

# **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**



maq

## Oracle 11g

20<sup>th</sup>/01/14 (4.00 p.m.)

- Oracle is a database server s/w.  
Ex:- SQLserver DB2 oracle sybase Ingrees MySQL Teradata SQLite.
- All are implemented by ORDBMS (<sup>Object</sup>~~Relational~~ Relational DBMS)
- To store data permanently 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 corporation.
- Oracle is a database Server which is used to store data permanently 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.

Information:- When we are processing data that we are

easily meaningful preserves result this is called as information.

Ex:- 1) Student marksheet.

2) invoice of a customer.

Data {  
101 soma 2000  
102 Patel 300

slno	name	sal
101	soma	2000
102	Patel	300

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 :-

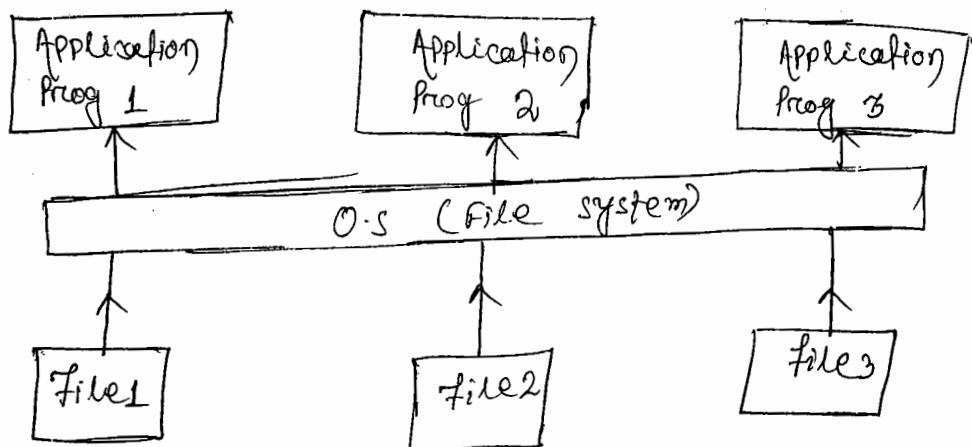
- \* Data retrieval.
- \* Data redundancy.
- \* Data integrity?
- \* Data security?
- \* Data indexing?

\* Data retrieval :-

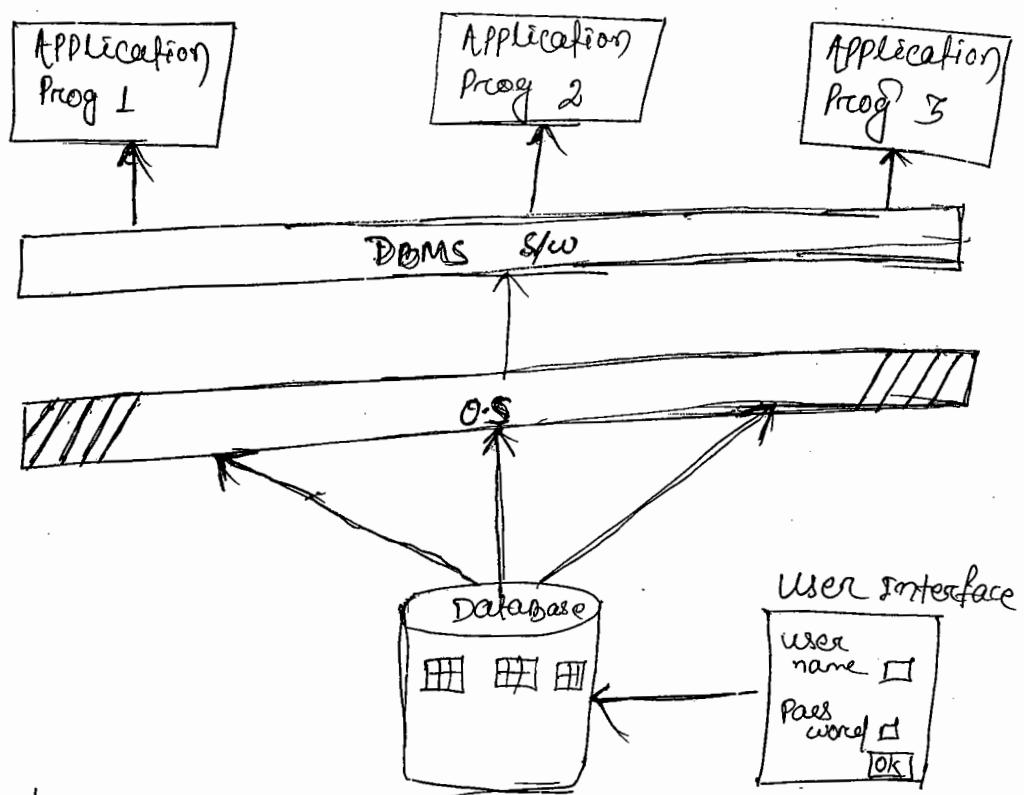
If we want to retrieve data from flat files that be must a develop application & programs in High level languages. whereas as once a data stored in databases that we are retrieve a data using standard ~~SQL~~ language (SQL).

- Flat file mechanism every application program make a maintain it's own file separate from other application.

## (FLAT file Mechanism)



when we use the DBMS s/w,



• when Data . we are installing DBMS s/w automatically some places creating in disk , this is called physical Database and also called automatically on user interfaces is created. Through files we are interacting with the database otherwise through app prog indirectly we are interactive with the DB.

Mark

(04.00 PM) 21/01/19

### \* Data Redundancy :-

- Sometimes we're maintaining multiple copy of same data in different locations.
- This Data is also called as Redundant Data or Replicate Data, in flat file mechanism where we're modifying data in one location, it is not reflected in another location. This is called inconsistency.
- Databases automatically maintain consistency data through ACID Properties i.e. in databases every transaction internally having 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 maintaining Proper data through constraints, triggers.
- If we want to maintain Proper data in flat files we must develop application programs in a High level languages.

### \* Data Security :-

- Data stored in flat files can not be secured bcoz flat files does not provide security mechanism. whereas as databases provides role based security.

### \* Data indexing :-

- If we want to retrieve data very quickly that we are using indexing mechanism in databases whereas as flat files does not provide indexing mechanism.

organizations suffering from some flat file mechanism to store data.

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

DBMS: (Database Management system)

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

→ whenever we are 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 standard inst. has established 3 level architecture for DBMS.

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

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

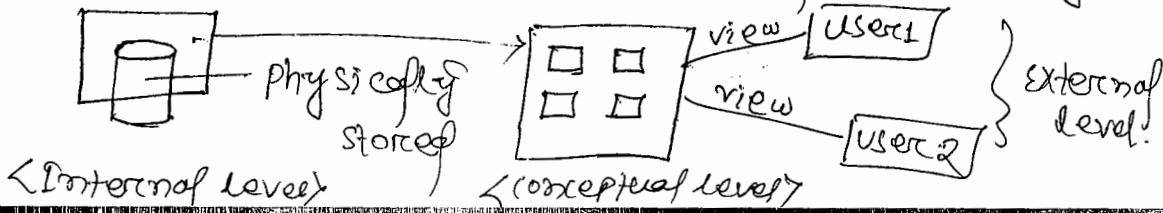
→ 3 levels are :-

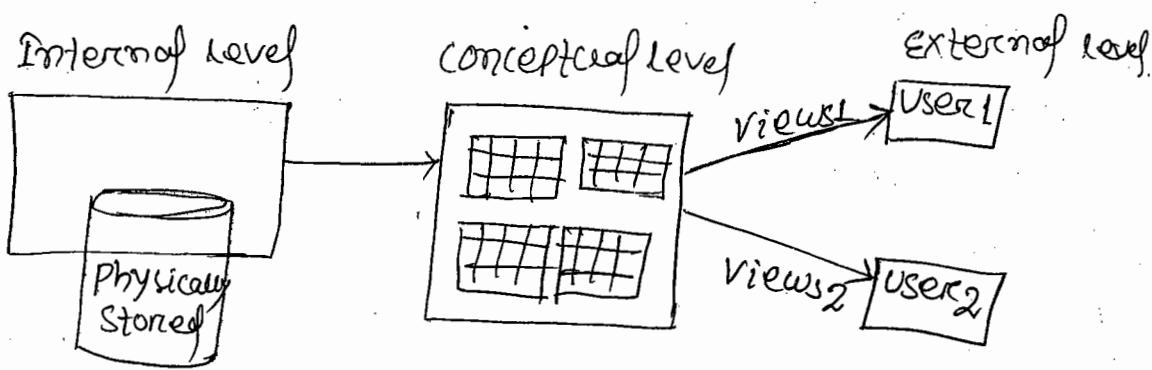
1) External Level.

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(acno number(10), name varchar2(10), balance number(10));

→ conceptual view designed by either developer or Database Administrator (DBA)

#### Internal level :-

Eg:- acno 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 personally database.

#### Data Model :-

→ How data layout 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.

Primary Key  
Dept

Dept no	Dname	Loc
10		
20		
30		
40		

Emp:

Empno	Ename	Dept no
		20
		30
		30
		20
		30
		30

3) Relational Data Model :-

<Master table>

<child table>

→ whenever a table column 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 then impossible becoz it exist in child table repeated data are exist.

→ 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.

↳ Hierarchical Data Model :-

Ex:-

Primary Key	Empno	Ename	Dept
	1001	Murali	S/W
	1002	Abc	HR
	1003	Zyz	Admin

foreign key

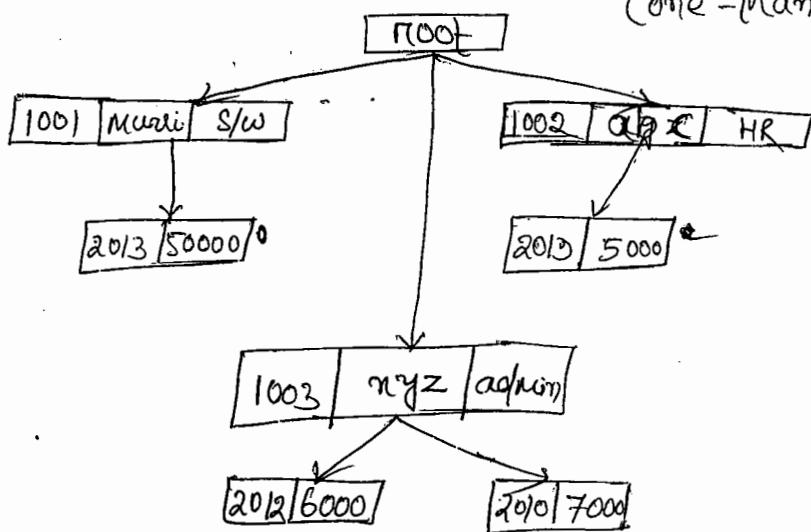
Loan Details.

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

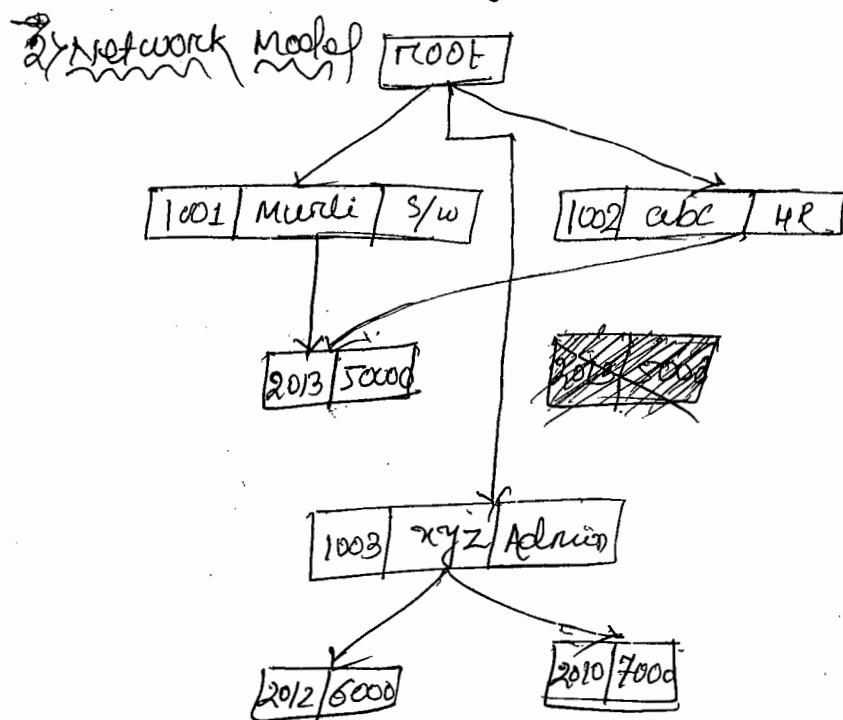
<Master table>

<child table>

(One-Many), i.e by hierarchical.



→ In Hy. model data are repeated. So more redundant occurring here.



< Networking model >

→ In networking model data are not repeated so that no redundant occurring here.

→ it is many-many relationship.

## 1) Hierarchical Data Model :-

- In Hierarchical data model organizes data in tree like structure.
- This hierarchy also called as parent child hierarchy.
- This model implemented based on one to many Relationships. i.e. Many child records having single Parent. that's why in this structure ~~all~~<sup>some</sup> child data segments are repeated. that's why this structure maintains more duplicate data.
- In hierarchical model data stored in a format of records and also record types is corresponding to the table in Relational model.
- In 1960 IBM introduced IMS (Information mgmt system) Product. Based on ~~Hierarchical~~<sup>an</sup> data model.
- If we want to operate Product we are using Hierarchical data model.
- If we want to retrieve data from hierarchical node onwards. then we are data root retrieve data very slowly. these type of Product from the database.  
eg:-

## 2) Network Data Model :-

- In 1970 CODASYL (conference on Data system language) committee introduced Network model.
- This Datamodel based on Many to many Relationships. that's 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 management system) based on network data model.

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

e.g.,

### 3) Relational Data Model :-

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

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

→ Relational data model consist of 3 components i.e

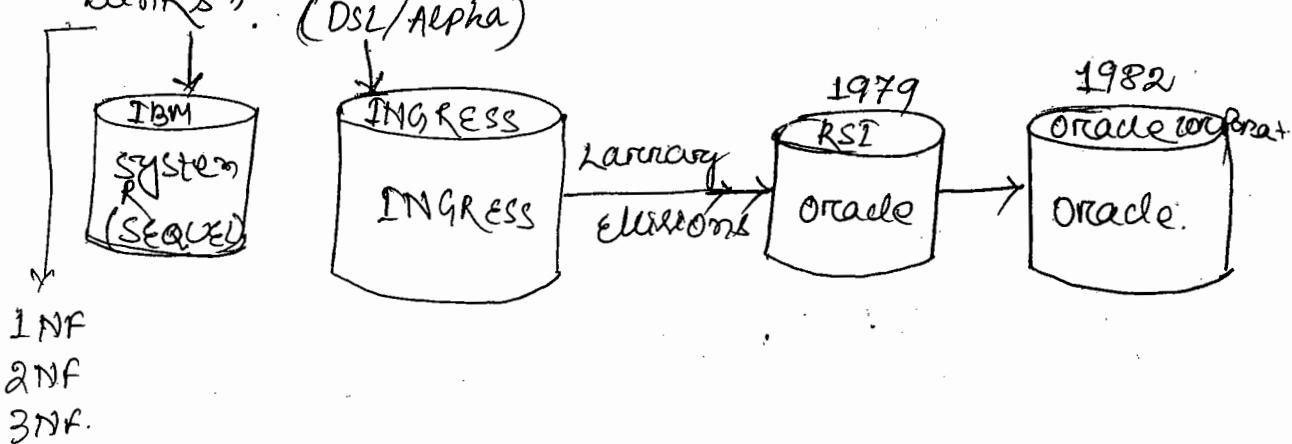
1) collection of objects.

2) Set of operations.

3) Integrity rules.

\* → In 1970 E.F. CODD written a paper

"Relational model of Data for large shared data banks". (DSL/Alpha)



## Oracle :-

1) SQL

2) PL/SQL

3) Dynamic SQL.

## Oracle Versions :-

Oracle 2.0 is 1<sup>st</sup> version released on 1979.

→ first Public release.

→ Basic SQL functionality, 1978.

Oracle 3.0 is 2<sup>nd</sup> version released on 1983.

→ commit, rollback.

→ rewritten in C language.

Oracle 4.0 is 3<sup>rd</sup> version released on 1984

→ Real consistency

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).
- columns increase per a table up to 1000.
- Asets instead of triggers.

Oracle 8i (Internet) ----- 1998.

- Materialized based indexes.
- enable / disable triggers.
- Rollup, cube.
- autonomous transactions.

Oracle 9i ----- 2001

- Renaming a column
- merge statement.
- multitable insert.
- flashback query.
- ansi (or) 9i joins.

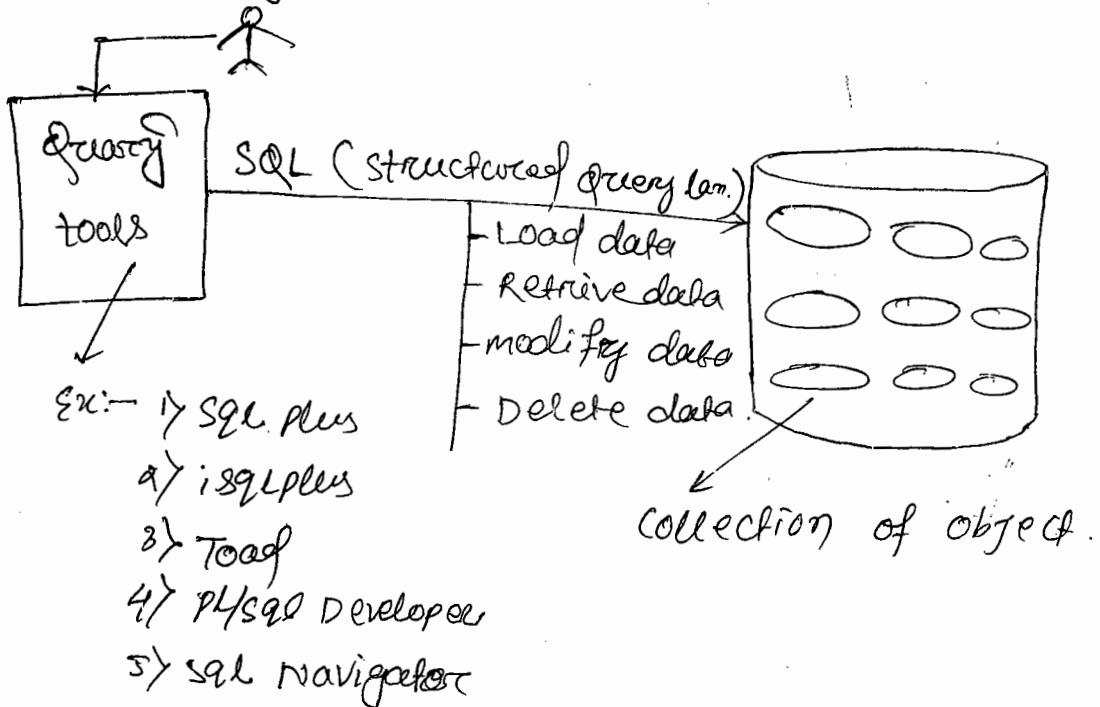
Oracle 10g (grid technology) ----- 2005

- 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
- Virtual column
- Simple integer datatype introduce in PL/SQL
- follows clause in triggers.
- sequences used in PL/SQL without drop table.

