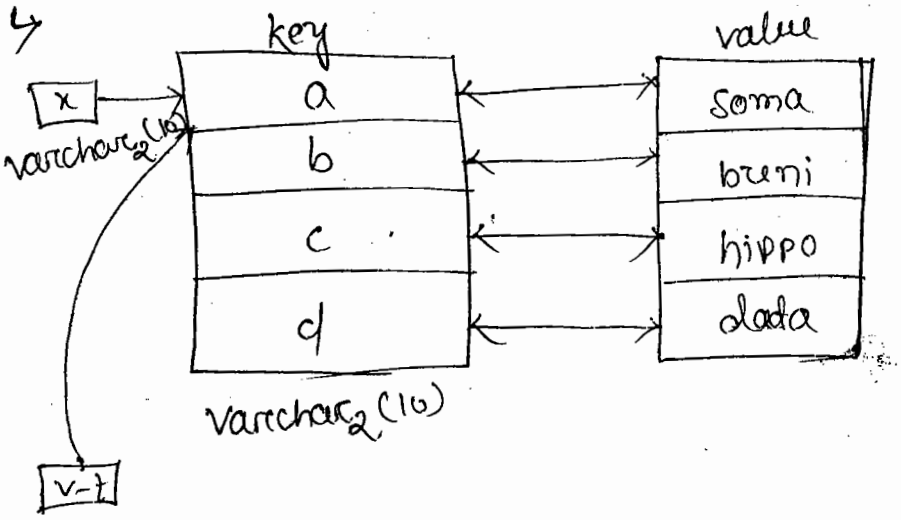
Q) W.A. PL/SQL Program to store next 10 dates in to index by table and also display content from index by table.

A)

```
declare
type t1 is table of date
index by binary_integer;
v_t t1;
begin
for in in 1 .. 10
loop
v_t(i) := sysdate + i;
end loop;
for i in v_t.first .. v_t.last
loop
dbms_output.put_line (v_t(i));
end loop;
end;
/
```

Q) W.A. PL/SQL Prog^n to transform all employees joining dates from emp table & store those dates in to index by table & display content from index by table.

```
declare
  type t1 is table of date
  index by binary_integer;
  v_t t1;
  begin
  select hiredate bulk collect into v_t from emp;
  for i in v_t.first..v_t.last
  loop
  dbms_output.put_line(v_t(i));
  end loop;
  end;
  /
```

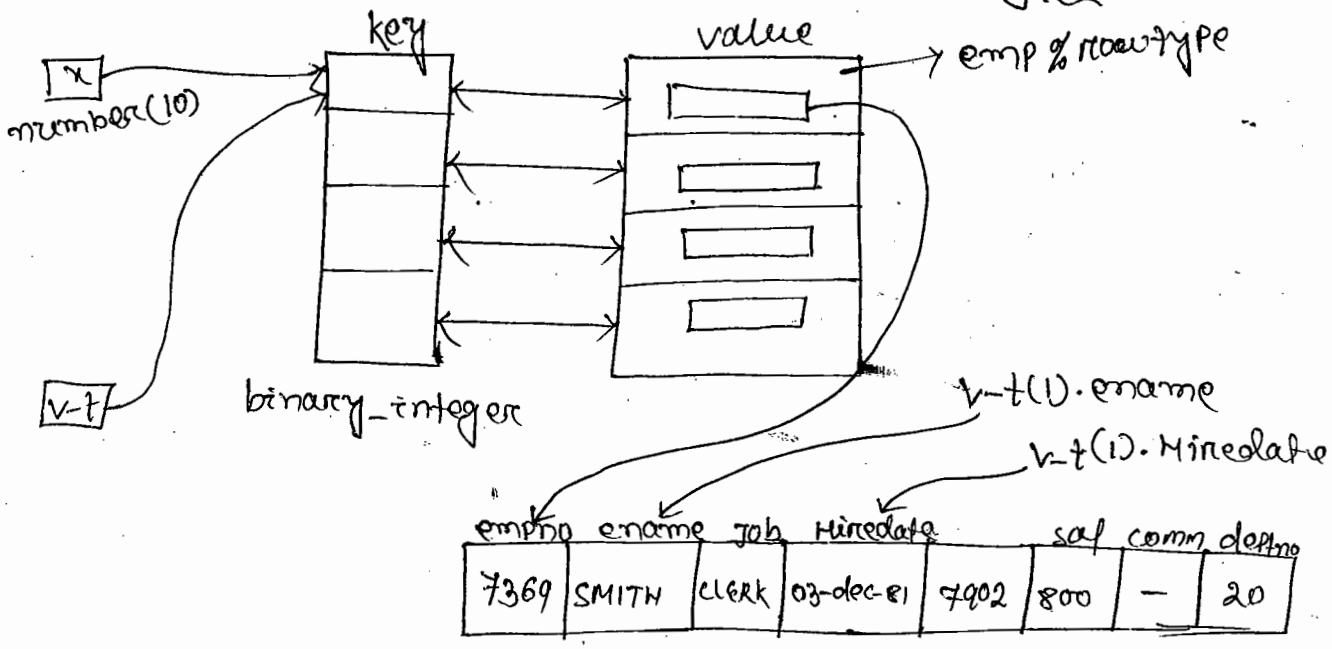


Ex:-

```
declare
  type t1 is table of varchar2(10)
  index by ~~binary~~ varchar2(10);
  v_t t1;
  x varchar2(10);
  begin
  v_t('a') := 'soma';
  v_t('b') := 'bruni';
  v_t('c') := 'hippo';
  v_t('d') := 'dada';
```

```
x := 'a';
loop
dbms_output.put_line (v_t(x));
x:=v_t.next(x);
exit when x is null;
end loop;
end;
/
```

Using value field as record type datatype (%rowtype) :-



Ex:-

```
declare
type t1 is table of emp%rowtype
index by binary_integer;
v_t t1;
x number (10);
begin
select * bulk collect intov_t from emp;
x:=1;
loop
```

```
dbms_output.put_line(v_t(x).ename || ' ' || v_t(x).sal
                    || ' ' || v_t(x).hiredate);
x := v_t.next(x);
exit when x is null;
end loop;
end;
/
```

(OR)

ex:-

```
declare
type t1 is table of emp%rowtype
index by binary_integer;
v_t t1;
begin
select * bulk collect into v_t from emp;
for i in v_t.first .. v_t.last
loop
dbms_output.put_line(v_t(i)).ename || ' ' || v_t(i).sal
                    || ' ' || v_t(i).deptno);
end loop;
end;
/
```

Mag

16/07/14

? Return Result Sets :-

In oracle if we want to return large amount of data from DB server into client application then we are using following two methods.

1) using index by table
2) using ref cursor.

if we want to develop this type of application then first we must develop DB server application using function, i.e through the function only we are returning bulk of data and then only execute these application using client application.