- Enable, disable clauses used in trigger specification
- Named, mixed notations are used in a job 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 relocking a user :-

Username

[ sys as sysdba ] ↳

Password

[ sys ] ↳

SQL> Alter user scott account unlock;

SQL> Conn scott/Triger ↳

Password: Triger ↳

Confirm pwd: tiger ↳

→ This Process is only 1<sup>st</sup> 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 nonprocedural 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 square into sequel and  
sequel into SQL.

→ In 1986 introducing ANSI.

→ In 1987 " ISO.

→ In 1989 " ANSI / ISO SQL89

→ 1992 " ANSI / ISO 1992

→ 1999 " ANSI / ISO SQL99

→ 2003 " ANSI / ISO SQL 2003.

SQL command types :-

• DDL (Data definition language)

→ Create

→ Alter

→ Drop

→ Truncate

→ Rename (Oracle 9i)

• DML (Data manipulation language).

→ Insert

→ Update

→ Delete

→ Merge (Oracle 9i)

• DQL (Data query language) OR DRL (Data Retrieval lan)

→ Select.

• TCL (Transactional control language)

→ Commit (Save)

→ Rollback (like undo).

→ Save Point

• DCL (Data control language)

→ Grant (Permission)

→ Revoke (Deny).

By default.

→ DDL commands are automatically save it; it is a standard.

→ DML and TCL commands are used directly to the PL/SQL.

(4.10 PM) 25<sup>th</sup> Jan, 14

Ma

## Data Types :-

- Datatypes identifies — type of data with in a table column.
- 1) Number (P,S) ex:- number (7,2), number (4). → for float
  - 2) Char → varchar (Max size) → for integer
  - 3) Date

### Number (P,S) :-

- Number (P,S) → scale
- Precision → Total no. of digits.
- It is used to stored fixed, floating point nos.

#### Syntax:-

columnname	number (P,S)
------------	--------------

→ Max limit of precision is upto 38.

#### Char :-

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

#### Syntax:-

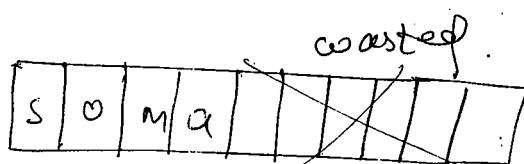
columnname	char (size)
------------	-------------

→ By default char datatype having 1 byte.

→ Max limit is upto 2000 bytes.

T<sub>1</sub> → char(10)

name
soma



This type Problem is occurred then we go to varchar.

To check interstorage data see query is.

SQL Select \* from dump(name) from g;

off

space in 1cell

Table name-

		12	32	32	32	32	32	32
--	--	----	----	----	----	----	----	----

### Varchar<sub>2</sub> (max size) :-

- It is used to stored variable length alphanumeric in bytes.
- max limit is up to 4000 bytes.

Syntax :-

columnname varchar<sub>2</sub> (maxsize)

Date :-

- it is used to stored fixed oracle date format.

Syntax :-

columnname date

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

DDL :- (Data def<sup>n</sup> language)

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

Create :-

It is used to create database objects like Tables, Views, sequences, etc.

\* Creating a table : keyword

Syntax - Create table tablename ( columnname1 datatype (size), columnname2 datatype (size), ... )

Ex:- create table η<sub>1</sub> ( sno number(10), name varchar<sub>2</sub>(10));  
— Table is created.

SQL desc η<sub>1</sub>;

Table η<sub>1</sub>

Ex:-

sno	-----	number (10)
name	-----	varchar <sub>2</sub> (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 number(10)};

sno - - - - - number(10).

Name - - - - - varchar(10)

Salary - - - - - number(10)

⇒ Modify :-

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

Syntax :

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

Ex:- I want to change my sno ~~to~~ which is name is changing to Date.

Syntax : alter table n1 modify sno date; ↓ desc n1;

sno - - - - - date

Name - - - - - varchar(10)

Salary - - - - - number(10)

⇒ Drop :-

→ it is used to remove columns 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 parentheses then we are using following syntax.

alter table tablename drop column keyword columnname;

Ex:- alter table n1 drop column sno; ↴ desc n1 ←  
Table n1

Name ----- varchar(10);  
 Salary ----- number(10);

Method 2:-

→ if we want to drop a single <sup>or</sup> multiple column with using Parenthesis 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 columns in a table;

\* ~~Drop~~

Drop :-

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

Syntax:

drop objecttype objectname .

e.g:-

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

Dropping a table :-

Syntax:

drop table tablename

e.g:- drop table n1;

Table dropped.

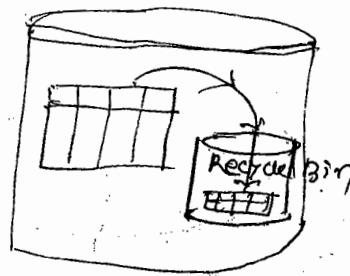
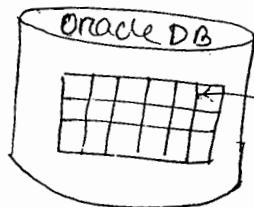
→ Oracle before 10g

dropping a table

oracle10g (Enterprise edition)

→ drop table tablename;

→ drop table tablename;



→ get it Back from Recyclebin

→ flashback table tablename  
to before drop;

→ To drop permanently

→ drop table tablename Purge

Recyclebin which is stored:-

eg; SQL> flashback table n1 to before drop;

SQL> flashback comple. (it come from recyclebin).

Permanently drop:-

eg; SQL> drop table n1 purge;

Table dropped.

(table is not exist even in recycle bin).

testing

SQL> flashback table n1 to before drop;

Error: object not in recycle bin.

Recyclebin :-

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

→ This table stores dropped 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 tablename;

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

Purge recyclebin;

sql) create table t1 (no member (10)); sql) drop table t1;

sql) desc recyclebin; (to view recyclebin;)

sql) select ORIGINAL-NAME from recyclebin;

ORIGINAL NAME

C7.

sql) Purge table C7; (← dropping from recyclebin;)

4) Truncate :-

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

Syntax:

truncate table tablename;

→ eg:

sql) create table t2 as select \* from emp;

say Select \* from cq;

say Truncate table cq;

Testing:-

→ want to view all data.

say Select \* from cq;  
no row selected.

→ want to view Table;

say desc cq;

5) Rename:-

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

Renameing a table:-

→ Syntax:

rename oldtablename to newtablename;

eg:- say rename cq to dq;  
table created.

→ Renameing a column (Oracleqi):-

Syntax:

alter table tablename rename column oldcolumn  
to newcolumnname.

→ only one column at a time.

eg:- alter table emp rename empno column emplno  
to slmo.

say desc emp;

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

## 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 q1(SLNO Number(10), NAME Varchar(10));

SQL> insert into q1 values (1, 'soma');

1 row inserted.

SQL> select \* from q1;

SNO	Name
1	soma

These are called record.

#### Method 2 :- (using substitutional operator(%)

→ Syntax:

insert into tablename values (%col1,%col2...);
--

eg:- SQL> insert into q1 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.

SQL select \* from g1;

SLNO	Name
1	Soma
2	Soma Patel
3	Patel

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

eg:- & SLNO. → &1

Method 3:- (skipping column).

Syntax:-

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

e.g)- insert into g1 (name) values ('Om');  
1 row created.

SQL select \* from g1;

SLNO	Name
1	Soma
2	Soma Patel
3	Patel
4	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 & add.

SQL alter table g1 add address.  
1 column is created.

SQL select \* from g1;

SLNO	Name	Address
1	Soma	
3	Soma Patel	

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

→ It is possible using the 'update' command.

### UPDATE :-

It is used to modify data in a table.

Syntax:

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

eg:- update ~~g1~~

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

N.B:-

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

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

1 row updated;

SQL select \* from g1;

<u>SNO</u>	<u>Name</u>	<u>Address</u>
1	soma	mumbai
2	somapatel	
3	Patel	

→ If I want to delete a Particular data from table like in add column mumbai is delete, it is possible using 'update' command.

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

SQL select \* from g1;

<u>SNO</u>	<u>Name</u>	<u>Address</u>
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 b/w truncate and delete command :-

→ whenever we are using truncate or delete from tablename then Oracle server automatically deletes total data.

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

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

→ when we are using truncate we cannot get it back deleted data using rollback becoz truncate is a DDL command by default all DDL commands are automatically committed.

Data retrieval language or Data query language (DQL):~

↳ Select :~

Syntax:

```
select columnname1, col2 from table name [where  
condition / group by columnname / having condition]  
order by columnname [asc/desc];
```

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

e.g:-

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  
to another table constraints never copied.

Creating a new table from existing table without copying  
data :~

Syntax:

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

Q:-

SQL create table second as select \* from emp  
where 1=2;

SQL select \* from second,  
no row selected (becz there is no row are  
exist). *this type*  
say desc second;

Operators used in select statement :-

- 1) Arithmetic operators. (+, -, \*, /) → After select col<sub>1</sub>, col<sub>2</sub>.
- 2) Relational operators. (≤, ≥, =, ≠)
- 3) Logical operators. (AND, OR, NOT)
- 4) Special operators.

Must use 'where'.

- ~~is used~~ Arithmetic oper. are used in either where condition or After select col<sub>1</sub>, col<sub>2</sub>.
- Relational, logical or special operators are used in 'where'.
- Arithmetic operators are used in number, date datatype column.

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

Say Select ename, sal, sal\*12 "annual" from emp;  
SQL,

Q. write a query to display a employee accept  
Job as clerk from emp table?

SQL Select \* from emp where job < > 'CLERK';  
SQL,

Q: write a query to display the emp who are more than 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 then 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 depts 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
- 2) Between
- 3) IS NULL
- 4) LIKE

##### IN :-

- it is used to pick the value one by one from list of values.
- generally instead of all operators we are using 'IN' operator because '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.

##### Opposite

- not in
- not between
- is not null
- not like

All are used in where

4.00 PM, 30th Jan/24

NB:

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

Select \* from emp where ename in

('SMITH', 'JONES');

Select \* from emp where deptno IN(10,20,NULL);  
(it works)

Select \* from emp where deptno NOT IN(10,20,NULL);  
~~Error~~: no rows selected. (it doesn't work).

NB:-

NOT IN operator does not work with null values.

Between :-

- This operator is used to retrieve range of values
- This operator is also called as between and operator.

Syntax:-

Select \* from tablename where columnname bet<sup>n</sup>  
AND highvalue;

eg:- Select \* from emp where sal bet<sup>n</sup> 5000 and  
2000

Error: no row selected.

beacoz:- bet<sup>n</sup> operator is not work high to  
so, low value.

SQL select \* from emp where sal bet<sup>n</sup> 2000;

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

NULL:-

→ Null is a undefined, unavailable, unknown  
value.

→ it is not same as '0'.

→ Any arithmetic operation perform null again it becomes null.

Ex:- null + 50 = null.

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

eg:-

SQl> Select ename, sal, ~~totcomm~~ comm, 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 in place of null.

Syntax:-

NVL (exp<sub>1</sub>, exp<sub>2</sub>)

→ Here exp<sub>1</sub>, exp<sub>2</sub> must belongs to same datatype.

→ If exp<sub>1</sub> is null then it returns exp<sub>2</sub> otherwise it returns exp<sub>2</sub>, ~~otherwise~~.

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

O/P:- 50

O/P:- 30

eg:- 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';

<u>ENAME</u>	<u>SAL</u>	<u>COMM</u>	<u>SAL + COMM</u>
SMITH	400	--	400

$$\begin{aligned}
 & \text{Sal} + \text{NVL}(\text{comm}, 0) \\
 \Rightarrow & 400 + \text{NVL}(\text{NULL}, 0) \\
 \Rightarrow & 400 + 0 \\
 \Rightarrow & 400.
 \end{aligned}$$

→ if exp<sub>1</sub> & exp<sub>2</sub> datatypes are not same then it does not work so we convert to type casting.

eg:- select ename, sal, comm, nvl(to\_char(comm), 0)  
 from emp ~~where~~ where name = 'SMITH';

~~SMITH~~ ~~400~~ ~~comm~~  
~~is null~~, is ~~not null~~ :-

→ These two special operators are used in where condition.

→ These 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 where mgr=null;  
 no row selected (not error but show)  
 this type of qn.

- Select \* from emp where comm is null;

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

-s/q Select \* from emp where comm is null;

Like:-

- This operator is used to retrieve data based on character pattern.  
→ along with like char \& special char there are  
    1) % (Percentage) → string or group of character.  
    2) \_ (underscore) → single character

Syntax:-

Select \* from tablename where colname like  
'character pattern';

Q: write a query to display the emp whose ename start with capital letter '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 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 func?
- 2) userdefined func?

## Predefined functions :~

The Predefined functions are :

- 1) Number functions
- 2) Characters functions
- 3) Date functions.
- 4) Group functions (or) Aggregate funn.

## Number function :~

These funn operate number data.

### 1) abs() :~

→ These funn is used to convert -ve values into +ve values.

e.g:- select abs (-50) from dual;  
O/P :- 50.

### → Dual :~

Virtual

→ Dual is a Predefined table, which is used for testing the predefined, <sup>user-defined</sup> funn(). which contains only one row & one column.

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

e.g:- This table also used to perform mathematical operations.

say select 50+90 from dual;

O/P :- 140

→ By default <sup>Table</sup> Dual has 1 column DUMMY + VARCHAR<sub>2</sub>  
~~say select \* from dual;~~  
X  
~~say desc dual;~~

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

2) Mod (m,n) :-

→ it returns remainder of the  $(m/n)$  remainder.

eg:- select mod (10,5) from dual.  
O/P :- 0.

3) Round (m,n) :-

→ it rounds off 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

$$\begin{array}{r} 1.234\cancel{5}6 , 3 \\ \cancel{1}00 \\ 56 \text{ above } 50\% \text{ add } 1 \\ 1.234 \\ \hline 1.235 \end{array}$$

N:B

Round always checks remaining no. if remaining 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 → e.g. 1280

$$\begin{array}{r} + 1 \\ \hline 1290 \end{array}$$

$$1285.356 , -1$$

5 → 0

above 50%

$$\begin{array}{r} 1280 \\ \hline 1 \end{array}$$

$$\begin{array}{r} + 1 \\ \hline 1290 \end{array}$$

Q) say select round(1295,-2) from deal.

O/P:- 1300

Step 1: 1295

Step 2: 95----00      95 → 0  
95 ---- above 50%      5 → 50% above

1295                    1290

Step 3: 1290

1  
—  
1300 - 0

1290  
+ 1  
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 float value no. ~~exp m~~, based on 'n'. Select trunc.

Say select trunc(1.8) from deal.

O/P:- 1

Say select trunc(1.23456,3) from deal.

O/P:- 1.234

5) Greatest(exp1, exp2 ...), least(exp1, exp2 ...):~

→ greatest returns maximum value within given expression whereas least returns minimum values among given expression.

Eg:- Select greatest(4,7,9) from deal;

O/P:- 9

→ If we pass only one column it doesn't work. It Passes one or more columns required.

Select ename, sal, comm, greatest(sal, comm) from emp where column 'comm' is not null.

01/02/14

## Ques

ceil(), floor():~

→ ceil date nearest greatest integer where as  
floor returns nearest lowest integer.

e.g:- Select ceil (1.4) from dceaf.

O/P - 2.

- say select floor (1.9) from dceaf;

O/P - 1

characters functions :~

1) upper() :-

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

e.g:- say select upper('abcd') from dceaf;

O/P :- ABCD

← column name.

say select upper(ename) from emp;

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

2) lower() :-

e.g)- say select lower ('ABCD') from dceaf;

O/P :- abcd

say select lower(ename) from emp;

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

→ if want to change my ename all data in column into uppercase to lowercase then:

say update emp set ename = lower(ename);

3) initcap():-

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

~~→ SQL> Select initcap(ename) from emp;~~

$\rightarrow$  SQL> Select initcap('ab cd ef') from dual;

O/P :- Ab cd ef

length() : ~

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

१५

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

O/P:- 6 (AB, CD E)  
1 2 3 4 5 6

5) Substring : ~ substr()

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

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

O/P:- EFGHI

Select substr('ABCDEFGHI', 2, 3) from deaf;

O/P) - BCD

SQL Select substr('ABCDEFGHI', -2,3) from deaf;

O/P :- ~~100~~ HD

say select substr ('ABCDE~~F~~GHIJ', -6, 3) from str;

◎P:- EFG

Syntax :-

`Substr(` *classname* *on stringname* *, starting position* *, number of char from position* `)`

must remember.

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

Say Select \* from emp where substr(ename,2,2) = 'AR';

O/p:- ENAME

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.

Q) instr():

Eg:-

Say Select instr('ABCD', '\*') from dual;

O/p:- 0

Say Select instr('ABD', '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 func 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.

→ e.g. say select instr('AB\*CD\*EFG\*JKLMN\*XYZV', '\*', 5, 2) from dual;

Ans:- 14

SQl) Select insert('ABXcDEFGXKLMNXXZXR'; '\*', -5, 2) from dual;  
O/p :- 3

Say Select instr ('AB \* CDEFG \* KLMN \* ZXV', 'x' - 4, 2) from drop  
 op:- 9

Syntax :-

`instr ( columnname of  
      or  
      stringname,     , 'str', start,  
                        , searching string, number  
                        of  
                        from position  
                        occurrence )`

→ always intrinsic returns position based on 3rd, 4th & 5th parameters and also Oracle Server counts no. of char in left side 1st position onwards.

माता

Lpad():~ (Lebt Padding)

02/02/14

→ It will fill remaining spaces with this specified character on left side.

eg: Syntax) -

`Upad (c# name or  
'strongname', total length, 'specified char')`

e.g)- Select Upad('ABCD', 10, '\*') from Dref;

⑨/P? - \* \* \* \* \* ABCD.

RPad() :~ (Right padding).

22 Select

e.g.:- Select  $k_{PAQ}('ABC0', 10, \#)$  from deaf;

Op: A B C D # # # # #

→ syntax and defn same as lang.

Say select Rpad('Ename, 30, :') from emp;

Say 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 set of chars)  
Stringname

Say select Ltrim('SSMISSThes', 's') from Dual;

O/P:- MISSThes → SSMISSThes.

Say select job, LTrim(job, 'c') from emp;

O/P:- LERK → clerk

Say select job, LTrim(job, ('cm')) from emp;

O/P:- LERK → clerk ↗ (either C or M)  
Alesman → salesman  
Manager → Manager.

RTrim() :-

e.g. :-

Say select Rtrim('SSMISSThes', 's') from dual;

O/P:- SSMISSThes → SSMISSThes

Trim() :-

Oracle 8.0 introduce trim() function.

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

Syntax:

trim('specified char' from 'Stringname');

Say select trim ('s', from 'MISSTHSS') from deaf;

O/P:- MISSH

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

Say select length(trim(' Smith ')) from deaf;

O/P:- 5      → to check the character length.

Translate(), Replace():-

Eg:- say select translate('JOHN', 'H', 'N') from deaf.

same O/P:- → O/P:- JONN.

say select Replace('JOHN', 'H', 'N') from deaf.

→ O/P:- JONN.

Say select Replace('JOHN', 'H', 'SHKJL0') from deaf

O/P:- JOSHKAJL0N ← it replace one char by string. but not do in a translate.

Say select Translate('JOHN', 'H', 'EWEKLMN') from deaf.

O/P:- JOEN → translate() is translate single char.

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

Syntax of translate():-

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

Eg:- select translate ('A>BEDef', 'FEDCBA', '12345678') from deaf.  
O/P:- 654321.

Eg:- Select Job,Replace (Job, 'Salesman', 'marketing') from deaf;  
O/P:- Salesman → marketing;

Say select Replace ('ISSUESHSS', '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 subtract no. of months to the specified date based on second parameter.

Syntax :-

Add-months ('date', number);

eg:- say select add-months(sysdate, 1) from dual;

O/P:- 03-MAR-14

Say 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 occurrence day from it's specified date based on second parameter.

e.g:- [Select next-day(sysdate, 'sunday') from dual;]

O/P :- 09-FEB-14

→ MONTHS-BETWEEN() :-

Syntax :-

[months-between(date1, date2);]

- This function always returns number datatype.
- It returns no. of months b/w two specified dates.
- Here always date1 > date2, otherwise it returns -ve sign.

e.g:- Select months-between('16-jan-05', '16-aug-05')  
from dual;

O/P :- -7.

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

e.g:- Select round(months-between('26-jan-05',  
'16-aug-05')) from dual;

O/P :- -7

e.g:- Select ename, sal, round(months-between(  
sysdate, hirndate)) from emp;

DATE Arithmetic :-

- Date + number ✓
- Date - number ✓ } Possible
- Date1 + Date2 ✗
- Date1 - Date2 ✗ }

eg:- say Select sysdate + 1 from dref;

sql> Select sysdate - 1 from dref;

sql> Select sysdate - sysdate from dref;

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

eg:-

[say Select add\_months(last\_day(sysdate), -1) + 1

O/P:- 01-FEB-14. from dref;

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 dref;

O/P:- 04/02/2014

NB:-

To-char() is a case sensitive function.

eg:- Select to\_char(sysdate, 'DY') from dref;

O/P:- TUE

Select to\_char(sysdate, 'DAY') from dref;

O/P:- TUESDAY.

DAY → TUESDAY

day → tuesday

D  
 DD  
 DDD  
 MM  
 YY  
 YYYY } ← number      DAY  
 } ← character.  
 DY  
 MONTH  
 MON  
 YEAR

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 dref;

O/P:- Fourth FOURTH

SP → SPELL OUT.

eg:- [select to\_char(sysdate, 'HH:MI:SS') from dref;]

sqlselect to\_char('20-jun-05', 'DD/MONTH/YY') from dref;

O/P:- error.

NB:-

Whenever we are using to\_char, always 1<sup>st</sup> parameter must date type.

to\_date():-

→ It is used to convert char type into date type i.e. it converts date string into date type (oracle date format).

eg:- select to\_date('27/june/04') from dref;

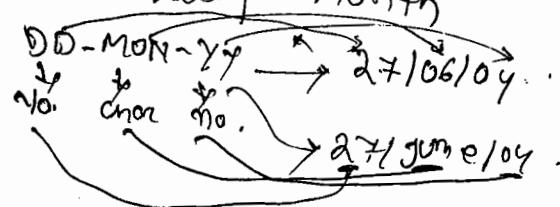
O/P:- 27-JUN-04

→ Oracle date format.

eg:- sqlselect to\_date('27/06/04') from dref;

O/P:- error - not a valid month

beacause:-



~~Select~~

NB:-

when ever we are using to-date parsed 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 into datatype.

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 ('08-FEB-05')+6 from dual;  
O/P :- 14-FEB-05

SQl select to-date ('08-02-05', 'DD-MM-yy')+6  
from dual;

Q: write a query to display the employees whose joining month Dec from emp table, using To-char()  
from emp where  
④ Select 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 function to\_char():

P → 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: whenever we are parsing date string into date()

then oracle server automatically converts date string into date type. But here parsed parameter return value matched with the default date values datatype return value.

eg:-

say select cast-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.

Soln:-

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 truncate() that date can be changed based on time portion.

eg:-

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

O/P:- 05-Feb-09 14:00:00.

→ whenever we are using round function then automatically one day added to the current date and also time portion automatically set to zero, if time portion  $\geq 12:00:00$

Say 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 using truncate function oracle server automatically set to time portion '0' and also refers same date.

Say 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.

By say insert into emp(empno, ename, hiredate) values (1, Some, sysdate);

Say select \* from emp where hiredate = sysdate;

O/P:- no rows selected:

Sol:-

Say select \* from emp where trunc(hiredate) = trunc(sysdate);

Group Function or Aggregate function:-

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

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

max():~

→ it returns a max value within a column.

SQL select max(sal) from emp;

O/P:- 28,000.

SQL select max(hiredate) from emp.

O/P:- 23-MAY-87.

SQL select max(ename) from emp;

O/P:- wacl.

min():~

→ Select min(sal) from emp;

O/P:- 800

→ SQL select min(hiredate) from emp;

O/P:- 17-DEC-80.

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

Avg():~

SQL select avg(sal) from emp;

O/P:- 128.43.

SQL select avg(comm) from emp;

O/P:- 550

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

SQL select avg(nvl(comm,0)) from emp;

O/P:- 157.142857.

Sum():~

SQL select sum(sal) from emp;

O/P:- 29125

COUNT(\*) :-

→ it counts no. of rows in a table, including null values.  
say select count(\*) from emp;

O/P:- 14

COUNT(columnname) :-

→ Select count(emppno) from emp;

O/P:- 14

→ it counts no. of nonnull 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

group by:-

This ~~target~~ class is used to arranged 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 using groupby class.

Ans

Q7)

Q: write a query to display no. of employee job wise from emp table. using group by.

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

O/P:- Job      count(\*)

CLERK	4
SALESMAN	4
PRESIDENT	1
MANAGER	3
ANALYST	2

SQL) select deptno, max(sal), min(sal) from emp  
group by Job;deptno;

O/P:-

<u>Deptno</u>	<u>max(sal)</u>	<u>min(sal)</u>
30	2850	950
20	3000	800
10	5100	1300

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

e.g:- SQL) select deptno from emp group by deptno.

O/P:- dept no

30
20
10

Rule :-

→ other than group function columns specified after select those columns must be 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
--------	----------	-------

N.B:-

if we are try to display group function with another column then database server return error to overcome this problem we must use group by clause.

Step 1 :-

say select avg(sal) from emp;

avg(sal)

2080.35714

Step 2 :-

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:-~~ display those dept having more than 4 employe from emp table using group by.

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

error:-

SQL :-

say select deptno, count(\*) from emp group by deptno  
 having count(\*) > 4.

O/P:-	<u>Deptno</u>	<u>Count(*)</u>
	30	6
	20	5

Having :-

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

- nows
- Generally if we want to nest in a table, then we are using 'where' as if we want to nest groups after group by then we must be use having clause.
  - generally we are not able to use group functions in where class. but we can also use group fun in having class.

say select deptno, sum(sal) from emp group by deptno  
 having sum(sal) > 9000

<u>Deptno</u>	<u>sum(sal)</u>
30	94000
20	108750

Order by :-

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

- 1) Asc
- 2) desc

→ By default order by class having ascending order.

e.g:- Select sal from emp order by sal;

O/P :- 1000

1500

2450

2850

} lowest to highest.

Syntax:-

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

e.g: Select sal from emp order by sal desc.

Select \* from emp order by ename desc.

N.B:-

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

e.g:- Select \* from emp order by 6 desc.

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

O/P:-

<u>deptno</u>	<u>count(*)</u>
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, group total based on no. of columns then we are using cube.

e.g:-  
SQL) Select deptno, job, sum(sal) from emp  
group by Rollup(deptno, job);

SQL) Select deptno, job, sum(sal), count(x) from emp  
group by cube(deptno, job);

→ want to display ename & sal with total sal.  
SQL) Select ename, sum(sal) from emp group by  
Rollup(ename);

### Joins:-

↳ Joins are used to retrieve 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 gi Joins.

Oracle gi Joins are ansi Joins:-

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

→ In Oracle without using joining condition also without retrieving multiple tables. In this case Oracle server internally uses cross join.

→ Cross join internally works based on cardinal product that why this join returns more duplicate data.

e.g:-

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

gi Joins :-

1) Equi Join / inner join :-

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

↳ Whenever tables contains common col names then we are using equi join.

syntax:-

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

(40) 08/02/04

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

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

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

N.B.- generally to avoid ambiguity feature, we must specify every column along with table-name.

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 aliasname must be different name.

Syntax:-

Table name<sub>1</sub> aliasname, Table name<sub>2</sub> aliasname

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

No: using equijoins we are retrieving matching rows only.

(here deptno 40 does not display if we are 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 ejoin.

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

O/P:-

<u>ename</u>	<u>loc</u>
WARD	CHICAGO
TURNER	"
ALLEN	"
JAMES	"
BLAKE	"
MARTIN	"

N.B: If we want to restrict row after joining condition then we must use ~~joining~~ operation AND operation in 8; joins where as in joins we can also use where clauses.

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

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

O/P :-

<u>DNAME</u>	<u>SUM(SAL)</u>
A/C	9050
RESEARCH	11275
SALES	9600

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

<u>Deptno</u>	<u>Deptname</u>	<u>sum(sal)</u>
10 A/C	A/C	9050
20 Res	Res	11275
30 Sales	Sales	9600

→ Select DNAME, sum(SAL) from emp E, DEPTD  
where E. Deptno = D. deptno group by Dname  
having sum(sal) > 1000.

O/P:

Dname	sum(sal)
Research	11275.

→ Select Dname, sum(sal) from emp e, deptd  
where E. Deptno = D. deptno  
group by Rollup (dname).

/

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

### Outerjoin:-

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

→ generally whenever we are using equijoin we are retrieve matching rows only.

→ If we want to retrieve non matching rows also then we are using join operator (+) with eqijoin. This is called oracle 8i outer-join.

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

### Syntax:-

e.g. 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 self join.
- here joining conditional cols must belongs to same datatype.
- if we want to compare two different col values in a single values then we must use self ~~join~~ join
- here these two col must belongs same datatype.
- If we want to compare single col value also then we must use self join.

EMP	
eno	eno <sub>1</sub>
10	10
20	30
30	40

+ two diff col compare  
in a single col.

EMP	
eno.	
10	
20	
30	

one col value is compare to each other then  
we are use <sup>self</sup> join.

- in database whenever we are referencing a single table more than one time then we must use table aliases in self join or we are referencing a table more than one table that why we must create alias name in form class

These alias name creates memory at the time of execution of the query.

→ These aliasname must be diff' name otherwise DB server returns a ambiguity.

Syntax:-

From tablename aliasname, tablename aliasname<sub>2</sub>

e.g.: say from test a, test b where a.city = b.city and a.name = 'murali'.

say create table test ( name Varchar<sub>2</sub>(10), city Varchar<sub>2</sub>(10)),

say insert into test ('soma', 'BBSR');

say select \* from test;

<u>name</u>	<u>city</u>
soma	hyd
Reema	sbp
Omkar	BBSR

Q: w.A.Q to display the employee who are working in same city as murali, using self join

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

<u>name</u>	<u>city</u>
murali	hyd
zzz	hyd

test.a	
<u>name</u>	<u>city</u>
murali	x

test.b	
<u>name</u>	<u>city</u>
?	x

Join.

say update emp

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

Q From emp e<sub>1</sub>, emp e<sub>2</sub> where e<sub>1</sub>.sal = e<sub>2</sub>.sal  
AND e<sub>1</sub>.name = 'SMITH'

↔ Join

emp e <sub>1</sub>		emp e <sub>2</sub>	
ename	sal	name	sal
SMITH	X	?	n

SQl) Select e<sub>2</sub>.ename, e<sub>2</sub>.sal from emp e<sub>1</sub>,  
emp e<sub>2</sub> where e<sub>1</sub>.sal = e<sub>2</sub>.sal AND e<sub>1</sub>.ename  
= 'SMITH'.

Q: write a query to display employee name &  
their manager names from emp table using  
self join.

Q) SQL) Select e<sub>1</sub>.name "employees", e<sub>2</sub>.name "managers"  
from emp e<sub>1</sub>, emp e<sub>2</sub> where e<sub>1</sub>.mgr = e<sub>2</sub>.  
empno.

or

SQL) Select e<sub>1</sub>.name "employees", e<sub>2</sub>.mgr, e<sub>2</sub>.empno,  
e<sub>2</sub>.ename "managers" from emp e<sub>1</sub>,  
emp e<sub>2</sub> where e<sub>1</sub>.mgr = e<sub>2</sub>.empno.

Q: write to display the employee who are getting  
more than their manager salaries from  
EMP table using self join.

Q) SQL) Select e<sub>1</sub>.ename "employees", e<sub>1</sub>.sal,  
e<sub>2</sub>.sal, e<sub>2</sub>.ename "managers" from emp e<sub>1</sub>,  
emp e<sub>2</sub> where e<sub>1</sub>.mgr = e<sub>2</sub>.empno and  
e<sub>1</sub>.sal > e<sub>2</sub>.sal

1

2

Q: W.A.Q to display the employees who are join before there managers from emp table using self join.

P: 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 employees who are getting same salary in different dept from emp table using same self join.

P: SQL 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: W.A.Q to display the employees who are joining in same month and same day of the week from emp table using self join.

P: 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-equi joins :-

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 doesn't have common col. 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 bet<sup>n</sup> 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 every tables having common col then only we are using inner joins.

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

→ This join performs is very high compare to Oracle 8i inner join equijoin.

SQL select ename, loc, d.deptno, d.dname, loc from emp e join dept d on e.deptno = d.deptno.  
in place of where.

Q: w/AQ to display the employees who are working in a location Chicago from emp, dept tables using q; inner joins.

A) Select ename, loc from ~~emp~~<sup>join</sup> dept on emp.  
deptno = dept.deptno ~~where~~<sup>AND</sup>, loc = 'CHICAGO'.

\* → in q; in place of 'and' we can use 'where', but in SQL it will not work.

O/P:-	<u>ename</u>	<u>loc</u>
	WARD	CHICAGO
	TURNER	CHICAGO
	ALLEN	CHICAGO

NB:- we can also use 'using' clause in place of 'on' clause in q; joins.

generally 'using' clauses performance is very high compare to 'on' clause.

e.g:-

SQL Select ename, loc from emp join dept  
on using(deptno) where loc = 'CHICAGO'.

Q: NB:- Always 'using' clauses returns common col's one time only. 11/02/14

e.g:- Select \* from t<sub>1</sub>;

A	B	C
X	Y	Z

Select \* from t<sub>2</sub>;

A	B	C
X	Y	Z

say select  $x$  from  $t_1$  join  $t_2 \rightarrow$  using  
on  $t_1.a = t_2.a$  and  $t_1.b = t_2.b \}$  join & on.

1

O/P:-  $\begin{array}{ccc} \underline{A} & \underline{B} & \underline{C} \\ x & y & z \\ \underline{x} & \underline{y} & \end{array}$

say select  $x$  from  $t_1$  join  $t_2 \} \text{ using}$   
 $\text{using}(a,b); \} \text{ using clause}$

1

O/P:-  $\begin{array}{ccc} \underline{A} & \underline{B} & \underline{C} \\ x & y & z \end{array}$

N:B (Rule)

when ever we are using 'using' clause we are  
not allowed to use alias name on joining  
conditional col.

eg:- say select ename, sal, deptno, dname, loc  
from emp & join dept of using(deptno)  
left outer join:-

left outer 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  $x$  from  $t_1$ :

O/P.  $\begin{array}{ccc} \underline{A} & \underline{B} & \underline{C} \\ x & y & z \\ p & q & r \end{array}$

Select \* from t<sub>2</sub>;

o/p:- A B  
x y  
s t

sql) select \* from t<sub>1</sub> left outer join t<sub>2</sub>  
on t<sub>1</sub>.a = t<sub>2</sub>.a and t<sub>1</sub>.b = t<sub>2</sub>.b;  
/

%:- A B C  
x y z  
P Q R  
} left side all data are shown.

A B  
x y ? → matching row are only shown.  
? ? ↳ non matching value are null value displayed.

Right outer join:-

e.g:- sql) select

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

e.g:- sql) select \* from t<sub>1</sub> right outer join t<sub>2</sub>  
on t<sub>1</sub>.a = t<sub>2</sub>.a and t<sub>1</sub>.b = t<sub>2</sub>.b;  
/

%:- A B C      A B  
x y z      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.

e.g:- SQL select \* from t<sub>1</sub> full outer join t<sub>2</sub>  
on t<sub>1</sub>.a = t<sub>2</sub>.a and t<sub>2</sub>.a = t<sub>2</sub>.b;

Op:-	A	B	C	A	B
	x	y	z	x	y
	p	q	r	q	q
	t	o	o	s	t

Natural Join :-

→ This ~~join~~ also returns matching rows only.

This join performance is very high compare to inner join.

→ when tables having common col. then only we allowed to use 'natural join', whenever we specify the natural join internally DB server only establishes joining condition that way explicitly we are not allowed to use joining condition.

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

e.g:- Select \* from t<sub>1</sub> natural join t<sub>2</sub>;

Op:-	A	B	C
	x	y	z

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

Cross join:-

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

join more than two tables

e.g:- (Join 3 tables)

### 8: Joins

Syntax:-

```
Select col1, col2, col3, ...  
from tablename1, tablename2,  
tablename3  
where tablename1.commoncol  
= tablename2.commoncol
```

AND

```
tablename2.commoncol  
= tablename3.commoncol;
```

### 9: Joins

Syntax:-

```
Select col1, col2, col3, ...  
from tablename1 join tablename2  
on tablename1.commoncol  
= tablename2.commoncol  
join tablename3 on  
tablename2.commoncol  
= tablename3.commoncol;
```

## CONSTRAINTS :-

Constraints are used to prevent 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 Business Rules.
- If we want to defined these business rules all DB system uses constraints, triggers.
- Oracle server having following constraints is :
  - 1) Not null,
  - 2) unique.
  - 3) Primary key.
  - 4) Foreign key.
  - 5) check.