DB server application using index by table :-

```
SQL> create or replace package ph1
     is
     type t1 is table of emp%rowtype
     index by binary_integer;
     function f1 return t1;
     end;
     /
SQL> create or replace body ph1
     is
     function f1 return t1
     is
     v_t t1;
     begin
      select * bulk collect into v_t from emp;
     return v_t;
     end t1;
     end;
     /
```

execution using PL/SQL client :-

```
SQL> declare
       x ph1.x1;
       begin
         x:= ph1.f1;
         for i in x.first..x.last
         loop
```

```
dbms_output.put_line(x(i).ename || ' ' || x(i).sal
                                || ' ' || x(i).hiredate);
  end loop;
end;
/
```

'EXISTS' collection Method :-

- Exists collection method used in index by tables, Nested table, varray.
- Exists collection method always return boolean value either true or false.
- This collection method is used to test wheather requested data available in collection or not using indexes.

SQL> declare

```
        type t1 is table of Number(10)
        index by binary_integer;
        v_t t1;
        z boolean;
        begin
        v_t(1) := 10;
        v_t(2) := 20;
        v_t(3) := 30;
        v_t(4) := 40;
        v_t(5) := 50;
        dbms_output.put_line(v_t(1));
        z:= v_t.exists(3);
        if z = true then
        dbms_output.put_line( 'an index exists with
                        having an element' || v_t(3));
```

```
else
  dbms_output.put_line('index 3 does not exists');
end if;
  for i in v_t.first .. v_t.last
loop
  dbms_output.put_line(v_t(i));
end loop;
end;
/
```

Ex:-
```
declare
  type t1 is table of varchar2(10)
  index by binary_integer;
  v_t t1;
  begin
  select ename bulk collect into v_t from emp;
  v_t.delete(3);
  for i in v_t.first .. v_t.last
  loop
  dbms_output.put_line(v_t(i));
  end loop;
  end;
/
```

O/P:- SMITH
ALLEN
ORA-01403: No data found.

<u>Note:-</u>
whenever index by table or nested table having gaps and also if we are try to display collection data then oracle server returns an error.
'<u>ORA-01403: no data found</u>'

To overcome this Problem we must use EXISTS collection method.

sol^n:-

```
declare
  type t1 is table of varchar2(10)
  index by binary_integer;
  v_t t1;
begin
  select ename bulk collect into v_t from emp;
  v_t.delete(3);
  for i in v_t.first .. v_t.last
  loop
    if v_t.exists(i) then
    dbms_output.put_line(v_t(i));
    end if;
  end loop;
  end;
/
```

Mag

17/07/14

Nested table, varray:-

Oracle 8.0 ~~inc~~ introduced nested table, varray. These are also ~~s~~user-defined data type which is used to store number of data items in a single unit. Before we are storing data into these collection we must use constructor.

Here constructor name is same as typename.

* Nested table :-

- This is an user defined which is used to store number of data item in a single unit.
- This is also unconstraint table same like index by table, but this table doesn't have key value pair
- In this collection always index are start with 1 and also these index as conjucative.
- Generally index by table are not stored permanently in to DB and also we can't add or remove indexes.
- To overcome these problems oracle 8.0 introduced extension of index by table called nested table which is used to store permanently to DB using SQL and also we can add, remove indexes using <u>extend, trim</u> collection methods.
- This is an user defined type so we are creating in two step process, first we are creating type then only we are creating a variable of that type.

Syntax :-

1) Type TypeName is table of datatype (size);

2) variableName TypeName := TypeName();

constructor.

Mqa 7 ~~23/07/14~~

SQL> declare
```
    type t1 is table of number(10)
    index by binary_integer;
    v_t t1;
  begin
    v_t (500) := 80;
    dbms_output.put_line (v_t(500));
  end;
/
```

o/p:- 80

By default index by tables are basically sparse i.e we are not allowed to allocate memory explicitly. internally oracle server only automatically allocate the memory based on key, value pair.

SQL> declare
```
    type t1 is table of number(10);
    v_t t1 := t1();
  begin
    v_t (500) := 80;
    dbms_output.put_line (v_t (500));
  end;
/
```

error: subscript beyond count.

when we are using nested table we must reserved memory explicitly by using extend collection method.

```
SQL> declare
        type t1 is table of Number(10);
        v_t t1 := t1();
     begin
        v_t.extend(500);
        v_t(500) := 80;
        dbms_output.put_line(v_t(500));
     end;
     /
o/p:- 80
```

```
SQL> declare
        type t1 is table of number(10);
        v_t t1 := t1();
     begin
        v_t.extend(5);
        v_t(1) := 10;
        v_t(2) := 20;
        v_t(3) := 30;
        dbms_output.put_line(v_t(1));
     end;
     /
```

<u>Note</u>:- we can also specify actual value with in constructor itself in nested table, varrays in this case not required extend collection method.

```
SQL> declare
        type t1 is table of Number(10);
        v_t t1 := t1(10, 20, 30, 40, 50);
     begin
        dbms_output.put_line(v_t(1));
        dbms_output.put_line(v_t.first);
     end;
     /
o/p:- 10
      1
```

SQ1) declare

```
        type t1 is table of Number(10);
        v-t t1;
        v-t2 t1:=t1();
    begin
        if v-t1 is null then
        dbms_output.put_line('v-t1 is null');
        else
        dbms_output.put_line('v-t1 is an null');
        end if;
        if v-t2 is null then
        dbms_output.put_line('v-t2 is null');
        else
        dbms_output.put_line('v-t2 is null');
        end if;
        end;
```

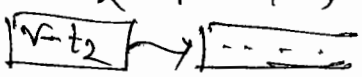
o/p:- v-t1 is null,
v-t2 is not null.

1) declare

```
    type t1 is table of number(10);
    v-t1 t1;
```

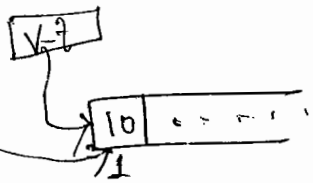
2) declare

```
    type t1 table of number(10);
    v-t2 t1:=t1();
```



3) declare

```
    type t1 is table of Number(10);
    v-t t1 :=t1();
    begin
    v-t.extend;
    v-t(1) := 10
    dbms_output.put_line(v-t(1)); end;
```



Maa 18/07/14

ex:-

```
declare
  type t1 is table of varchar2(10);
  v-t  t1 := t1('a','b','c','d');
  begin
  dbms-output.put-line(v-t(1));
  dbms-output.put-line(v-t.count);
  v-t.extend;
  dbms-output.put-line(v-t.count);
  v-t(5) := 'e';
  v-t.extend(2,3);
  ~~for~~ v-t.trim(2);
  for i in v-t.first..v-t.last
  loop
  dbms-output.put-line(v-t(i));
  end loop;
  v-t.delete;
  dbms-output.put-line(v-t.count);
  end;
```

Q) w.A. PL/SQL Program to transfer all employees names from emp table & store it in to next table & also display content from next table.
<old concept>

A)
```
declare
  type t1 is table of varchar2(10);
  v-t t1 := t1();
  cursor c1 is select ename from emp;
  n number(10) := 1;
  begin
  for i in c1
```

```
loop
v_t.extend;
v_t(n):=i.ename;
n:=n+1;
end loop;
for i in v_t.first..v_t.last
loop
dbms_output.put_line(v_t(i));
end loop;
end;
/
```

NB:-

when we are using bulk collect clause for tranfering data in to collections, then we are not allowed to use extend collection clause in nested table , becoz internally bulk collect clause only reserved memory for those collections.

{new concept}

```
sqlplus
declare
type t1 is table of varchar2(10);
v_t t1:=t1();
begin
select ename bulk collect into v_t from emp;
for i in v_t.first..v_t.last
loop
dbms_output.put_line(v_t(i));
end loop;
end;
/
```

4) ~~VARRAY~~ Varray :-

This is also an user-defined type. which is used to store similar data items in a single unit.

↳ varray stores upto 2GB data. Here also by default indexes are started from 1.

↳ Before we are storing data in to varray then we use constrant of this is an user-defind type so we are creating 2 step Process.

1st we are creating type then only we are creating a variable of datatype.

Syntax :- (of 1st step)

```
Type Typename is varray(maxsize) of datatype
                                    (size);
```

Syntax :- (of 2nd step)

```
Variable Typename := Typename();
```

constructor.

ex:-

SQL> declare

```
type t1 is varray (10) of number(10);
v_t t1 := t1(20, 30, 30, 40);
begin
dbms_output.put_line(v_t.limit);
dbms_output.put_line(v_t.count);
v_t.extend(4,2);
v_t.trim(2);
dbms_output.put_line(v_t.count);
```

```
for i in v_t.first .. v_t.last
loop
dbms_output.put_line(v_t(i));
end loop;
v_t.delete;
dbms_output.put_line (v_t.count);
end;
/
```

NB:-

In varray we are not allowed to delete particular element or using delete collection method but we can delete all the elements using delete collection method.

Q) W.A. PL/SQL Prog to transfer 1st 10 employee name from emp table into varray & also display content from varray.

A)
```
declare
type t1 is varray(10) of varchar2(10);
v_t() t1 := t1();
begin
dbms_output.put_line (v_t.count);
select ename bulk collect into v_t from emp
where rownum <= 10;
for i in v_t.first .. v_t.last
loop
dbms_output.put_line(v_t(i));
end loop;
end;
/
```

* differences betn index by table, nested table & varray :—

Index by table	Nested table	varray
1) This is an unbound table & also having key-value Pairs.	↳ This is an unbounded table.	↳ This is an bounded table which stores upto 2GB data.
2) we can not store index by table Permanently into DB.	↳ we can not store Nested table permanently into DB using SQL.	↳ we can store varray Permanently into DB using SQL.
3) we can not add, remove indexes.	↳ we can add, remove indexes using extend, trim collection methods.	↳ we can add, remove indexes using extend, trim collection methods.
4) Index values are either numbers or characters and also +ve & -ve numbers.	↳ By default indexes are stated with 1.	↳ By default indexes are start with 1.
5) Index by table having exists, First, last, prior, next, count, delete (index), delete (index$_1$, index), delete collection methods.	↳ Nested table having exists, extend, trim, First, last, Prior, next, count, delete (index), delete (index$_1$, index) delete collection methods.	↳ varray having exists, trim, limit, extend, first, last, Prior, next, count delete collection methods.

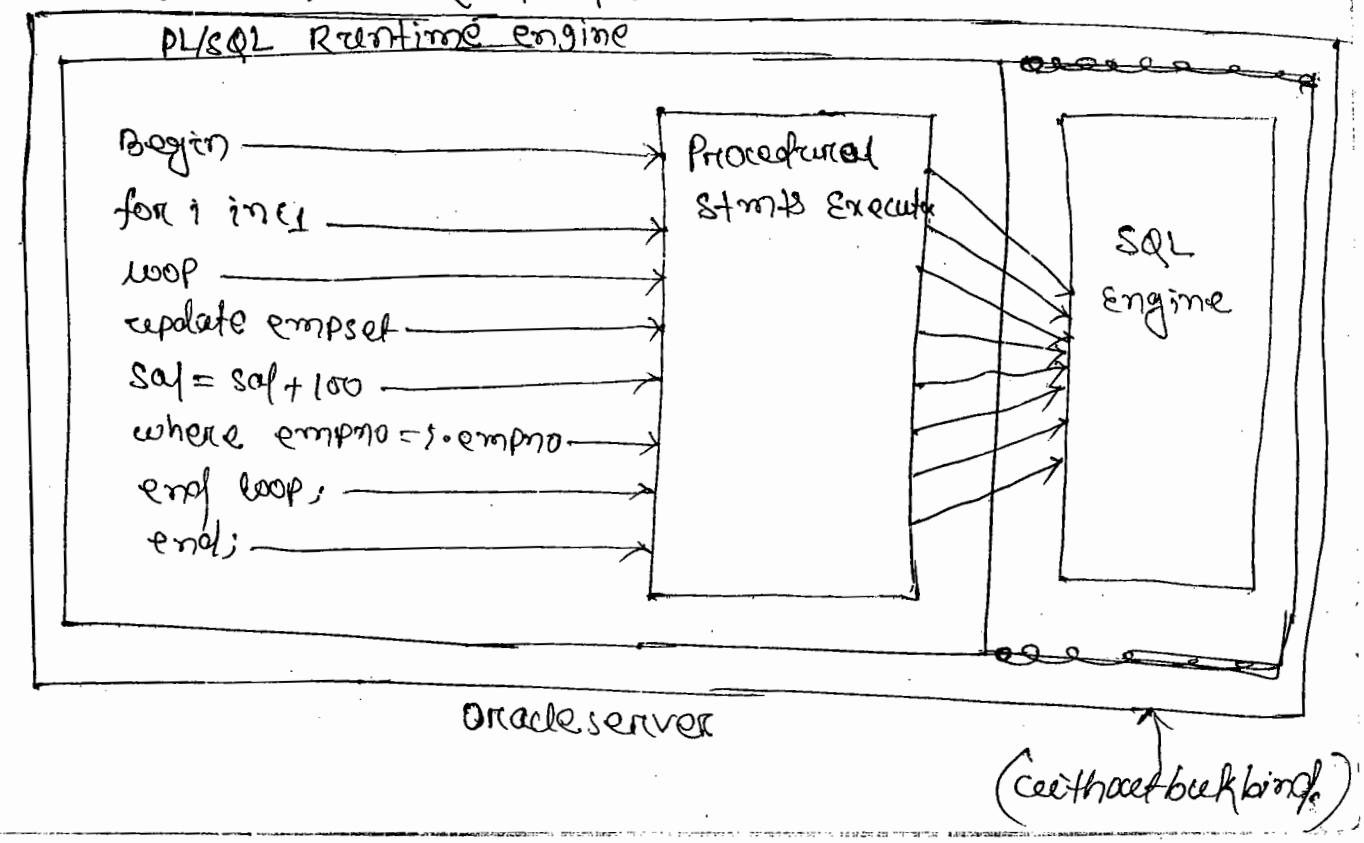
—o— —o— —o—

Maa ?