Now All the above constraint are defined two ~~levels~~ <sup>levels</sup>

1) column level

2) Table level

Column Level :-

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)  
constrainttype, 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), ..., constrainttype (col1, col2 ...));
```

## NOT NULL constraint :-

→ it doesn't support table level.

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

## column level :-

→ say create table x1(sno number(10) notnull,  
name varchar(10));  
Table created.

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

Error: 1400 : cannot insert null into sno.

## Unique constraint :-

→ Unique constraint refers defining col level or table level.

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

→ whenever we are creating any constraints Oracle server automatically creates three indexes on those columns.

col level :-

say create table tablename  $x_2$  (sno number(10)  
unique, name varchar(10));

Table level :-

say create table tablename  $x_3$  (sno number(10),  
name varchar(10), unique (sno, name));

say insert into  $x_3$  values (1, 'murali');  
1 row inserted.

say insert into  $x_3$  values (1, 'abc');  
inserted.

say insert into  $x_3$  values (1, 'abc');

error:- 00001 : unique constraint violated.

say select \* from  $x_3$

sno	name
1	murali
1	abc

Primary key :-

→ Primary key uniquely identifies record in a table. where there can be only one primary key in a table and also primary key does not accept duplicate, null values.

→ whenever we are creating a primary key, internally oracle server creates btree index on those columns.

column level :-

syntax:-

create table tablename  $x_4$  (sno number(10) primary key,  
name varchar(10));

### Table level :-

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

→ Table created;

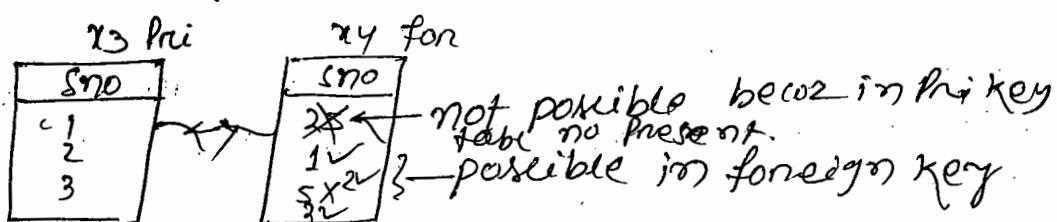
→ This is also called as composite Primary key i.e. It is a combination of cols as single Primary key.

### Foreign key :-

→ If we want to establish relationship between tables then we must use referential integrity constraint. 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 accept duplicate, null-values whereas foreign key accepts duplicate, null values.

### Col level :- (References)

→ Syntax :-

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

sql> create table z<sub>4</sub> ( sno number(10) references x<sub>4</sub> );  
Table created. (OK)

sql> create table y<sub>4</sub> ( a number(10) references x<sub>4</sub>  
( sno ));  
Table level :- ( foreign key, references )  
syntax:-

```
create table tablename ( col1 datatype(size),  
col2 datatype (size), ..., foreign key (col1,  
col2) references mastertablename (primarykey  
columnnames));
```

sql> create table z<sub>5</sub> ( sno number(10), name from  
Varchar<sub>2</sub>(10), col<sub>3</sub> number(10), foreignkey  
(sno, name) references x<sub>5</sub> )

Mac  
?

15/02/14

whenever we are establishing relationship  
between table Oracle violates following two  
rules:

- 1) Deletion in Master.
  - 2) insertion in child.
- 3) 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 <sup>corresponding in</sup> Master-table record in it.

Otherwise use an on delete cascade clause.

On delete cascade :-

Whenever we are using 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) references  
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 (Primarykeyof
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 define logical conditions according to our business rules.

Column level :-

Syntax :-

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

eg:-  
create table test (sal number(10) check(sal > 5000));  
Table created.

eg:-  
create table test1 (name varchar(10) check(name = upper(name)));

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

Error: check constraint (SCOTT.SYS-005489) violated.

Table level :-

→ Predefined constraint name.

SQL> create table test2 (name varchar(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 form of sys-# this is called predefined constraint name.

→ We can also create our own name in place of sys-# using constraint keyword.

→ This is called user defined constraint name  
syntax:

Constraint userdefinedname constraintType

↳ constraint  
name

↳ notnull  
Primary k  
foreign  
unique  
check

e.g:-  
say create table tests ( sno number(10),  
Primary key);

sql> insert into tests values(1);

sql> insert into tests values(1);

error: unique constraint (SCOTT.sys-  
constraint violated.)

e.g:- say create table table6 ( sno number(10)  
constraint P-abc Primary key);  
↳ user defined constraint name

sql> insert into tests values(1);

sql> insert into tests values(1);

error: unique constraint (SCOTT.P-ABC)  
violated.

Note

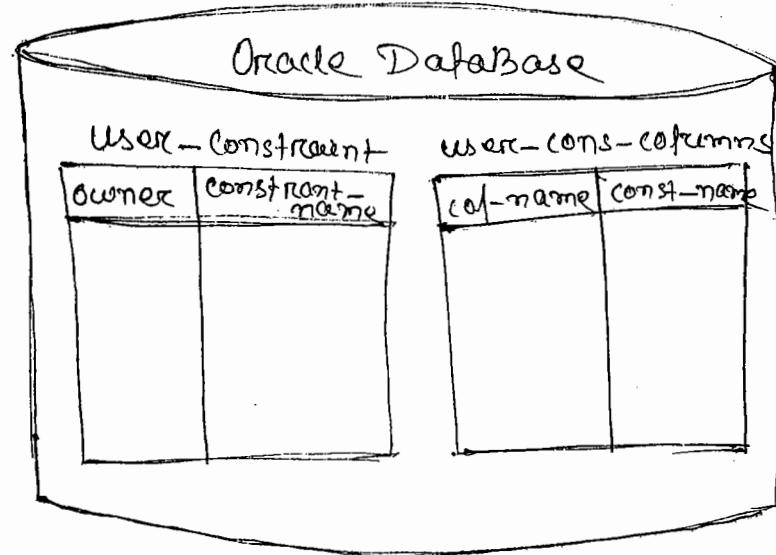
14/02/14

when ever we are installing oracle server automatically readyonly tables are created in db.  
These tables are also called as data dictionaries.

→ (After diagram) diagram (1.1)

→ All constraint information stored under  
user - constraint data dictionary

sql> 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  
P<sub>K</sub>-EMP  
F<sub>K</sub>-DEPTNO

constraint-type  
P  
R

NB:- If we want to view col names along with constraints name then we are using 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/R:- constraint-name  
F<sub>K</sub>-DEPTNO  
P<sub>K</sub>-EMP

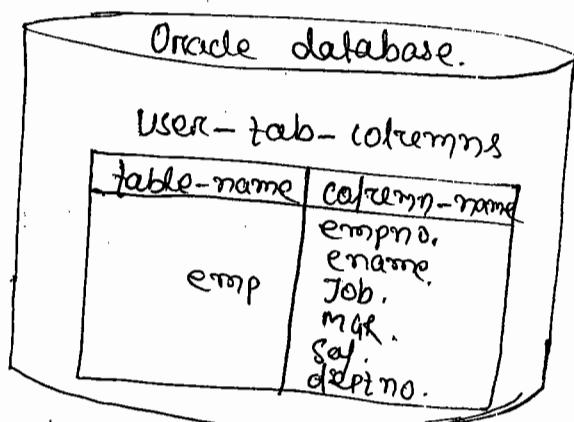
column-name  
DEPT NO  
EMPNO

NB:- If we want to view logical condition of the check constraint then we are using Search-condition Properties from user\_constraint Data dictionary.

sql> 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.



e.g:- sql> desc user-tab-columns;

sql> Select column-name from user-tab-columns  
where table-name = 'EMP';

Q:- sql> select count(\*) from

\* W.A.Q to display no. of columns number from emp table.

sql> select count(\*) from user-tab-columns  
where table-name = 'EMP';

Ans

15/02/13

Default constraint :-

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

Syntax:-

colname datatype(size) default actualvalue;

sql> create table test(name varchar(10), sal number(10) default 2000);

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

sql) select \* from test;

name      say

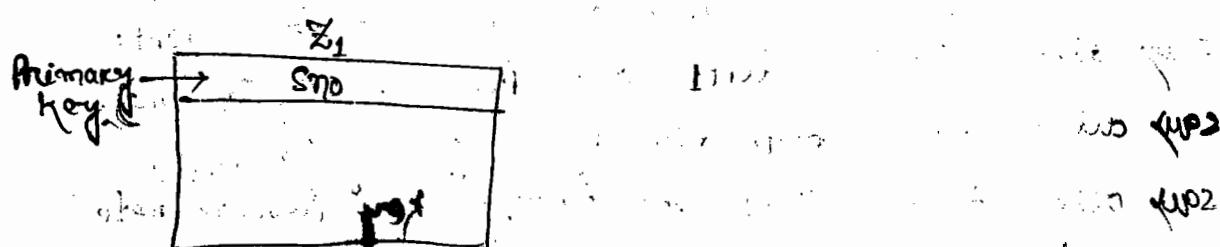
N.B:- if we want to view default values for a col then we are using, data-default Properties from user-tab-cols data dictionary.

eg:- sql) select column-name, data-default from user-tab-columns where table-name = 'TEST';

O/P:-    column-name      data-default  
say                          2000

Adding and dropping constraints on existing tables  
using alter we can also add and drop constraint on existing table and existing column.

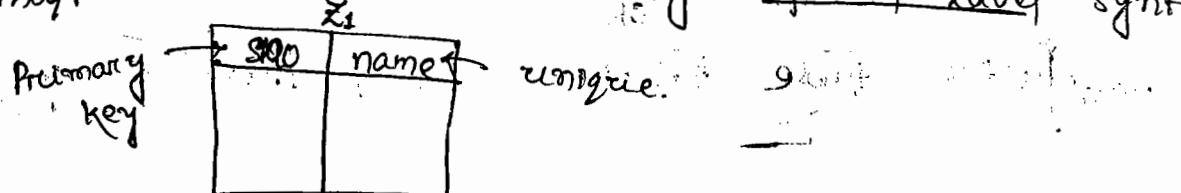
N.B if we want to add constraint of existing table existing column then we are using table level syntax method.



sql) create table  $Z_1$  (sno number(10));

sql) alter table  $Z_1$  add Primary key (sno);

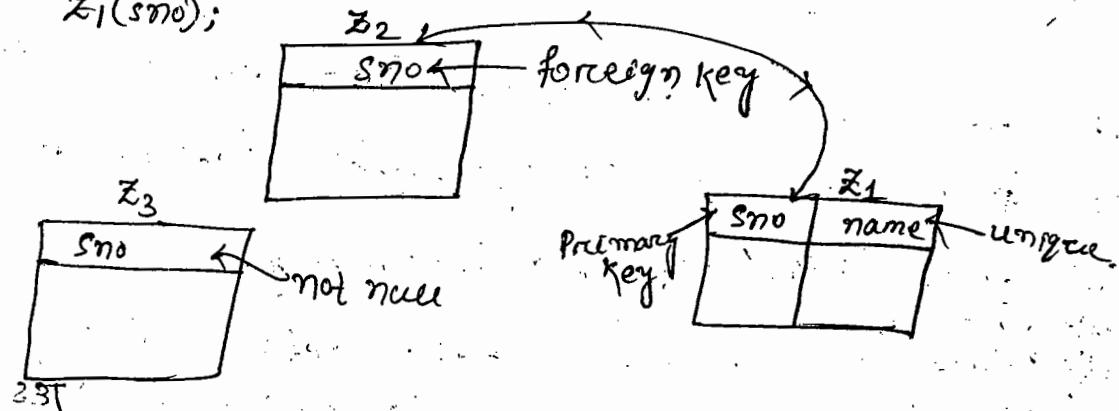
N.B:- if we want to add a new column along with constraint then we are using column level syntax.



SQL alter table Z1 add name varchar(10) unique;

SQL create table Z2 (Sno number(10));

SQL alter table Z2 add foreign key (Sno) references Z1(Sno);



Ques:- if we want to add not null constraint of existing table and exisitng column then we can alter with modify.

SQL alter table tablename modify column not null;

e.g:- alter table Z3 (sno number(10)) add

SQL alter table Z3 modify sno not null;

SQL alter table Z3 add new varchar(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;

Method-2 alter table tablename drop Primary key;

(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 using cascade  
clauses 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;

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

sql) alter table w1 drop constraint ab;

Subqueries:-

- Query with another query is called nested query or subquery.
- generally child queries are also called subqueries.
- These queries are used to retrieve data from single or multiple table based on more than one step process.
- All DB system having two types of subqueries
  - 1) Non-correlated subqueries.
  - 2) Correlated subqueries.
- In non-correlated subquery child query is executed 1st, then only parent query is executed.
- whereas in correlated subqueries parent query 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.

## Non-correlated subqueries

### Non-correlated subqueries

single row  
subquery

multiple row  
subquery

multiple column  
subquery.  $\downarrow$   
In Line  
view  
(OR)

subquery used  
in (from clause),  
who are

\* W.A.Q to display the employee getting more than the average salary from emp table.

{ Select empname, empsal from emp where  
sal > avg(sal); } wrong,

Blunder mistake

{ 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 using  $<=$ ,  $<$ ,  $>$ ,  $\geq$ , operators.

### Execution

Step 1: say select avg(sal) from emp;

Step 2: say select \* from emp where sal > 2216.0743;

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 where deptno = (select deptno from emp 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 dname = 'SMITH');

Q: Q.W.W.A.Q to display innermost employee details from emp table.

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

W.A.Q to display the employee who are working under BLAKE from emp table using empno, mgr columns.

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

Q: W.A.Q to display lowest avg. sal taken 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);

Note: Nested group by function without group by.

When a child query contains nested group functions then we must use group by clause.

\* SQL: 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');

<u>Job</u>	<u>Avg(Sal)</u>
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);

<u>Deptno</u>	<u>Min(sal)</u>
20	1100
10	2650

Q. ~~How~~ 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);

→ mention Single-row subquery (referencing more than one row).

\* This is a multiple row subquery becoz 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);

<u>DEPTNO</u>	<u>SAL</u>	<u>ENAME</u>
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 department from emp,  
 dept table.

Ans Select \* from emp where deptno in (select deptno  
 from dept where dname = 'SALES' or dname = 'RESEARCH');

TOP-N analysis :-

19/02/14

↳ inline views

↳ rownum

inline views :-

Oracle 7.2 introduced inline views.

→ Generally we are not allowed to use order by clause  
 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 behaves 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.

e.g.: - SQL Select rownum, ename from emp;

<u>Rownum</u>	<u>ENAME</u>
1	SMITH
2	ALLEN
3	WARD
4	JONES

These are  
Temporary  
values

→ Rownum having temporary values.

example:- SQL Select rownum, ename from emp where deptno >= 10;

<u>Rownum</u>	<u>ENAME</u>
1	CLARK
2	MILLER

Q: W.A.Q to display 1<sup>st</sup> row from emp table using rownum.

Ans:- SQL Select \* from emp where rownum=1;

Ans:- SQL Select \* from emp where rownum=2;

Q: W.A.Q to display 2<sup>nd</sup> row from emp table using rownum.

N.B: Generally rownum does not work with more than one two integers i.e it works with <, >, = operators.

Q: W.A.Q to display 1<sup>st</sup> 5 rows from emp table using rownum.

Ans:- SQL Select \* from emp where rownum <= 5;

Q: W.A.Q to display 5<sup>th</sup> highest sal employee from emp table using rownum.

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

Q: W.A.Q to display 5<sup>th</sup> highest salary employee from emp table using 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 using rownum.

A) Select \* from emp where rownum <= 2  
Minus

Select \* from emp where rownum <= 1.

say select \* from emp, <sup>where</sup> rownum between 1 and 5;  
working.

say select \* from emp where rownum between 2 and 5;  
not working.

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 1 to 5.

25/02/13

Q: w<sup>A</sup>Q to display the rows between 3 and 9 from emp table using rownum.

Ans Select \* from emp where rownum <= 9  
minus

Select \* from emp where rownum <= 3;

Q: w<sup>A</sup>Q to display last two rows from emp table using rownum.

Ans Select \* from emp where rownum  
minus

Select \* from emp where rownum <= (Select count(\*) - 2 from emp);

N.B.:~ when we are creating a reference name for rownum in inline resq then that reference name works with all sql operators.

Q: w<sup>A</sup>Q to display 5<sup>th</sup> row from emp table using rownum alias name.

Ans Select \* from (Select rownum n, ename, sal from emp) where n=5;

Rename	1	SAL
5   MARTIN		1150

(OR)

SQL Select \* from (select rownum n, e. \* from emp) where n=5;

Q: w<sup>A</sup>Q to display 1<sup>st</sup> row, last row from emp table using rownum alias name.

Ans SQL Select \* from (select rownum n, ename, sal from emp) where n=1 OR n (Select (X) 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 using rownum alias name.

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

Q: W.A.Q to display 5th highest salary employee from emp using rownum alias name.

Q: Select \* from (select \* rownum r, ename, sal from (select \* from emp order by sal desc)) where r=5; (8n),  
use when it need  
1, 2, 3 ...

O/P:- R ENAME SAL  
5 MILLER 2700

Analytical functions used in inline views in Oracle 8i introduce analytical function these are:-

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

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

Syntax:- Analytical functionname() over  
(Partition by colname order by colname [asc/dsc])  
Optional Part

→ row-number() analytical function automatically aligns different rank numbers when values are same.  
 whereas rank, dense\_rank() analytical funn automatically aligns same rank nos when values are same.  
 But rank skips next consecutive to rank number.  
 whereas dense\_rank() does not skip next consecutive to next rank number.

e.g:- row-number() :-

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

Rank() :-

<u>Dept</u>	<u>Ename</u>	<u>sal</u>	<u>R</u>
30	BLAKE	1150	1
30	TURNER	1150	2
30	WARD	1000	3

✓ good. Dense\_rank() :-

<u>Dept</u>	<u>Ename</u>	<u>sal</u>	<u>R</u>
30	BLAKE	1150	1
30	TURNER	1150	1
30	WARD	1000	2

→ Select \* from (Select deptno, ename, sal, row-number())

over ( Partition by deptno order by sal desc)

" from emp) where R <= 10;

(alias name)

Q: W.A.Q to display second highest salary employee in each dept from emp table using analytical functions.

A) > select \* from (select deptno, ename, sal, dense\_rank() over(partition by deptno order by sal desc) r from emp) where r=2;

O/P:-

<u>Deptno</u>	<u>ename</u>	<u>sal</u>	<u>R</u>
10	MILLER	2700	2
20	JONES	2975	2
30	TURNER	1400	2

Q: W.A.Q to display 5<sup>th</sup> highest salary employee from emp table using analytical function (dense\_rank).

A) Select \* from (select deptno, ename, sal, dense\_rank() over(partition by ~~deptno~~ order by sal desc) r from emp) where r≤5;

O/P:-

<u>Deptno</u>	<u>ename</u>	<u>sal</u>	<u>R</u>
10	MILLER	2700	2

Ans:

Some like a normal function.

Analytical functions also accept parameters.

e.g:- lead():-

This function computes an expression on next rows and return values to the current row.

Syntax:-

lead(<sup>↑</sup>expression, offset, default) over(partitions  
(means of name) by colname order by colname [asc/desc])  
numbers

25/02/14

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

O/P:-

>Select deptno, ename, sal, lead(sal, 1, 0)  
over (partition by deptno order by ename) n  
from emp.

lag:

This function computes an expression 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) n  
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 hired from emp table.

Q) > Select to\_char(hiredate, 'yyyy') "year", count(\*)  
from emp group by to\_char(hiredate, 'yyyy')  
having count(\*) > 1

year	count(*)
1987	2
1988	
1989	8
1990	

Q: W.A.Q to display duplicate data from a table.

→ create table test (sno number(10));

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

I/P:-		O/P:-	
sno	count(*)	sno	count(*)
30	3	30	3
20	2	20	2
40	1		10
50	1		4
10	4		

{ ← Duplicate data.

→ Select sno, count(\*) from test group by sno.

O/P:-

Rowid :-

Q: ~~W.A.Q~~ Rowid is pseudo col. it behaves 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 id 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;

N.B:- We can also use Max, Min functions to rowid.

SQL> Select max(rowid) from emp;

O/P:- rowid

AAA MAZ AAAA g AAA.

SQL> Select min(rowid) from emp;

→ Generally rowid are used to delete duplicate row except one row in each group.

\* Q: w.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 subqueries.

→ in this type of queries we must specify Parent query where conditional col with in parentheses.

Syntax:-

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

Q: W.A.Q to display the employees whose Job, MGR match with in Job, MGR of employee select from emp tables.

> Select \* from emp where (job, MGR) in (select Job, MGR from emp where ename='SCOTT');

O/P:-

<u>EMPNO</u>	<u>ENAME</u>
7788	

Q: W.A.Q to display the employee who are getting max sal from emp in each dept from emp table using multiple rows subquery.

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

O/P:-

<u>deptno</u>	<u>sal</u>	<u>ename</u>
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 using multiple col subquery.

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

<u>deptno</u>	<u>sal</u>	<u>ename</u>
30	2750	BLAKE
20	3000	SCOTT
10	5000	KING

Q: W.A.Q to display e\_name, d\_name, salary of the employee who salary, commission match with the sal, commission with the employees working in the location "Dallas" using emp, dept table.

A) Select e\_name, d\_name, sal from emp where deptno where e.deptno = d.deptno and (sal, comm)  
 in (select sal, comm from emp, dept where e.deptno = d.deptno and d.loc = 'DALLAS')

↳ Q: W.A.Q to display the employees who are getting more than the highest 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	e_name	job	MGR	sal	comm	deptno
7838	King	clerk	25-Feb-2000	5000		20

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

A) Select \* from emp where sal < (select min(sal) from emp where deptno=10);

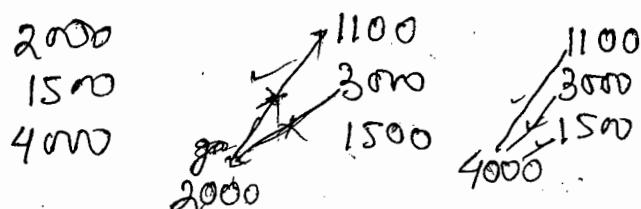
↳ whenever resource table large amount of data & also when we are comparing no.of values & also if child query contains max or min functions then those type of subqueries degrades performance of the application. To overcome this problem all database provided sumgrouper special operators to improve performance of the query these query

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

- 1) Select \* from emp where  $\text{sal} \geq \text{all}$  (select sal  
from emp where deptno = 20);
- 2) Select \* from emp where  $\text{sal} \geq \text{any}$  (select sal  
from emp where deptno = 10);

'all', 'any' operators used in multiple row subquery  
IN → it references same values in the list → child query.

All → it satisfies all values in the list.



Any → it satisfies any value in the list.

IN → ANY.

↳ when ever we are using subquery 'any' optimizer internally uses logical operator AND also we are using subquery special operator 'any' optimizer internally uses logical operator (or).

- ↳ 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 IN  
not same ↳ ANY.

Say select \* from emp where deptno = any (10, 20).  
Co-related subqueries :-

generally in noncorrelated subqueries child query  
ie executed first then only parent query  
executed.

whereas a correlated subquery parent query is executed 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 clause.

Syntax:-

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

e.g:- 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 per parent table whereas in correlated subqueries child query is executed for each and every row in parent query table.

27/02/14

generally correlated subqueries are used in denormalization process.

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

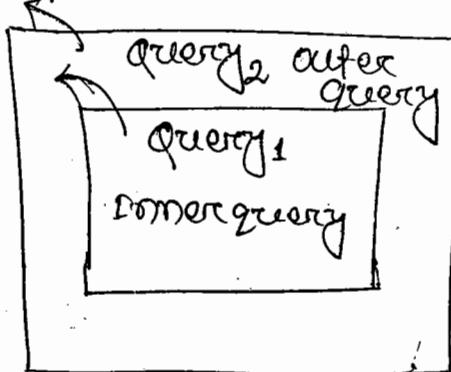
Syntax:-