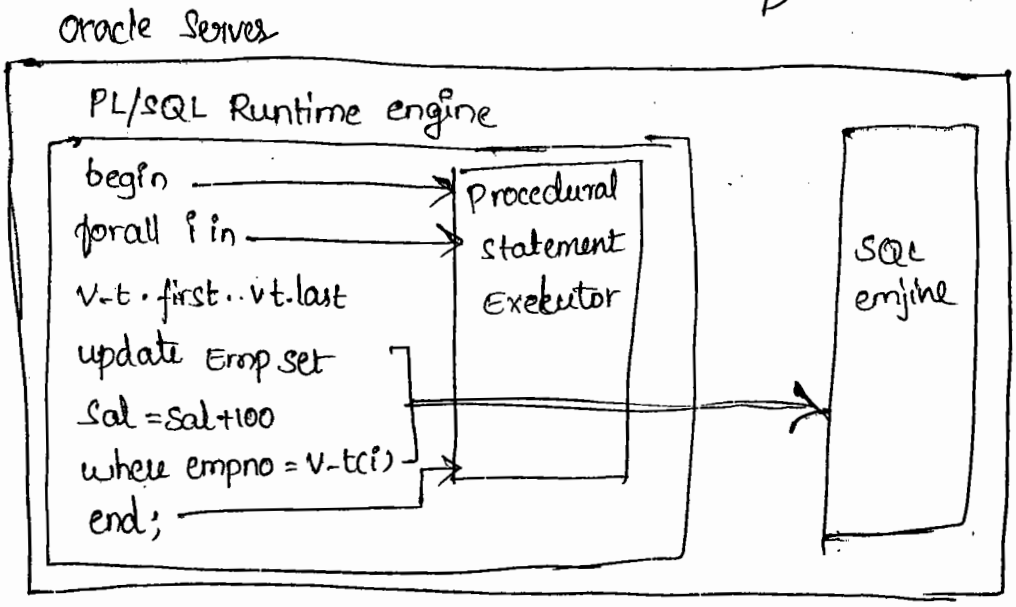
BULK BIND

19/07/14

- when a PL/SQL block contain more number of SQL, Procedural statement then all Procedural statement are executed in Procedural statement executor ~~whit~~ with in PL/SQL engine and also all SQL stmts are executed with in SQL engine. These type of execution method are also called as context switching execution methods.
- when PL/SQL block contains more number of SQL, PL/SQL statement then always context switching execution method degrades Performance of the application.
- To improve Performance of the application of the applicat ~~t~~ Oracle introduced Bulk Bind Process using collection.
- In this Process we can execute all value in a collection at a time using forall statements.
- In bulk bink Process we are using bulk update, bulk delete, bulk inserts.



using bulk bind (improve performance because of less over-head)



forall syntax :-

forall indexvarName in collection var.first..Collectionvar.last
DML statement where
colName = collectionVarName (indexvarname);

→ Before we are using bulk bind process (forall statement) we must fech data from resource into collection using Bulk collect clause, thats why bulk Bind is a two step process.

Step ①: fetch data from resource into collection using bulk collect clause

Step ②: process all data with in a collection using SQL engine through forall statment's (actual bulk bind).

Step 1:-

bulk collect clause used in

1) select ... into clause
2) cursor .... fetch ... statements
3) DML ... returning clause.

1) bulk collect clause used in select .... into clause

syntax:-

```
select * bulk collect into collection varname
from tablename
where condition;
```

ex:-

SQL>
```
declare
    type t1 is table of emp%rowtype
    index by binary_integer ;
    v_t t1;
  begin
    select * bulk collect into v_t from emp;
    for i in v_t.first .. v_t.last
    loop
     dbms_output.put_line(v_t(i).ename);
    end loop;
   end;
   /
```

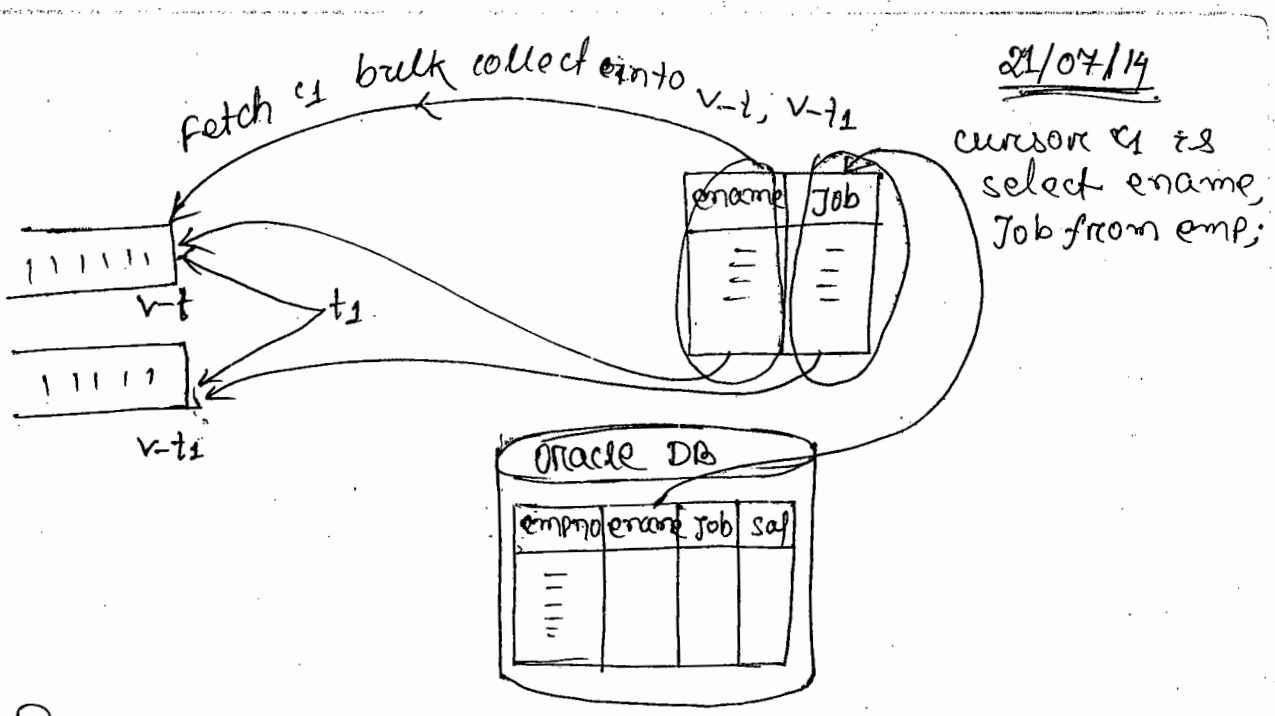
2) Bulk collect clause used in cause .... fetch ... statement

syntax:-

```
fetch cursorname bulk collect
into collectionvarname [limit anynumber]
```

Mon
7

21/07/14



Prog:-

```
SQL> declare
     type t1 is table of varchar2(10)
     index by binary_integer;
     v_t t1;
     v_t1 t1;
     cursor c1 is select ename, job from emp;
     begin
     open c1;
     fetch c1 bulk collect into v_t, v_t1;
     close c1;
     for i in v_t.first .. v_t.last
     loop
     dbms_output.put_line(v_t(i) || ' ' || v_t1(i));
     end loop;
     end;
     /
```

↳ Calculating ~~perfo~~ elapsed time using dbms_utility package :-

↳ In oracle if we want to calculate elapsed time for each process then we are using get_time

method from dbms_utility Package.