```
update tablename, aliasname  
set colname = (select colname from tablename  
aliasname2 where aliasname1 .  
common colname = aliasname2 .  
common colname .
```

- > alter table emp add dname varchar(10);
- > update emp e set dname = (select dname from dept d where e.deptno = d.deptno);
- > select \* from emp;

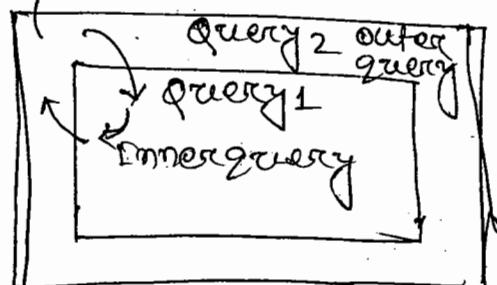
Noncorrelated subquery

Final output.



Correlated subquery

Final output.



> Q: w.A.Q to display 1st highest salary employee from emp table using correlated subquery.

> Select \* from emp e<sub>2</sub> where e<sub>2</sub>.sal = (select coent(\*) from emp e<sub>1</sub> where e<sub>2</sub>.sal  $\geq$  e<sub>1</sub>.sal);

O/P:-

Kong 50,000.

Phase 1:

Step by step: get a candidate row (first row) →

Q:- w.a.q to display 2<sup>nd</sup> highest sal employee from the following table using correlated subquery.

> Create table emp1(ename varchar(10), sal Number(10));

> Select \* from emp1;

<u>ename</u>	<u>sal</u>
abc	100
xyz	150
zzz	200
bbb	300

> Select \* from emp1 e<sub>1</sub> where 2 = (Select count(\*) from emp1 e<sub>2</sub> where e<sub>2</sub>.sal >= e<sub>1</sub>.sal),

<u>ename</u>	<u>sal</u>
zzz	200

Execution :-

SQL Select \* from emp1 e<sub>1</sub> where 2 = (Select count(\*) from emp1 e<sub>2</sub> where e<sub>2</sub>.sal >= e<sub>1</sub>.sal);

Phase 1 :-

Step 1 :- get a candidate row (first row) (abc 100)

Step 2 :- Select count(\*) from emp1 e<sub>2</sub> where e<sub>2</sub>.sal >= 100

Step 3 :- Select \* from emp1 e<sub>1</sub> where 2 = 4 (false),

Phase 2 :-

Step 1 :- get a candidate row (xyz 150)

Step 2 :- Select count(\*) from emp1 e<sub>2</sub> where e<sub>2</sub>.sal >= 150

Step 3 :- Select \* from emp1 e<sub>1</sub> where 2 = 3 (false)

Phase 3 :-

Step 1 :- get a candidate row (zzz 200)

Step 2 :- Select count(\*) from emp1 e<sub>2</sub> where e<sub>2</sub>.sal >= 200

Step 3 :- Select \* from emp1 e<sub>1</sub> where 2 = 2 (true).

zzz 200 ← O/P.

- > insert into emp<sub>1</sub> values ('soma', 2000);  
insert
- > Select \* from emp<sub>1</sub> e<sub>1</sub> where 2 = (select count(x) from emp<sub>1</sub> e<sub>2</sub> where e<sub>2</sub>.sal > e<sub>1</sub>.sal);
- O/P:- no row selected.

Soln:-

- > Select \* from emp<sub>1</sub> e<sub>1</sub> where 2 = (select count  
(distinct(sal)) from emp<sub>1</sub> e<sub>2</sub> where e<sub>2</sub>.sal > e<sub>1</sub>.sal)

O/P:-      Ename            sal

zzz	200
soma	200

n-1 method:

- > Select \* from emp<sub>1</sub> e<sub>1</sub> where  $\frac{n}{(1-1)}$  = select count  
(distinct(sal)) from emp<sub>1</sub> e<sub>2</sub> where e<sub>2</sub>.sal > e<sub>1</sub>.sal);

N.B.

In case of n-1 method we can't use ' $=$ ' or ' $\neq$ ',  
only we can use ' $<$ ' & ' $>$ ' depending on required  
query.

Q: W.A.Q to display nth highest sal employee from  
emp table using correlated subquery.

- > Select \* from emp<sub>1</sub> e<sub>1</sub> where n = (select count  
(distinct(sal)) from emp<sub>1</sub> e<sub>2</sub> where e<sub>2</sub>.sal > e<sub>1</sub>.sal);

O/P:- Enter value for n: 1

King        5000

Q: W.A.Q to display the employees who are getting  
more than the avg salary of their job from  
emp table using correlated subquery.

- > Select \* from emp<sub>1</sub> e<sub>1</sub> where sal > (select avg(sal)  
from emp<sub>1</sub> e<sub>2</sub> where job = e<sub>1</sub>.job);

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 whether a given set is ~~is not~~ 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 using ~~any~~ operator, operators that we are not allow to use of cname in where condition.

Syntax:-

Select \* from tablename aliasname where exists  
(Select \* from tablename where cname = aliasname  
cname).

Q: W.A.Q to display departments from dept table having employee in emp table using 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: Q.A.Q to display those dept from dept table does not have employee in emp table using non correlated subqueries.

-select \* from dept where deptno not in (select deptno from emp);

<u>DeptNo</u>	<u>DName</u>	<u>Loc</u>
40	operations	BOSTON

→ Generally "not in" operator does not work with null values. To overcome this problem we must use not exists operator in correlated subqueries.

→ insert into emp(empno, deptno) values (1, null);

→ Select \* from dept where deptno not in (select deptno from emp);

→ If no row selected,

→ select \* from dept d where deptno not exists 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 cursor of a table.

→ Generally views are created by database administrator.

→ In all dbs if you want to restrict 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 having 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 whereas complex view is a view

which is created from no. of base tables.

QUESTION

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 definitions 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 varchar(10)),

create or replace view v;

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 some other table col are not affected.

- To overcome this problem, oracle provides "instead  
of trigger" ProjDefault "instead of triggers" are now  
level triggers and also instead of triggers are  
created on views.

Q in oracle if we want to view modifiable or  
non modifiable col then we are using  
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.
  - ↳ Statement level
  - ↳ Row level.
- ↳ In statement level triggers, trigger body is executed only once per DML statements, whereas 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

trigger timings

trigger events

trigger specifica

[for each row]

begin

  {

  } trigger body

end }

Diff bet<sup>n</sup> statement level trigger and row level trigger.

- Create table test (col1, Date);

Statement level trigger :-

Create or replace trigger tr  
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;

col1

03-MAR-14

Row level trigger  
~~in~~

- create or replace trigger t12  
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 t12

mag

05/03/14

Materialized views :-

fast refresh materialized views :-

These type of materialized view are also called as incremental fast refresh

These materialized view performance is very high compare to complete refresh materialized views becoz in past refresh materialized views row additions are not changed when we are refreshing materialized views no. of times also.

Syntax:-

```
create materialized view viewname  
refresh fast as  
select statement;
```

Before we are creating fast refresh materialized view database administrator must create materialized view log on base tables.

Syntax:-

```
create materialized view log on basetablename;
```

sql) create materialized view log on base;  
% created.

sql) create materialized view mvs  
refresh fast  
as select \* from base;  
% view created.

sql) select rowid, sno, sname from mvs;

O/P:-

RowID	SNO	Name
AAAIDAAA8AA	1	A
11 8AB	2	B
8AC	3	C
8AD	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 mvs; (Here rowid not changed).  
on demand / or commit :~

> in oracle we can refresh materialized view used in two ways.

- 1) automatically.
- 2) manually.

manually :~

in manual method we are using DBMS\_MVIEW package for Mviews package refreshing materialized views.

> This method is also called as ondemand method.

> in oracle by default method is ondemand.

Automatically :~

we can also refresh materialized views without using DBMS\_MVIEW Package. This method is also called as on commit method.

Syntax:-

create materialized view viewname

refresh complete/refresh fast on demand / on commit  
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	A
2	B
3	C
4	D

sql> update base set name = 'zz' 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 (cancel 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

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 exist.

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

↳ System Privileges:-

↳ System privileges are created by database  
Administrator only.

↳ Before we are creating database object

Database Administrator giving permission to create  
database object to system privileges.

Oracle having more than 80 system privileges, these are

create table, create session, create any materialized view, create any view, create procedure, create any index ... ,

Syntax:-

grant system privileges to username, ...;

SQL conn sys as sysdba,

Enter password : sys

SQL grant create table, create Procedure, create any materialized view to scott, murali, inida,

Scotta JSP

Region Req

Resp -&-

out J

Properties A JSP action

context A Application = new A object : Req App Page

out

Session scottka == a)

if (scottka == a)  
return true  
else

return false

1 w JSP: include

at. Str s=

equal

(1 and 2)

Req Dis

SendRecdRecd

(2)

Instance of

if (obj instance of String)

return true

if (s Inst String)

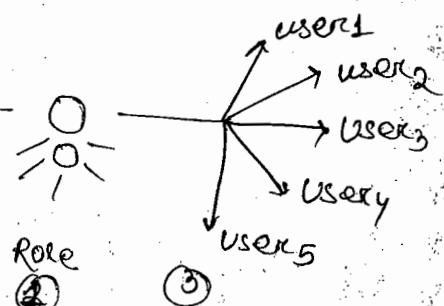
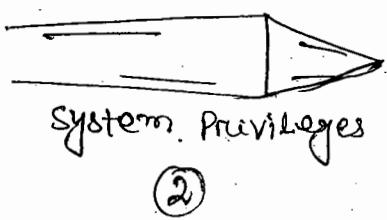


## Oracle (Copy - 2)

06/03/14

### What is Role :-

- 4 In all databases database administrators only create role.
- 4 Role is nothing but collection of system or object privileges
- 4 Generally in multiuser environment no. of users works on same project, in this case some set of users required same set of privileges.
- 4 In this case only dba creates a role and assigns common set of privileges into role & then only that role given to the no. of users.



### Step 1:- (Creating a role) -

Syntax :-

```
create role rolename;
```

### Step 2:- Assign system privileges to a role :-

Note:- Syntax :-

```
grant system privileges to rolename;
```

### Step 3:- Assigning rolename to number of users

Syntax :-

```
grant rolename to username1,username2 ...;
```

Ex:-

sql conn sys as sysdba;

Enter password : sys

sql create role r10;

sql 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. → administrator.

\* 4 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 ROLEIN ('DBA');

Object Privileges :-

↳ Object privileges are used to perform some operations  
on the object these privileges are given by  
either developer or database administrator.

↳ Oracle having following object privileges, these are :-

insert, update, delete, select.

Called as 'all'.

↳ execute, read, write.

Syntax:-

Grant objectPrivileges on objectname  
to usernames / Rolename / Public ;

say show user;

User is "SCOTT"

sql> grant all on emp to India;

sql> grant all on emp to ryo;

sql> grant all on emp to Public;

↳ All object privileges related to user stored under user-tab-Prvs data dictionary.

say desc user-tab-Prvs;

with grant option:-

↳ who receives with grant option class those user give privileges to another user?

Syntax:-

Grant objectprivileges on objectname  
to usernames/Public with grant option;

say> grant all on emp to India with grant option;

sql> grant all on emp to ryo with grant option;

say> errors: 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 cancel either system or objectprivileges.

Syntax:-

Revoke systemprivileges from usernames,username

Syntax:-

Revoke objectprivileges on objectname  
from usernames/Rolename;

↳ Generally if we want to restricts ~~the table~~ of &  
from one user into another user then we

must create a view with a required columns  
Is that view given to the no. of users.

say conn scott/tiger;

say create or replace view v<sub>1</sub> as

select

say grant all on v<sub>1</sub> to soma;

say conn soma/soma;

say select \* from scott.v<sub>1</sub>;

### MERGE :-

↳ Oracle 9i introduced MERGE statement.

↳ Basically this is an DML command.

↳ Generally MERGE statement used data warehousing

↳ Using MERGE statement we are transforming  
data from source table into target table.

Syntax:-

MERGE into target tablename

using source tablename

on (join condition)

when matched then

update set targettable colname = Sourcetablename,

when not matched then

insert (targettable colname) values (Sourcetablename);

Ques

< oracle >

07/03/14

- ↳ if merge statements are using insert, update statements conditionally.
- ↳ Through merge statement we can't modify on clause columns.  
e.g:-
  - 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.deptno = 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);

### 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 admin.
- ↳ Sequence is a independent database object, once a sequence has been created no. of users simultaneous access that sequence.
- ↳ In all database system sequence are used in automation concept.

Syntax:- (auto increment)

```
create sequence sequencename  
start with n  
increment by n  
minvalue n  
Maxvalue n  
cycle/nocycle  
cache/no cache;
```

eg:- create sequence s<sub>1</sub>

start with 5  
increment by 2  
maxvalue 100;

5  
7  
9  
11  
13  
15  
17  
19  
21  
23  
25  
27  
29  
31  
33  
35  
37  
39  
41  
43  
45  
47  
49  
51  
53  
55  
57  
59  
61  
63  
65  
67  
69  
71  
73  
75  
77  
79  
81  
83  
85  
87  
89  
91  
93  
95  
97  
99  
100

↳ If we want to generate sequence no. then we must use following pseudo col.

- 1) curval
- 2) nextval.

Syntax:-

```
sequencename.curval
```

Syntax:-

```
sequencename.nextval
```

↳ Sequences are used to insert, update, delete, Select statement.

↳ In Oracle if we want to generate Sequence values using select statement then we must use dual table.

Syntax:-

```
Select sequencename.curval 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. Beacoz curval pseudo column always returns.

current value of the sequence, if sequence session having a value.

e.g.: - select s1 . current from dual;

error: sequence s1 . current is not yet defined  
in this session.

sql > select s1 . nextval from dual;

NEXTVAL

sql > select s1 . nextval from dual;

NEXTVAL

5

:

>If we want to change sequence Property value then we are using alter sequence.

Syntax:-

Alter sequence sequencename Property name newvalue ;
---

sql > alter sequence s1 increment by -1;

N.B:- In all database system we can not change start with value using alter sequence.

e.g.: - alter sequence s1 start with 4;

error: cannot alter starting sequence number.

In sequences start with can not be less than minvalue.

e.g.: - create sequence s1

start with 3

increment by 1.

minvalue 5

maxvalue 10;

O/P:- error: START WITH cannot be less than MINVALUE.

cycle/nocycle :-

say create sequence s1

start with 5

increment by 1

minvalue 3

Maxvalue 10

cycle

// nocycle;

say select s1.nextval from dual;

op:-5

6

—

10

—

3 → cycle

4

—

10

—

3 → cycle

say create sequence s1

start with 5

increment by 1

minvalue 3

maxvalue 10

nocycle

nocache;

say select s1.nextval from dual;

op:-5

6

—

10

error:- sequence s1.NEXTVAL exceeds MAXVALUE and cannot be instantiated.

say create table test ( sno number(10) Primary key ,  
name varchar2(10));

say create sequence s1 start with 1;

say insert into test values (s1.nextval, 'abc');

Enter value for name :abc

say /

Q) SQL Select \* from test;

SNO NAME

1	abc
2	xyz
3	pqr

CACHE:-

## Methods of servlet context :-

↳ `java.lang.String getInitParameter(java.lang.String name);`

- Returns 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 initialization parameters as an enumeration of string objects, or an empty enumeration if the context has no initialization parameters.

How to get servletcontext object ?

1) `getServletContext()` method of `servletConfig` interface.

- `ServletContext ctx = getServletConfig().getServletContext();`

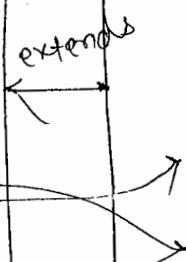
2) `getServletContext()` method of `GenericServlet`.

- `ServletContext ctx = getServletContext();`

### GenericServlet

```
private ServletConfig config;  
P.V. init(ServletConfig config){  
    this.config=config;  
}  
ServletConfig getServletConfig(){  
    return config;  
}  
P. ServletContext getServletContext(){  
    return config.getServletContext();  
}
```

### My servlet



09/03/14

Ques

### What is Cache in?

Cache is a memory Area which is used to stored sequence numbers.

Cache option used by Database Administrator to retrieve 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 program access sequence no. from memory area these process automatically improves performance of the application.

In Oracle by default cache value is 20 and also min value is 2.

When ever system creates automatically cache values are used.

e.g:- sql create sequence s1  
start with 1.  
increment by 1.  
maxvalue 100  
cache 1;

O/P:- Error: the number of value to cache must be greater than 1.