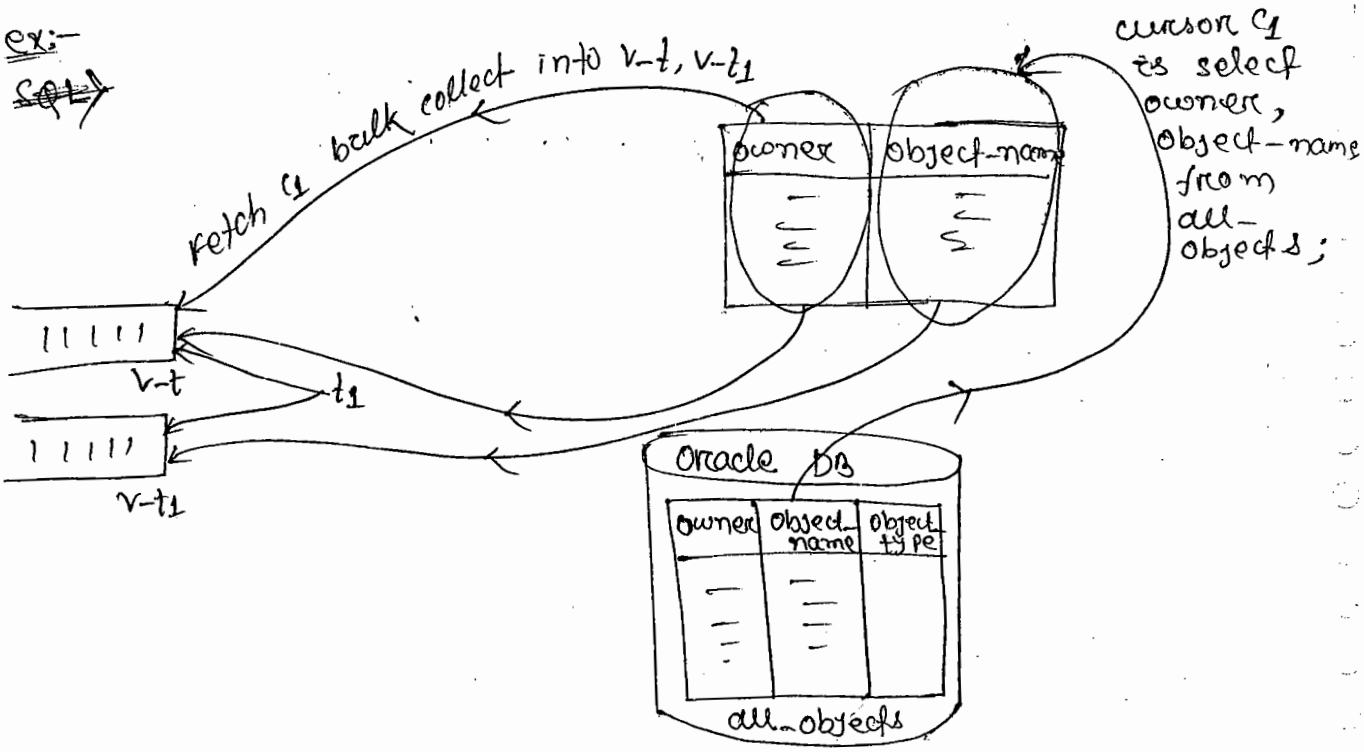
↳ This methods always returns number datatype.

Syntax:-

```
varname := dbms_utility.get_time;
```

ex:-

SQL>



Prog:-

```
SQL> declare
     type t1 is table of all_objects.OBJECT_NAME%type
     index by binary_integer;
     v_t t1;
     v_t1 t1;
     z1 number(10);
     z2 number(10);
     cursor c1 is select OWNER, OBJECT_NAME from
                                   all_objects;
     begin
     z1 := dbms_~~output~~ utility.get_time;
     ~~dbms_output~~
     for i in c1
```

```
loop
null;
end loop;
~~z2 := dbms_output.put_line~~
z2 := dbms_utility.get_time;
dbms_output.put_line ('Elapsed time for normal
    fetch ' || (z2-z1) || ' hsecs');
z1 := dbms_utility.get_time;
open c1;
fetch c1 bulk collect into ~~all_objects~~ v_t, v_t1;
close c1;
z2 := dbms_utility.get time;
dbms_output.put_line ('Elapsed time for bulk fetch '
        || ' ' || (z2-z1) || ' hsecs');
end;
```

O/P:- Elapsed time for normal fetch 104 hsecs.
Elapsed time for bulk fetch 81 hsecs.

Naga 22/07/14

Bulk collect clause used in DML returning into clause

we can also store DML transaction value into variables using ~~returning~~ returning into clauses.

Thats why Returning ... into clauses are used only in DML stmts.

eg:-

```
SQL> variable z varchar2(10);
     update emp set sal = sal + 100
     where ename = 'King' Returning job into :z;
     Print z;
     z
     president.
```

when we are processing multiple values we can also use Bulk collect in DML Returning ... into clauses through collection

```
SQL> declare
        type t1 is table of number(10)
        index by binary integer;
        v_t t1;
     begin
        update emp set sal = sal + 100
        where job = 'CLERK'
        RETURNING sal bulk collect into v_t;
        dbms_output.put_line('affected number of clerks are
                                '||' sql%rowcount);
        for i in v_t.first .. v_t.last
        loop
        dbms_output.put_line(v_t(i));
        end loop;
        end;
```

Step2: Process all data in a collection at a time using sqlengine through forall stmts:- (actual bulk bind)

Once data is available in collection we can process this data at a time using sqlengine. in this case we must use for all statement.

Syntax:-

```
forall indexvarname in collectionvarname.first .. collection
varname.last
DML stmts
where columnname = collectionvarname (index varname);
```

Ex:-

```
declare
    type t1 is varray(10) of Number(10);
    v_t t1:=t1(10,20,30,40,50);
```

```
begin
  forall i in v_t.first .. v_t.last
  update emp set sal=sal+100 where deptNo = v_t(i);
end;
/
```

Q) write a PL/SQL Prog to retrive all employee no. from emp table and store it into index by table using bulk collect. & also modify salaries of all these employees at a time using forall stmts.