All sequences info stored in user\_sequences data dictionary.

sql) desc user\_sequences;

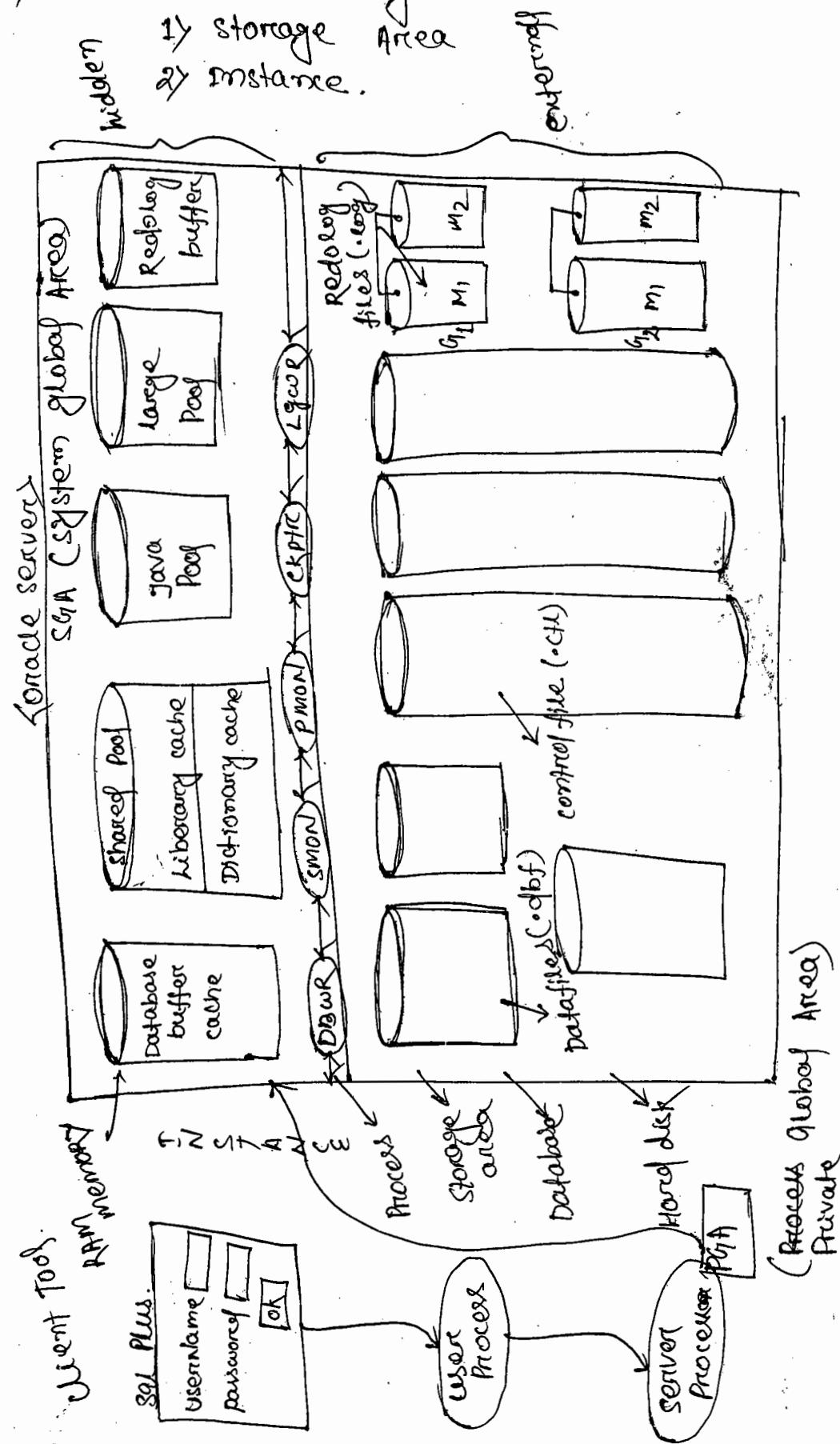
We can also drop sequence using drop-sequence sequencename.

sql) 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) Redolog files (.log).

These 3 files are also called as physical database or storage area.

### i) Datafiles

In Oracle whenever we are creating database object those objects are permanently stored datafiles.

All datafiles information stored under dba-data-files data dictionary.

eg:-  
sql> conn sys as sysdba;  
Enter password: sys  
connected.

sql> desc dba-data-files;

sql> select file\_name from dba\_data\_files;

### iii) Control files:-

Control files stores database information.

This files extent is .ctl.

Control files are used by dba in recovery process.

All control files <sup>information</sup> stored render V\$controlfile data dictionary.

eg:-  
sql> conn sys as sysdba;

Enter password: sys

sql> desc V\$controlfile;

sql> select name from V\$controlfile;

### ii) Redolog files:-

It stores committed information from Redolog buffer.

10/03/14

- ↳ Redolog files also used by dba in recovery process.  
e.g:- desc V\$logfile;  
sql> select Member from V\$logfile;

## 2) Instance :-

- ↳ 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 :-

- 1) i) SGA
- ii) Process.

### i) SGA :- (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.

### ii) Process

- when ever we are ~~submitting~~ executing sql, pl/sql code that code automatically stored in library cache within shared pool.

- ↳ This library cache reduces parsing.

- ↳ Library cache also stores cache values defined in sequence objects.

- ↳ shared pool also contains dictionary cache which execute DCL related data dictionary information.

- ↳ whenever we are requesting a table residing 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.

↳ Physical structure :-

- ↳ physical structure contains 3 files visible in O.S.
- ↳ physical files are handle by databaseAdmin. 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.

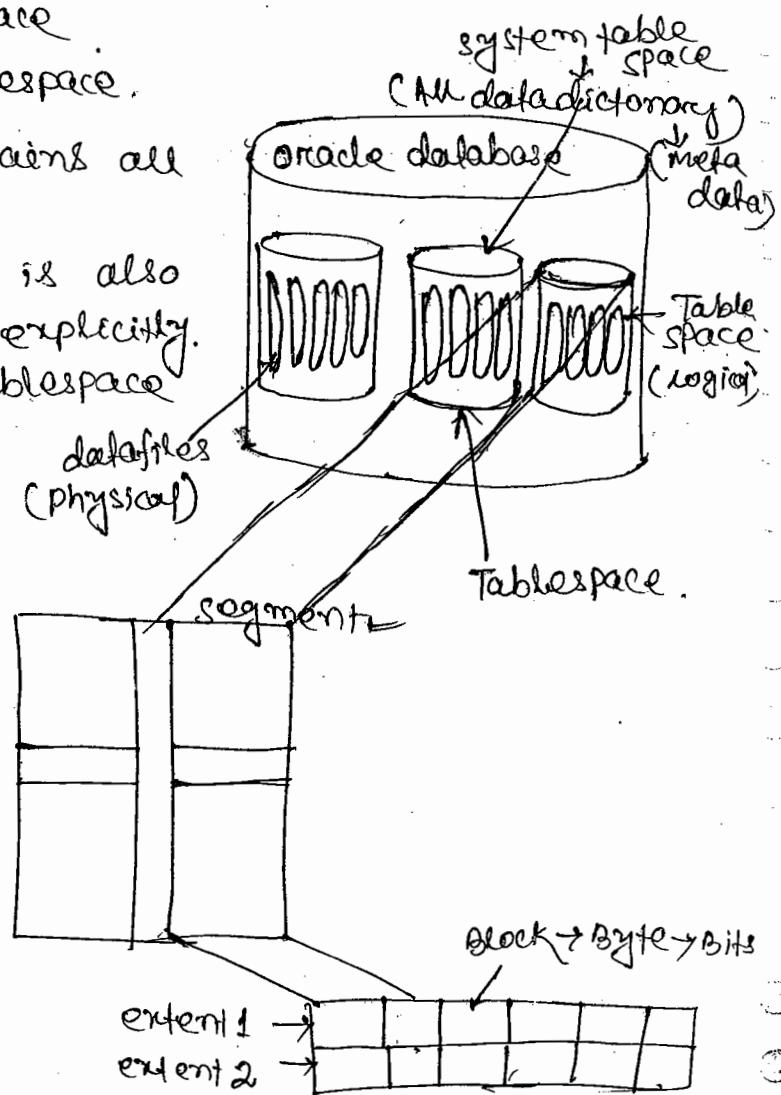
↳ whenever we are installing oracle server automatically 6 tablespaces are created.

↳ If we want to run oracle minimum 2 tablespaces are required, these are:-

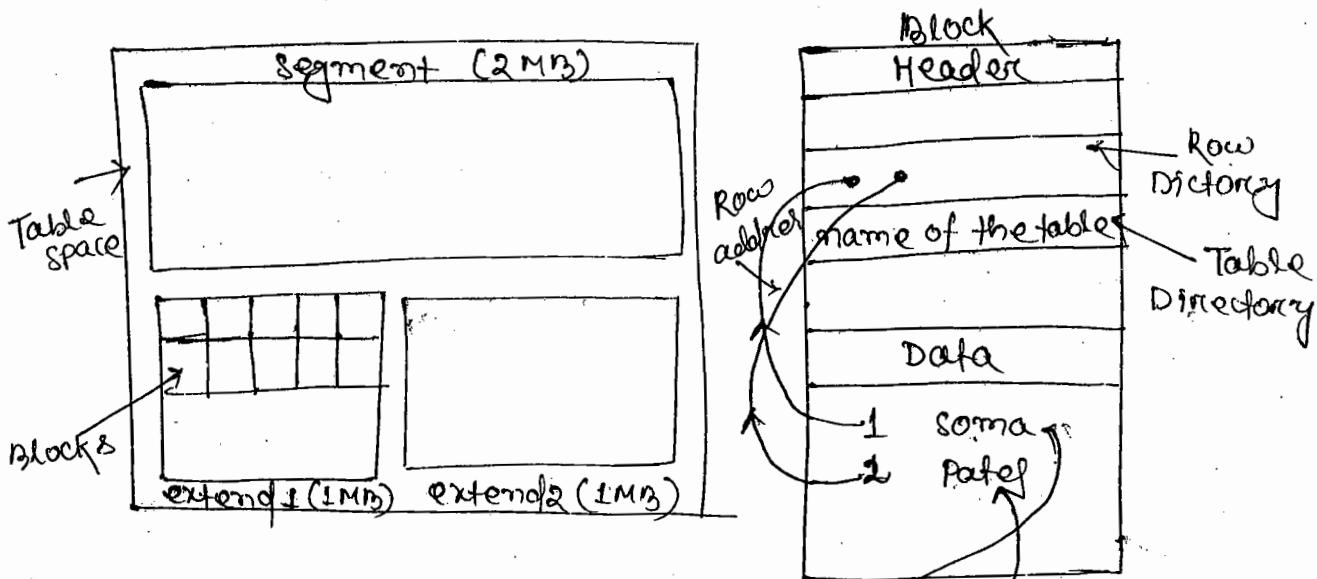
- system tablespace
- sysaux tablespace.

↳ System tablespace contains all data dictionaries.

↳ Database Administrator is also create tablespaces explicitly by using create tablespace privileges.



sql> create table T<sub>1</sub>(sno number(10), name varchar2(10))



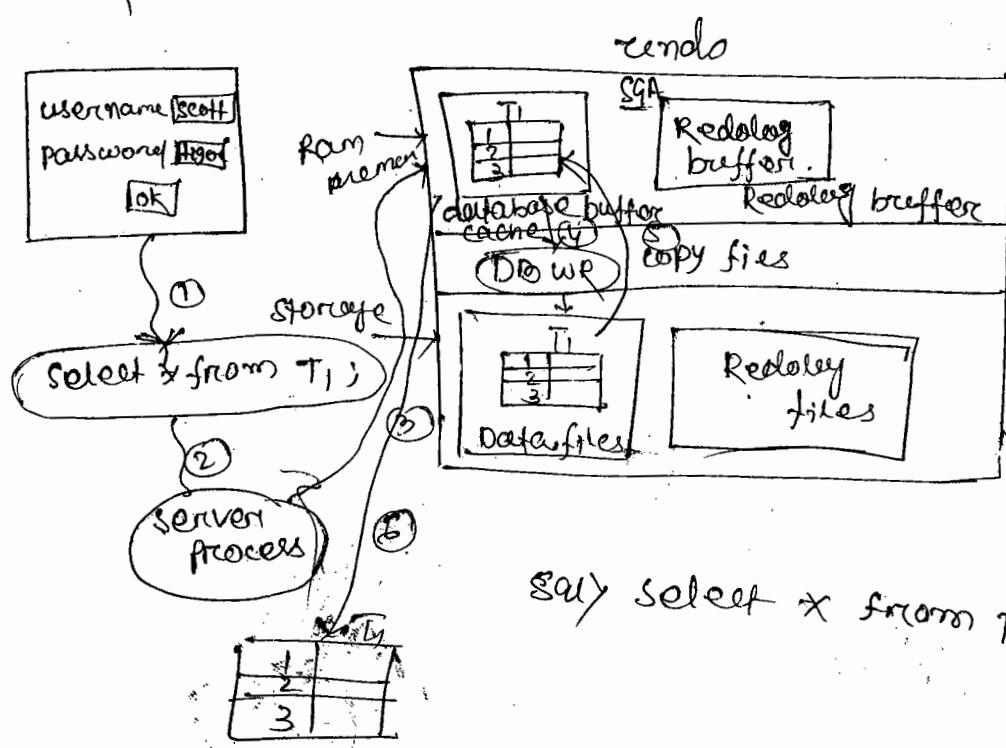
sql> insert into T<sub>1</sub> values (1, 'soman');

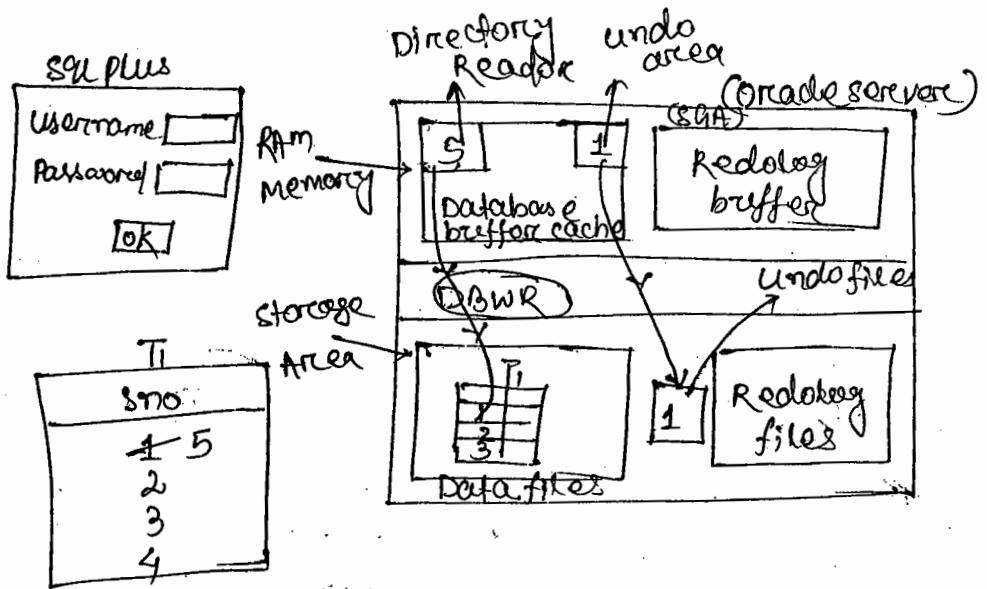
sql> insert into T<sub>1</sub> values (2, 'Patel');

### Segment:

↳ Segment is a collection of extents that forms a database object like, tables, indexes, views,

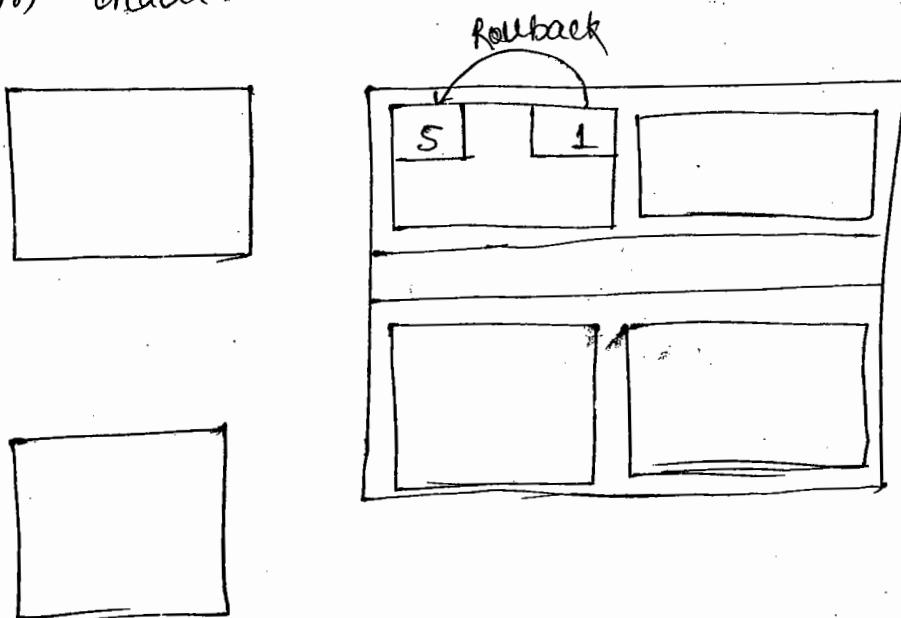
NB: in all database systems list storage element is Block.





**N.B:-** If oracle database whenever we are performing a transaction new value <sup>in DB</sup> for the transaction automatically stored ~~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.

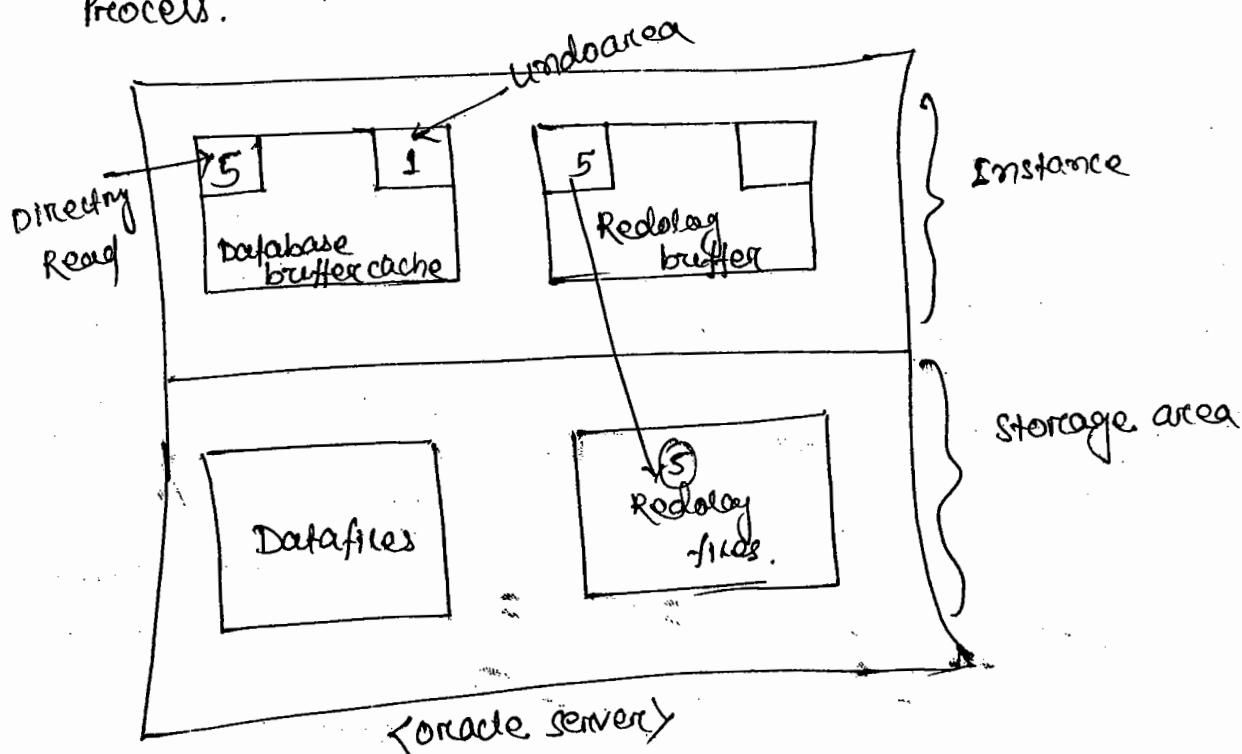
These undo files are used by flashback queries in oracle.