A) sol) declare

```
    type t1 is table of emp.empno%type
    index by binary_integer;
    v_t t1;
  begin
    select empno bulk collect into v_t from emp;
    forall i in v_t.first .. v_t.last.
    update emp set sal=sal+100
    where empno =v_t(i);
  end;
/
```

sol2) declare

```
    type t1 is table of emp.empno%type
    index by binary_integer;
    v_t t1;
  begin
    select empno bulk collect into v_t.delete(3);
    forall i in v_t.first .. v_t.last
    update emp set sal=sal+100
    where empno =v_t(i);
  end;
/
```

error:- element at index[3] does't exist.

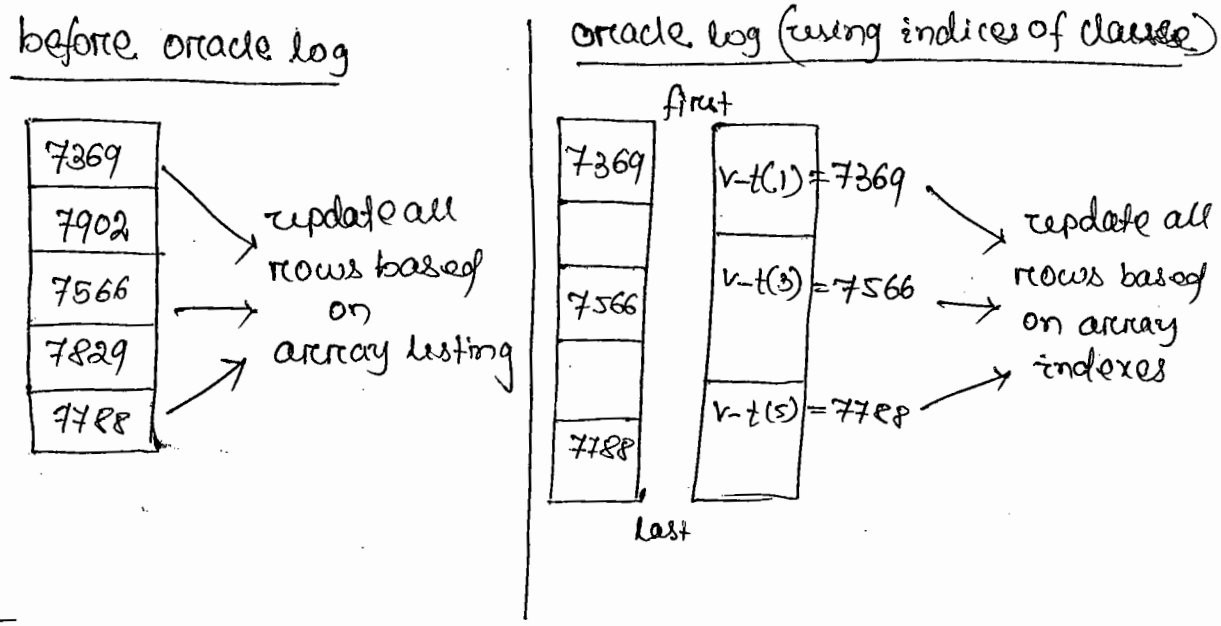
indices of

when ever index by table or nested table having gaps then we are not allowed to use bulkbind Process.

To overcome this problem oracle log introduced indices of clause in forall stmts.

Syntax:-

forall indexvarname in indices of collectionvarname
DML stmts where
columnname = collectionvarname (indexvarname);



Sol:-

```
declare
    type t1 is table of emp.emp%type
    index by binary_integer;
    v_t t1;
begin
    select empno bulk collect into v_t from emp,
    v_t.delete(3);
    forall i in indices of v_t
    update emp set sal = sal+100 where empno=v_t(i);
end;
/
```

Mar
?

Dt 23/07/14

bulk delete:~

```
SQL> declare
     type t1 is varray(10) of number(10);
     v_t t1 :=t1 (10,20,30,40,50,60);
     begin
     forall i in v_t.first.. v_t.last
     delete from emp where deptno=v_t(i);
     end;
     /
```

bulk insert:~

```
SQL> create table target (employeeno number(10));
SQL> declare
     type t1 is table of number(10)
     index by binary_integer;
     vt t1;
     begin
     select empno bulk collect into v_t from emp;
     forall i in v_t.first .. v_t.last
     insert into target values(v_t(i));
     end;
     /
SQL> select * from target;
```

SQL%bulk_rowcount:~

↳ SQL%bulk_rowcount attribut returns number datatype.
↳ if we want to return affected no. of rows number in each Process after bulk bind then we are using SQL%bulk_rowcount attribute.

syntax:~

SQL%bulk_rowcount(indexvarname)

Ex:-

```
SQL> declare
     type t1 is varray(10) of number(10);
     v_t t1 := t1(10, 20, 30, 40);
     begin
     forall i in v_t.first .. v_t.last
     update emp set sal = sal+100 where deptno = v_t(i);
     for i in v_t.first .. v_t.last
     loop
     dbms_output.put_line('affected number of rows in
          deptno '||' '||v_t(i)||' '||' is '||' '||sql%bulk_
          rowcount(i));
     end loop;
     end;
```

o/p:- affected number of rows in deptno 10 is 3.

dbms_utility Package:-

↳ This Package is used to calculate elapse time using get_time method. and also this Package internally having index by table.

↳ This Package heaving 2 methods :-

1) comma_to_table.
2) table_to_comma.

1) comma_to_table :-

↳ This method is used to convert comma separated string in to index by table values.

syntax:-

```
dbms_utility.comma_table_table(stringname,
        binary_integervarname, indexbytablevarname);
```

2) table_to_comma :~

↳ This method is used to convert index by table values into comma separated string.

syntax:-

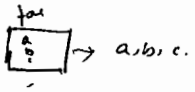
```
dbms_utility.table_to_comma(indexbytablevarname,
    binary_integervarname, stringname);
```

↳ Before we are using these methods it declare section of the PL/SQL block, we are creating a index by table variable using uncl_array type from dbm_utility Packages.

syntax:-

```
indexbytablevarname    dbms_utility.uncl_array;
```

Q) W.A. PL/SQL Program convert comma-separated string in to index by table values using dbms_utility Package and also display content from index by table.



A) SQL>
```
declare
    v_t dbms_utility.uncl_array;
    z binary_integer;
    str varchar2(200);
    begin
    str:='a,b,c,d,e,f';
    dbms_utility.comma_to_table(str,z,v_t);
    for i in v_t.first .. v_t.last
     loop
     dbms_output.put_line(v_t(i));
     end loop;
     end;
     /
```

O/P :- a
b
c
d
e
f.

Q) W.A. PL/SQL Prog retrive all department names from dept table & display those names in to comma separated strings using dbms_utility Package.

A) SQL> declare
v_t dbms_utility.uncd_array;
z binary_integer;
str varchar(200);
begin
select dname bulk collect into v_t from dept;
dbms_utility.table_to_comma(v_t, z, str);
dbms_output.put_line(str);
end;
/

O/P :- ACCOUNTING, RESEARCH, SALES, OPERATIONS.

~~FOR CURSORS~~

* ref cursors (or) cursor variable (or) dynamic cursor :-
imp

- Oracle 7.2 introduce ref cursors.
- This is a user_defined type which is used to Process multiple records. and also this is an record by record Process.
- Generally in static cursors database servers executes only one select stmts at a time for a single active set area. where as ref cursor DB servers executes no. of select stmts dynamically for a single active set area.

Mar 7 Time

24/07/14

ref cursor :-

- That's why this cursors is also called as dynamic cursors.
- Generally we are not allowed to pass static cursor as Parameter to the some Programs.

  To overcome this Problem ANSI/ISO SQL introduced ref cursor which is used to Pass Parameter to the some Programs becoz basically ref cursors is a user-defined type.

- Generally static cursors does't return multiple values in to the client applications, where as ref cursors returns multiple values in to client applications.
- 2 types of ref cursors in all DB.
  1) strong ref cursors.
  2) weak ref cursors.
- Strong refcursor is ref cursor which having return type, where as weak ref cursor is a ref cursor which does't have return type.
- This is an user-defined type so we are creating in 2 step Process.

  1st we are creating type then only we are creating a variable of data type.

  That's why this cursors is also called as <u>cursor variables</u>.

Syntax:-

① Type Typename is ref cursor return record datatype

② variablename Typename;

Strong refcursor variable.

② Type Typename is Ref cursor;
variablename typename;

weak refcursor variablename.

↳ In ref cursors we are specifying select stmts using open.... for stmts. This stmts is used in excutable section of the PL/SQL blocked.

Syntax:-

```
open refcursorvarname for select * from
                    tablename where condition;
```

Ex:-

```
SQL> declare
     type t1 is ref cursor;
     v-t t1;
     i emp%rowtype;
     begin
     open v-t for select * from emp where sal>2000;
     loop
     fetch v-t into i;
      exit when v-t%notfound;
     dbms_output.put_line(i.ename || ',' || i.sal);
     end loop;
     close v-t;
     end;
     /
```

Q) W.A. PL/SQL Prog using ref cursor whenever user entered deptno 10 then display 10th dept details from emp table and also when ever user entered deptno 20 then display 20th dept details from dept table.

A) SQL> declare

```
type t1 is ref cursor;
v_t t1;
i emp%rowtype;
j ~~dept~~ dept%rowtype;
v_deptno number(10) := &deptno;
begin
if v_deptno = 10 then
open v_t for select * from emp where deptno=10;
loop
fetch v_t into i;
exit when v_t%notfound;
dbms_output.put_line(i.ename||' '||i.sal||' '||
                                    i.deptno);
end loop;
elseif v_deptno=20 then
open v_t for select * from dept where deptno=20;
loop
fetch v_t into j;
exit when v_t%notfound;
dbms_output.put_line(j.deptno||' '||j.dname||
                                    ' '||j.loc);
end loop;
close v_t;
end if;
end;
/
```

Note:- sys_refcursor :-

↳ In place weak refcursor oracle introduced sys_refcursor predefined type.

syntax:-

```
refcursorvarname sys_refcursor;
```

Ex:-

```
SQL> declare
     v_t sys_refcursor;
     i emp%rowtype;
     begin
     open v_t for select * from emp where sal>3000;
     loop
     fetch v_t into i;
     exit when v_t%notfound;
     dbms_output.put_line(i.ename||','||i.sal);
     end loop;
     close v_t;
     end;
     /
```

↳ Passing ref cursor as parameter to the subprograms:-

↳ Passing refcursor as in parameter to the stored procedure:-

Q) Co.A. PL/SQL stored procedure on passing ref cursor as a 'in' parameter to display empdetails from emp tables.

A) ~~SQL> declare~~

Q) SQL> ~~declare~~ create or replace Procedure

```
P1 (v-t in sys_refcursor)
is
i emp%rowtype;
begin
loop
fetch v-t into i;
exit when v-t%notfound;
dbms_output.put_line (i.ename || ' ' || i.sal);
end loop;
end;
/
```

execution using PL/SQL client :-

```
SQL> declare
     v-t sys_refcursor;
     begin
     open v-t for select * from emp;
     P1 (v-t);
     close v-t;
     end;
     /
```

Passing refcursor as out Parameter to the stored Procedure:-

Q) w.A. PL/SQL stored Procedure Passing refcursor as a out Parameter which returns emp details from emp table.

Q) SQL> create or replace Procedure P

```
P1(v-t in sys_refcursor)
begin
open v-t for select * from emp;
end;
/
```

execution using PL/SQL client :-

```
SQL> declare
     v_t sys_refcursor;
     i emp%rowtype;
     begin
     P1(v_t);
     loop
     fetch v_t into i;
     exit when v_t%notfound;
     dbms_output.put_line (i.ename||' '|| i.sal);
     end loop;
     close v_t;
     end;
     /
```

Mar 25/07/14

Ex:-

```
SQL> create or replace package PK1
     is
     type t1 is ref cursor return
     emp%rowtype;
     Procedure P1(v_t1 out t1);
     type t2 is ref cursor return
     dept%rowtype;
     Procedure P2(v_t2 out t2);
     end;
     /
SQL> create or replace package body pm1
     is
     Procedure P1 (v_t1 out t1)
     is
     begin
     open v_t1 for select * from emp;
     end P1;
```

```
Procedure P2(v-t2 out t2)
is
begin
open v-t2 for select * from dept;
end P2;
end;
/
```

execution :- (using Bind variable)

```
SQL> variable a refcursor;
SQL> variable b refcursor;
SQL> exec Pk1.P1(:a);
SQL> exec Pk1.P2(:b);
SQL> Print a b;
```

*Note :-

↳ In all DB systems we are not allowed to used refcursors directly in Packages. Becoz Bare ~~cursor~~ refcursor is a user-defined type.

~~Utl_file Packages~~ —o—

Utl_file Packages

- Oracle 7.3 introduced utl_file Package.
- This Package is used to write data into an OS file and also read data from OS file.
- If we want to write data into OS file then we are using Putf() method from utl_file Package. and also if we want to read data from OS file then we are using get_line() method from utl_file Package.
- whenever we are using utl_file Package or lobs then we must creat alias directory related to physical directory using following syntax:-

syntax:-

```
create or replace directory
directory name as 'Path';
```

(case sensitive)

- Before we are creating alias directory DB administrator for given create any directory.

```
grant create any directory to username;
```
← syntax

Ex:-

```
SQL> conn sys as sysdba;
     Enter Password : sys
SQL> grant create any directory to scott;
SQL> conn scott/trigger;
SQL> create or replace
     directory ABC as 'D:\';
```

↳ Before we are performing read, write operations then we must use read, write object Privileges.

Syntax:-

```
grant read, write on directory
directoryname to username;
```

```
SQL> conn sys as sysdba;
     Enter Password : sys
SQL> grant read, write on
     directory ABC to scott;
SQL> conn scott/triger;
```

writing data into an external file:-

Step1: Before we are open in the file we must declare a file pointer variable in declaration section of the PL/SQL block, using File_type from utl_file Packages.