DT 11/03/14

Q Mea  
? Redolog buffer :-

When we are performing transaction, new values for the transaction automatically stored in Redolog buffer. Whenever user going commit / no 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 co-writer a paper "relational Model of data for large shared data banks".
- ↳ In this paper only, E.Codd introduced 1st 3 normal forms.

### Normal forms :-

- 1) first normal form (1NF)
- 2) second      "      " (2NF)
- 3) Third        "      " (3NF)
- 4) BCNF      (Boyce 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                                  candidate key                                  Item table

Item Name	color	Price	tax
Marker	black, Red	20	0.2
Pen	blue, green	30	0.3

$\xrightarrow{1NF}$

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

(NOT IN 1NF)
(IN 1NF)

Orderno :	<input type="text"/>
orderdate :	<input type="text"/>
customer Address :	<input type="text"/>
customer Name :	<input type="text"/>
phone Number :	<input type="text"/>
ItemName :	<input type="text"/>
Amount	<input type="text"/>
<input checked="" type="checkbox"/> add another item	
<input type="button" value="ok"/>	

\* In this example Itemname amount are repeating group. so draw it in another table and for more atomicity add Orderno to that table.

Primary Key

→

Orderno	order date	customer Address	customer Name	ph no

(Order - Master table)

Order no	Item name	Amount
1	Bluestar AC	40.000
1	LED TV	30.000
1	Refrigeration	45.000

→ order details of  
table or child table.

(1NF)

Identify Repeating Groups and put it into separate table in more atomic form.

In all database system by default 1<sup>st</sup> 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 1<sup>st</sup> normal form and also all non-key attributes fully functionally dependent on total candidate key.

Generally, 1<sup>st</sup> NF deals with Atomicity whereas 2NF deals with relationship b/w key, non-key attributes.

When a table contains particular non-key attributes then the table not in 2nd normal form.

### Process :-

Identify the part of 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 master table. In this table only all nonkey

attributes fully functionally dependent on total candidate key.

Mac

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

non-key attribute

Foreign Key

Preliminary Key

Master Table

12/03/14

Sno	name	Activity	cost
101	abc	Cricket	\$10
102	xyz	Hockey	\$30
103	zzz	Swimming	\$40
104	aaa	Cricket	\$10

K primary attribute  
Candidate key

Sno	Activity	cost
101	Cricket	\$10
102	Hockey	\$30
103	Swimming	\$40
104	Cricket	\$10

Non-key attribute  
Part of P

Sno	name	Activity	cost
101	abc	Cricket	\$10
102	xyz	Hockey	\$30
103	zzz	Swimming	\$40
104	aaa	Cricket	\$10

Sno	Activity	cost
101	Cricket	\$10
102	Hockey	\$30
103	Swimming	\$40
104	Cricket	\$10

Primary Table  
and half

Primary Table  
and half

Primary Table  
and half

Primary  
key

Primary Table  
and half

Primary  
key

key attribute  $\rightarrow$  candidate key

Project Table

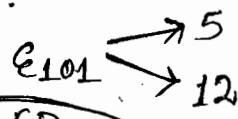
Ecode	Projectcode	Dept	Depthead	Hours	F
E101	P <sub>1</sub>	System	D <sub>1</sub>	5	
E102	P <sub>1</sub>	Sales	D <sub>2</sub>	7	
E103	P <sub>2</sub>	Research	D <sub>3</sub>	9	
E101	P <sub>2</sub>	System	D <sub>1</sub>	12	
E104	P <sub>3</sub>	S/W	D <sub>4</sub>	8	
E102	P <sub>3</sub>	Sales	D <sub>2</sub>	15	

not in 2<sup>nd</sup> NF

Phase 1 :- (uniqueness check called as FD)  
 FD (Functional dependency)

① Ecode  $\rightarrow$  Hours (X)

for every value for Ecode more than one value in hours.



② Projectcode  $\rightarrow$  Hours (X)



③ Ecode + Projectcode  $\rightarrow$  Hours (V)

Phase 2 :-

① Ecode  $\rightarrow$  Dept (V)

$$E101 \rightarrow \text{System}$$

② Projectcode  $\rightarrow$  Dept (X)

$$P_1 \rightarrow \text{System}$$

$\rightarrow$  Sales

Phase 3 :-

① Ecode  $\rightarrow$  Depthead (V)

② Projectcode  $\rightarrow$  Depthead (X)

Primary key (Master Table) → Foreign key (child table)

ecode	dept	depthead	ecode	Projectcode	hours
E101	System	D <sub>1</sub>	E101	P <sub>1</sub>	5
E102	Sales	D <sub>2</sub>	E101	P <sub>2</sub>	12
E103	Research	D <sub>3</sub>	E102	P <sub>1</sub>	7
E104	S/W	D <sub>4</sub>	E102	P <sub>3</sub>	15
			E103	P <sub>2</sub>	9
			E104	P <sub>3</sub>	8

Ans

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.

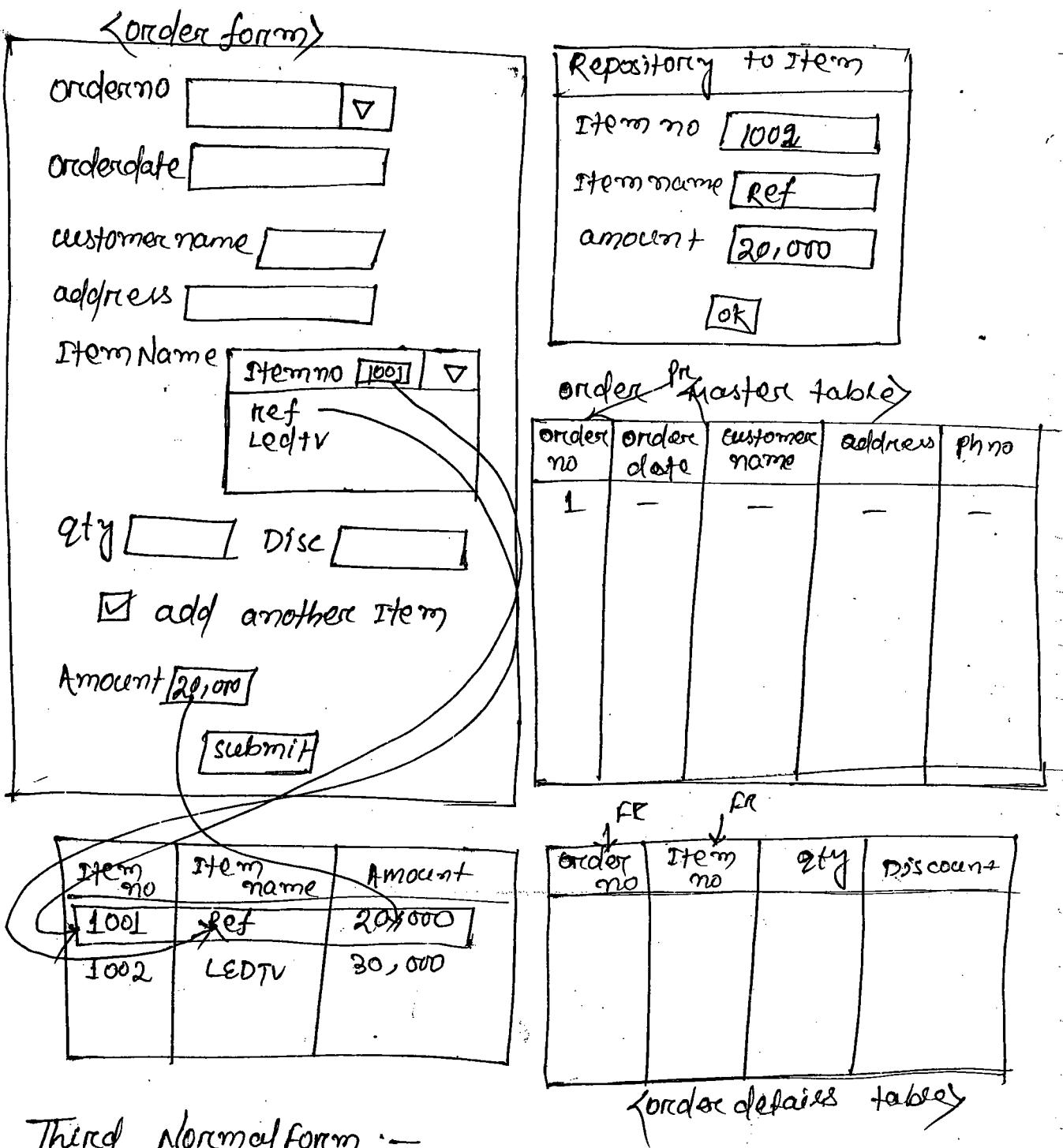
↳ Updation Problem :-

↳ In the above resource table ecode, dept, depthead values are repeated. whenever a particular employee transferred one dept to another dept then we must modified on these 3 attribute values. correctly other updation problem occurred. 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 using normalization process automatically insertion, updation, deletion problems are avoided.



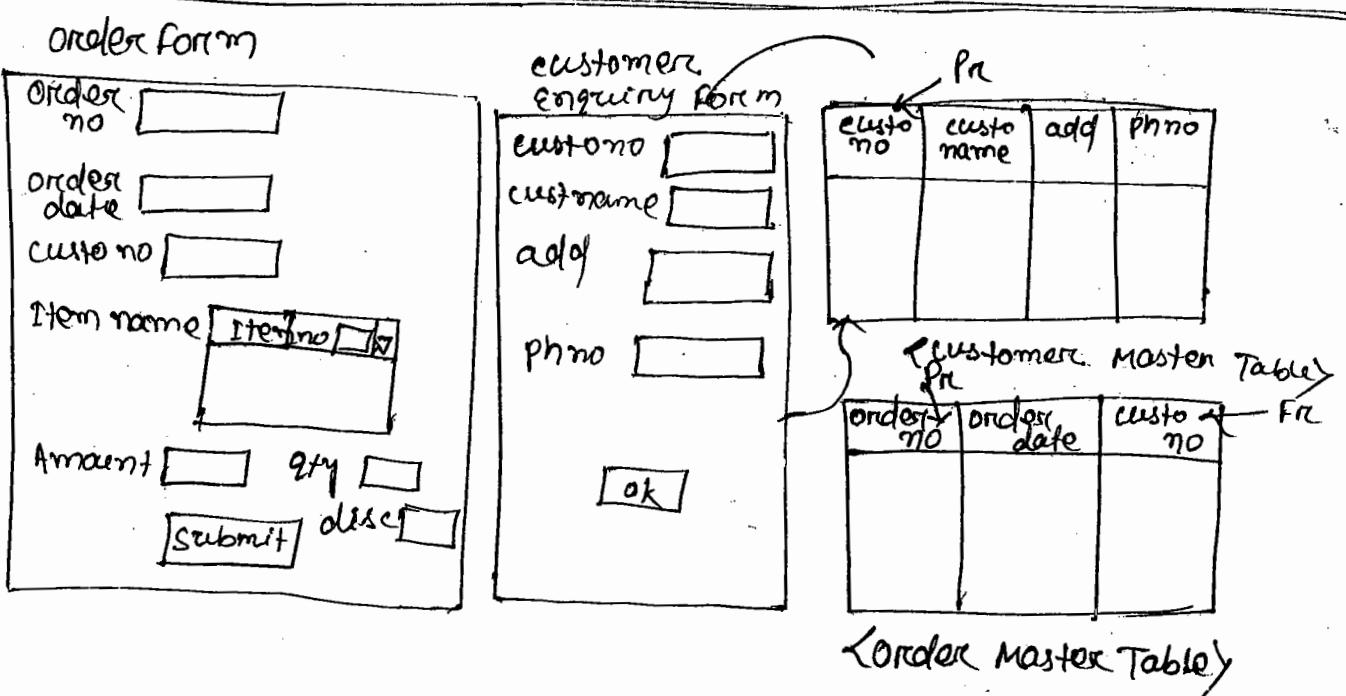
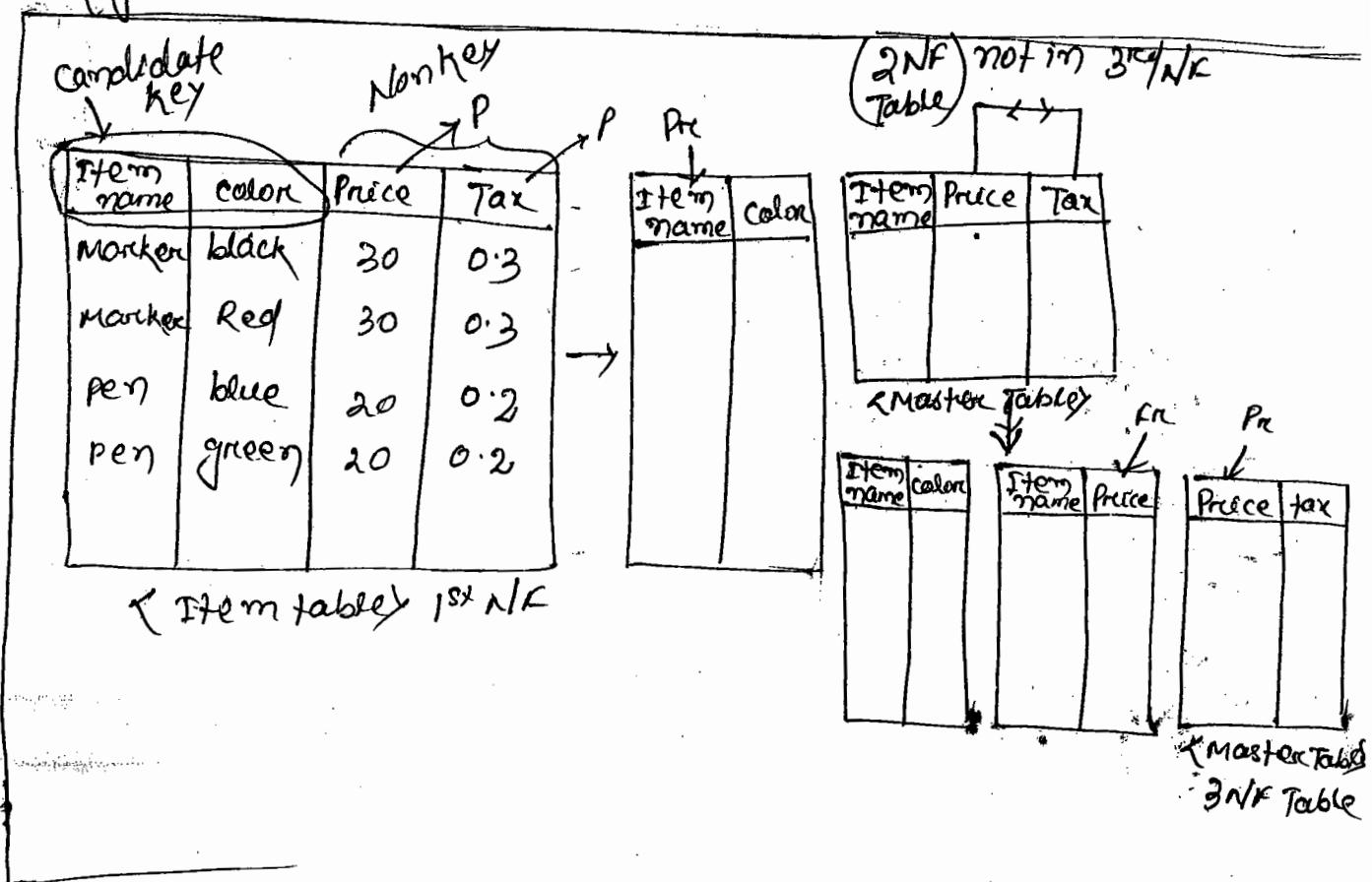
- A table is in 2NF if and also all non-key attributes only dependent on <sup>only</sup> primary key.
- whenever non-key attributes which are depends on another non-key attributes from that table not in 3NF normal form.

Process:-

- Identifying non-key attributes which depends on another non-key attributes those all attributes

put into separate table. These table is called 3rd NF Table.

By default this is an master table. in this table only all non-key attributes are only depends on primary key.



Pre 2NF		
Item no	Item name	Amount

Item Master

FR FR			
Order no	Item no	Qty	Disc

Order details Table

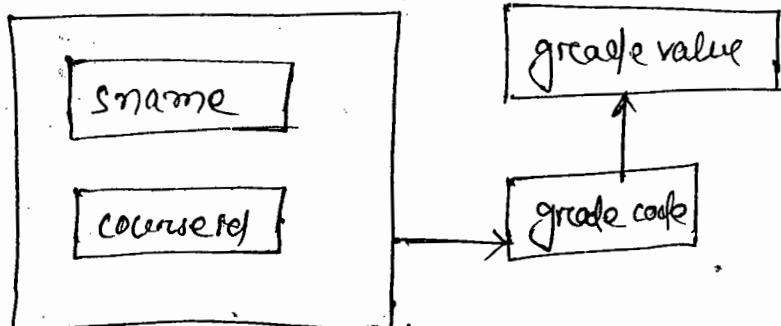
Map

Candidate  
key

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 3NF)

Logical Diagram:-



Functional Dependency:-

If any 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.