Syntax:-

```
filepointervarname    UTL_file.file_Type ;
```

Step2: Before we are writing data in to file we must open the file using fopen() from utl_file Package.

↳ This function accepts 3 Parameters and returns file_type datatype.

↳ fopen() is used in executable section of the PL/SQL block.

~~Syntax~~:-

Syntax:-

```
filepointervarname := UTL_file.fopen ('Aliasdirectoryname',
                                      , filename, 'mode');
```

→ w
→ r
→ a

Step3: For writing data into file then we must use Putf() from utl_file package.

Syntax:-

```
utl_file.Putf(filepointervar, content);
```

Step4: After writing data into file we must close the file using fclose() from utl_file package.

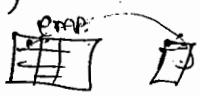
Syntax:-

```
utl_file.fclose(filepointervar);
```

Example:-

```
SQL> declare
     fp utl_file.file_type;
     begin
     fp:=utl_file.fopen('ABC','fan.txt','w');
     utl_file.Putf(fp,'Hi Hello world');
     utl_file.fclose(fp);
     end;
     /
```

Q) Wr. A. PL/SQL Program write all Employee names from emp table in to external file using utl_file Package.



A) TM

```
declare
fp utl_file.file_type;
begin
fp:=utl_file.fopen('EMP','emp.text','w');
```

```
declare
fp utl_file.file_type;
cursor c1 is select ename from emp;
begin
fp := utl_file.fopen('ABC', 'file1.txt', 'w');
for i in c1
loop
utl_file.putf(fp, i.ename);
end loop;
utl_file.fclose(fp);
end;
/
```

Mar 7

26/07/14

Note:-

* whenever we are using utl-file package to write columndata in to file then DB server automatically write data horizentaly in a file.

↳ To overcome this Problem we are using Putf() second Parameter with format specifier %S.

Example:-

```
SQL> declare
fp utl_file.file_type;
cursor c1 is select ename from emp;
begin
fp := utl_file.fopen('ABC', 'file1.txt', 'w');
for i in c1
loop
utl_file.Putf(fp, '%s\n', i.ename);
end loop;
utl_file.fclose(fp);
end;
/
```

Reading data from file :-

using get_line() from utl_file package, we can read data from the file.

syntax :-

```
utl_file.get_line(filepointervarname, buffervar);
```

Example :-

```
SQL> declare
     fp utl_file.file_type;
     x varchar2(200);
     begin
     fp:= utl_file fopen('ABC', 'fan.txt', 'r');
     utl_file.get_line(fp, x);
     dbms_output.put_line('data from file'||' '||x);
     utl_file.fclose(fp);
     end;
     /
```

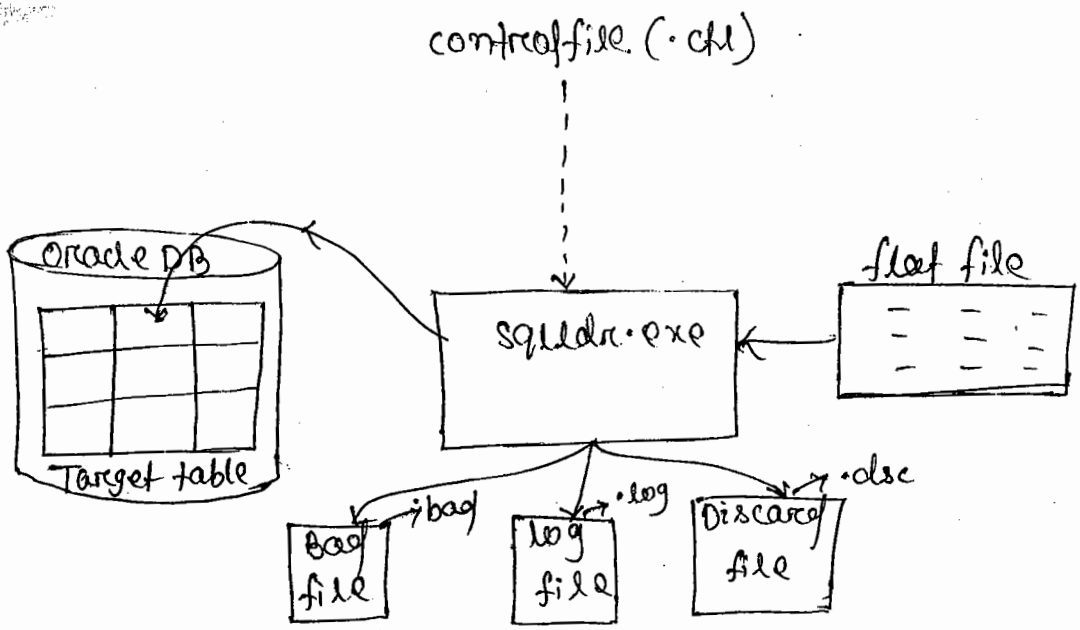
Q) w.A. PL/SQL Prog read multiple values from ~~file1~~ file1.txt and display those values using utl_file Package.

```
declare
fp utl_file.file_type;
x varchar2(200);
begin
fp:= utl_file fopen('ABC','file.txt', 'r');
loop
utl_file.get_line(fp, x);
dbms_output.put_line(x);
end loop
utl_file.fclose(fp);
exception
when no_data_found then
```

```
null;
end;
/
```

Note:-

- In oracle when we are reading multiple values from a file using utl_file Package, then oracle server returns an error ORA-01403 no_data_found the when control reach to the end of file.
- To overcome this Problem we must use no_data_found exception name.
- **SQL Loader:-**
  SQL loader is a utility Program which is used to transfer data from flat file into oracle DB.
- SQL loader is also called as <u>bulkloader</u>.
- Always SQL Loader executes <u>control files</u>. This file extension is '<u>.ctl</u>'.



- Based of types of flat file we are creating control file & then submit this control file to sql loader tools then only sql loader transfer data from flat file into oracle DB.
- During this Process sql loader automatically creates a logfiles, and also this file stored on other file interaction & also stored oracle errornos, error msg & also stored loaded, rejected ~~Visited~~ no. of record no.s.
- Based on some reasons some records are rejected from flat files, those records are stored in bad file, discard files.

Invoking SQL Oracle :-

- start → run → cmd ↵ d:\ sqlldr userid = scott/tiger ↵ control = Parth of the controlfile

flat file :-

- flat file is an structured file which contains no. of records.
- Oracle haveing two types of flat files i.e
  1) Variable record flatfile.
  2) fixed record flat file.
- variable record flatfile :-

A flatfile heaving delimiter & also each field having diff length then those type of flatfile cured called variable record flat file.

ex:- 101, abc, 2000
202, ccc, 7000

2) Fixed record flat file :-

A flat file which doesn't have delimiters, those flat files are called as fixed record flat file.

Ex :- 101abc2000
202cc7000.