14/03/14

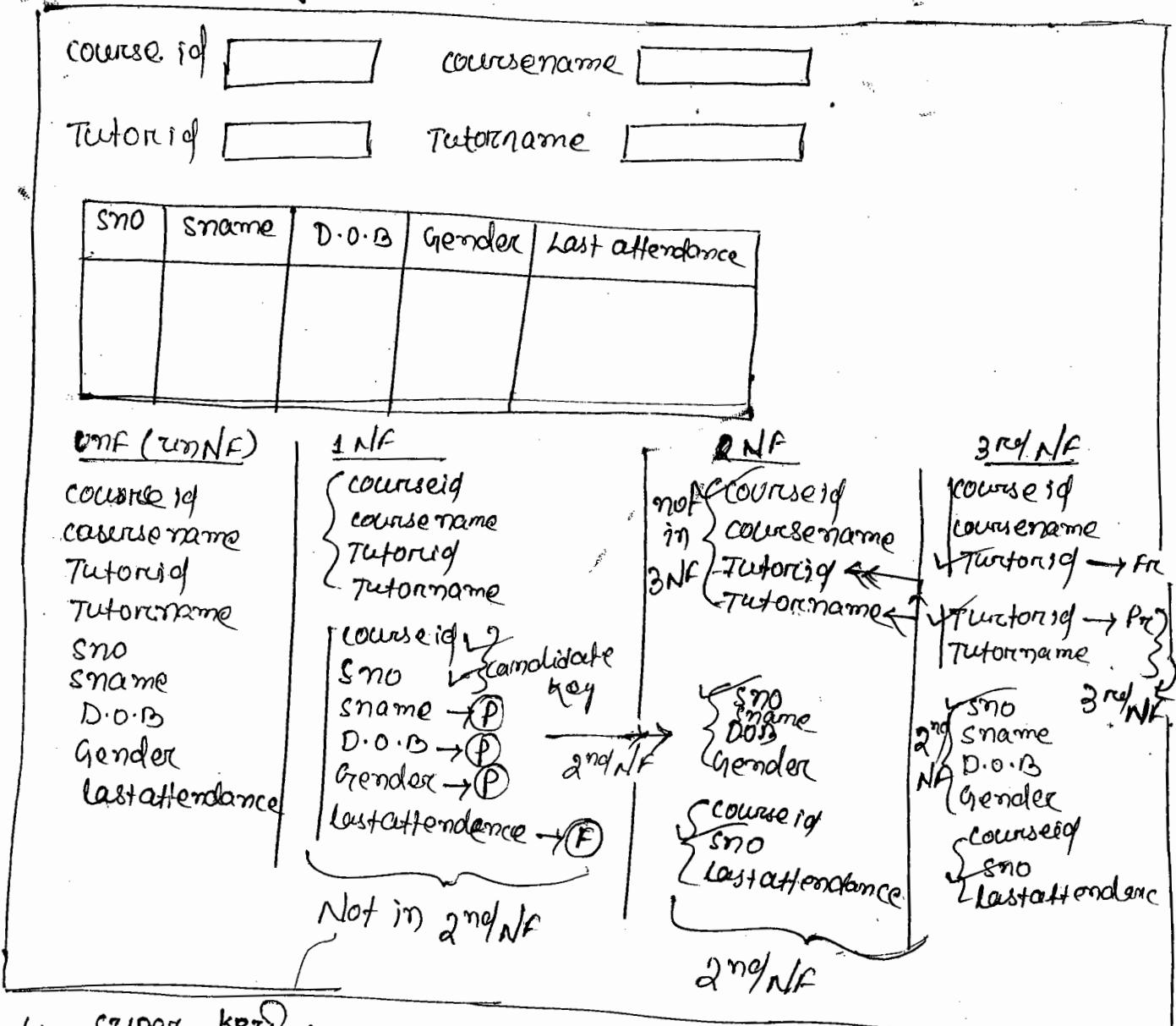
Pre (3NF)

sname	cust id	Grade
abc	CS111	A
xyz	CS111	B
zzz	CS222	C
aaa	CS222	A

Grade	grade value
A	4.00
B	3.00
C	2.00

Master table

- 1NF → remove repeating groups.
- 2<sup>nd</sup> NF → Remove Partial attributes.
- 3<sup>rd</sup> NF → Remove attributes those are not dependent on key  
(OR)  
Remove attributes which are dependent on another non-key attributes.



↳ super key :-

A column or operational column which uniquely identifies a record in a table.

↳ candidate key :-

Minimal superkey which uniquely identify record in a table is called candidate key.

↳ i.e. a superkey which is subset of another key & 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:-

Other than Primary key candidate keys are called as alternative key.

Empno	ename	skillid	skill	voterid
1	soma	1	oracle	v <sub>1</sub>
1	soma	2	sybase	v <sub>1</sub>
1	soma	3	Ingress	v <sub>1</sub>
2	abc	4	DB2	v <sub>2</sub>
2	abc	1	oracle	v <sub>2</sub>
3	xyz	5	Informix	v <sub>3</sub>
4	zzz			v <sub>4</sub>
5	aaa	6	mssql server	v <sub>5</sub>
5	aaa	4	DB2	v <sub>5</sub>

## 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 } → Alternative key.
- 3) voterid + skill }

15/03/14

Q Max

### CLUSTER

- ↳ Cluster is a database object which is used to improve performance of the join query.
- ↳ Cluster contains group of tables together and shared 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 requested tables are available in cluster or not.
- ↳ if from all tables available in cluster then database servers very fastly retrieve data from cluster table.

Generally database administrator only creates cluster at the time of table creation.

- ↳ Based on common col name

### Creating a cluster:-

Step:-1

#### cluster creating? on 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  
(colcolumnname);
```

Eg:-

sql> create cluster emp-dept (deptno number(10));

sql> create index z on cluster emp-dept;

sql> create table e1(deptno number(10), empno number(10),  
ename varchar2(10), sal number(10))  
cluster emp-dept (deptno);

sql> 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 using including  
tables clause.

Syntax:-

```
drop cluster clustername including tables;
```

Eg:-

sql> drop cluster emp-dept;

error: cluster not empty.

sql> drop cluster emp-dept including tables;

cluster dropped.

sql> desc user-clusters;

All clusters information stored under user-clusters  
data dictionary.

## BCNF:-

A Table is a BCNF, then every determinant is a candidate key.

Whenever tables having multiple candidate keys and also these candidate keys are overlapped 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 then deletion problem occurred, to overcome this problem we are using BCNF process.

## Process:-

Identifying one candidate key non-key attribute which depends on another candidate key non-key attributes put in to separate tables. This table is called BCNF.

In this table only every determinant behaves like a candidate key. and also this table doesn't contain non-key attributes.

Ecode	name	Projectcode	hours
E1	abc	P1	8
E2	xyz	P2	5
E3	zzz	P4	6
E4	soma	P1	7
E5	soma	P2	9
E6	aaa	P3	10

Non Key attribute

- FD
- ① Ecode + Projectcode → hours
  - ② name + Projectcode → hours
  - ③ Ecode + Projectcode → candidate key
  - ④ name + Projectcode → candidate key

(Project-table)

Ecode	name
E <sub>1</sub>	abc
E <sub>2</sub>	xyz
E <sub>3</sub>	xxz
E <sub>4</sub>	soma
E <sub>5</sub>	aaa

(BCNF Table)

① Ecode → name  
 ↓  
 determinant  
 ↓  
 candidate key.

② name → Ecode  
 ↓  
 determinant  
 ↓  
 candidate key.

Profid	Dept	H.O.D	hours
P <sub>1</sub>	chemistry	H <sub>1</sub>	5
P <sub>1</sub>	comp	H <sub>2</sub>	7
P <sub>2</sub>	Physics	H <sub>3</sub>	6
P <sub>3</sub>	Zoology	H <sub>4</sub>	10
P	Botany	H <sub>5</sub>	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 <sub>1</sub>
comp	H <sub>2</sub>
Physics	H <sub>3</sub>
Zoology	H <sub>4</sub>
Botany	H <sub>5</sub>

{BCNF Table}

② H.O.D → Dept  
 FD

↓  
 determinant

↓  
 candidate key.

① Dept → H.O.D  
 FD

↓  
 determinant

↓  
 candidate key

Profid	Dept	hours
P <sub>1</sub>	chem	5
P <sub>1</sub>	comp	7
P <sub>2</sub>	Physics	6
P <sub>3</sub>	Zoology	10
P <sub>3</sub>	Botany	12

Forth Normal Table (4<sup>th</sup> NF) :-

17/03/14

Q Ques

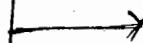
Q Fourth Normal Form :-

- ↳ A Table is a fourth normal form, Then there is no multivalue dependency, i.e.
- ↳ If a table is a 4<sup>th</sup> N.F then that table doesn't contain more than one independent multivalued attribute.
- ↳ Multivalued dependency occurs, when a table contains more than 2 bits and also identifies record using combination all bits & also one attribute set of values which depends on another value attribute set of values and also some attribute set of values which depends on another attribute set of values & also some attributes are not logically related then only we are using 4<sup>th</sup> N.F.

Process :-

- ↳ identifying independent attribute which depends on dependent attribute put it in to separate table.

Movie	Star	Procedures
M <sub>1</sub>	S <sub>1</sub>	P <sub>1</sub>
M <sub>1</sub>	S <sub>2</sub>	P <sub>1</sub>
M <sub>1</sub>	S <sub>3</sub>	P <sub>1</sub>
M <sub>1</sub>	S <sub>4</sub>	P <sub>1</sub>
M <sub>1</sub>	S <sub>5</sub>	P <sub>2</sub>
M <sub>1</sub>	S <sub>6</sub>	P <sub>2</sub>
M <sub>2</sub>	S <sub>7</sub>	P <sub>5</sub>
M <sub>2</sub>	S <sub>1</sub>	P <sub>5</sub>
M <sub>2</sub>	S <sub>2</sub>	P <sub>5</sub>
M <sub>2</sub>	S <sub>3</sub>	P <sub>5</sub>



Movie	Star

movie	Procedures

NB:- when a table contains only one independent multivalued attribute then that table automatically reduces representing information.

Project Table ← Candidate key		
empid	Project id	skills
1111	Railway Res	JSP
1111	Railway Res	Oracle
1111	Railway Res	.Net
1111	Library on	.php
1111	Library Pre	SQL Server
1111		HA

The diagram illustrates the decomposition of the 'Project Table' into two separate tables. An arrow points from the original table on the left to the two decomposed tables on the right.

empid	Project id
1111	Railway
1111	Library

empid	skills
1111	JSP
1111	Oracle
1111	.Net
1111	.PHP
1111	SQL Server

↳ 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 designer uses 4<sup>th</sup> NF Process.

### Fifth Normal Form :-

↳ A Table is in 5<sup>th</sup> NF then that table doesn't contain cyclic dependency. Cyclic dependency occurs when a table contains more than two fields & also identifying a record through combination of all fields and also all fields are related to one another then only database designer uses 5NF.

↳ 5<sup>th</sup> NF is also called as projection joined NF.

↳ In 4<sup>th</sup> NF some attributes are not logically related.

but in 5thNF all attributes are logically related.

Buyer	Product	company
B1	Shirt	Levi's
B1	Jeans	Arrow
B2	shirt	Pepe's

→

buyer	Product	company	buyer	comp

### Indexes:-

- ↳ it is a Database object which is used to improves performance of the application, becoz indexes are column retrieve data very fastly from the DB.
- ↳ Indexes are created of tables columns.
- ↳ Generally we are creating indexes two ways
  - ↳ Automatically.
  - ↳ manually.

### Automatically:-

whenever we are creating 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 using create index ~~privileges~~ Privileges.

syntax:-

```
create index indexname on tablename (colname);
```

- ↳ In all Database whenever select query contain where or order by <sup>clause</sup>, then only DB server checks those columns having indexes in DB.

whenever indexes are available then DB servers very fastly retrieve data from DB.

N.B.:- In oracle when a where clause contains not equal to ( $\neq$ ), is null, is not null operators, then oracle server does not search for indexes, if those cols already having indexes also.

↳ Generally indexes are created by DBA only.

↳ Oracle having two types of indexes:-

1) Btree indexes.

2) bitmap indexes.

↳ b-tree indexes :-

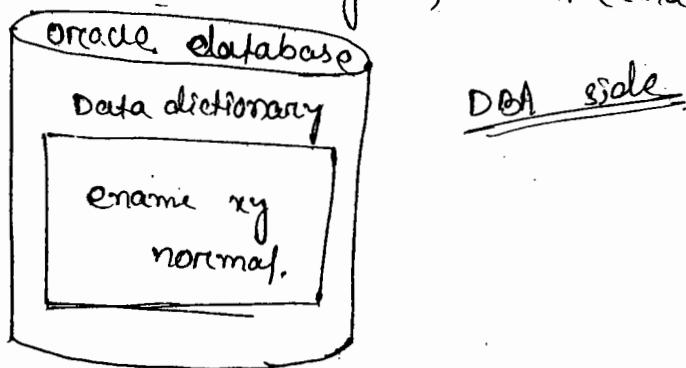
in oracle by default indexes are b-tree indexes.

↳ whenever we are creating b-tree indexes then Oracle server automatically creates a b-tree structure.

↳ In this b-tree str always leaf block stores actual data along with rowids.

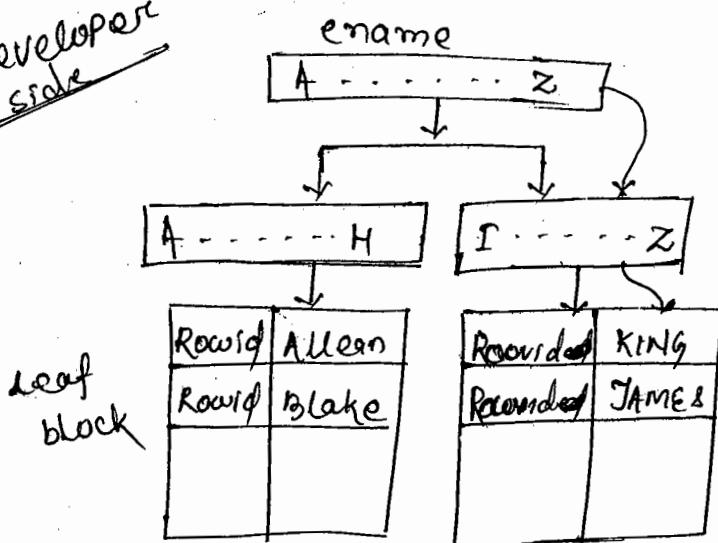
↳ whenever we are submitting index column in where clauses then oracle server searches index col available in oracle database, if it is available then Oracle server uses index scan method to search/retrieve requested data from b-tree structure.

SQL create index xy on emp(ename);



say select \* from emp where ename='KING';

developer side



↳ 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:-

<u>INDEX-NAME</u>	<u>INDEX-TYPE</u>
PK-EMP	NORMAL
xy	NORMAL

NOTE :- If we want to view col names along with index name then we are using user-ind-column Data dictionary.

e.g:- say desc user-ind-columns;

say select index-name, column-name from user-ind-  
columns where Table-name = 'EMP';

O/P:- Index-name

PK-EMP  
xy

column-name

EmpNo.  
ename:

May

18/03/14

- ↳ Oracle having 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 using following syntax.

create unique index indexname on tablename  
(colname);

N.B.— we are not allowed to create unique index on duplicate value columns.

sql create unique index ab on emp(job);  
error: Cannot create unique index; duplicate keys found.

bitmap indexes :-

- ↳ Oracle 7.3 introduced bitmap indexes.
- ↳ Generally bitmap indexes are used in data warehousing application.
- ↳ Generally bitmap indexes are created on low cardinality columns.
- ↳ These cols having more duplicate data.

create bitmap index indexname on tablename  
ab (colname);

- ↳ sql create bitmap index ab on emp(job);
- ↳ whenever we are submitting bitmap index oracle automatically create a bitmap table.

- ↳ whenever user is querying a data using logical operators, then Oracle server directly operates bits with in bitmap tables. then the result bitmap automatically converted into rows 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 b-tree indexes.
- In oracle whenever 'where' clause 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(functionname  
(colname));

e.g:- create index em1 on emp(ename)<sup>(upper)</sup>;  
- index created.

say Select index\_name, index\_type from user\_indexes  
where table\_name = 'EMP';

O/P:- Index-name

IN1

index-Type

function-based normal.

### Virtual columns:-

- ↳ Oracle 11g introduced virtual cols in a table.
- ↳ In Oracle DB to store stored expressions then we are using views or function based indexes.
- ↳ These two methods indirectly stores stored expressions in DB.
- ↳ Oracle 11g introduced to store stored expressions directly in a table using virtual cols through generated always as clause.

### Syntax:-

col name datatype(size) generated always as (function  
name or expression (col name));  
↳ [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;

<u>a</u>	<u>b</u>	<u>c</u>
10	20	30

Q: If we want to view Virtual col exprs, then we are using data-default property from user-tab-columns data dictionary.

say desc user-tab-columns;

↳ we can also drop index using drop index indexname.

- ① Set Operators :-
  - ② ↗ setof operators are used to retrieve data from single and multiple tables.
  - ③ ↗ These operators are also called as vertical joins.  
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)
- Ex:- Select Job from emp where deptno=10 Union  
 select Job from emp where deptno=20;

↳ when we are using setof operation corresponding expression must belongs to same datatype otherwise current server returns an Error.

Ex:- select deptno from emp union

select dname from dept;

Error: expression must have same datatype as corresponding expressions.

Say select dname from emp union } o/p- ename

select dname from dept; }

↳ always setof operators returns first query column names and colheadings.

Say select dname from dept union

select ename from emp;

O/P:- ename

≡

N.B we can also retrieve data using setof operators

If corresponding expression not belongs to same datatype also, in few case we must use corresponding type conversion functions.

say select deptno "deptno", to - char (null) "deptnames"  
from emp renion

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 having two types of conversions.

1) implicit conversions.

2) explicit conversions.

Note :-  
1) implicit conversion & (or) automatic conversion 19/03/14

Ex:-

1) In Oracle when an expression contains strings and also that string representing few numbers then Oracle server automatically converts that the string into number datatype.

say select sal + '100' from emp;

2) whenever we are passing number type in to character funn then Oracle server automatically converts number type into character type.

eg:- say select length (1111) from dual.

O/P:- length (1111)

3) In Oracle whenever 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

① default date format.

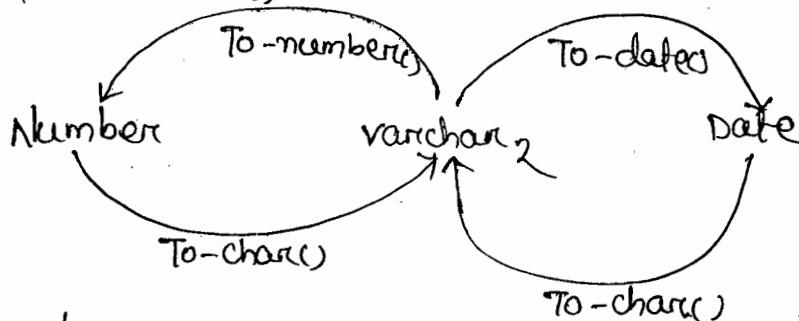
say select last\_day('17-Aug-05') from dual.

O/P :- 31-Aug-05.

② Implicit conversion Table :-

From	To	Assignment	evaluation of expression
Varchar <sub>2</sub>	num.	Y	Y
Varchar <sub>2</sub>	date	Y	Y
Number	varchar <sub>2</sub>	Y	N
Date	varchar <sub>2</sub>	Y	N

③ Explicit conversions:-



↳ To-number :-

Converting a string representing numeric value with format into numeric value without format.

say select to-number('\$45.6', '\$99.9') + 3 from dual.

O/P :- 48.6

↳ To-char :-

This is a overloading fun<sup>n</sup> which is used to convert number type into char type and also used to convert date type into date string.

say 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;  
8 19/03/14.

### To-date() :-

It is used to convert date strings into datatype.

### decode() :-

decode is a conversion function which is used to decoding 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, colname, value2, stmt2
       ... stmtn);
```

say select ename, sal, other deptno, decode(deptno,  
10, 'ten', 20, 'twenty', 'other') from emp;

O/P:-

10	ten
20	twenty
30	other

Q:- if we want to count diff data values for a single 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)) "deptno30" from emp  
group by job;

job	deptno10	deptno20	deptno30
clerk	1300	2100	950
salesman	-	-	5600
President	5000	-	-
Manager	2450	2975	2850
Analyst	-	6000	-

Ques

20/03/14

SQL>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, dept  
where e.deptno = d.deptno  
group by dname;

O/P:-

Dname	Clerks	Salemans	Managers	Analysts	Presidents
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:-

SQL>update emp set comm = 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

operations.

### Method 1 :- (syntax)

```
case colname  
when value1 then statements1  
when value2 then statements2  
-----  
-----  
else statements 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 statements1  
when column condition then statements2  
-----  
-----  
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 Queries:-

- ↳ In Relational DB, we can also store hierarchical data.
- ↳ if we want to store hierarchical data Relational table must contain min 2 cols and also

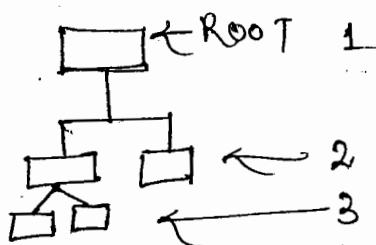
- this cols must belongs to same datatype.
- if we want to retrieve hierarchical data then we are using level pseudo col.

↳ Along with level pseudo col oracle provided following clauses.

- 1) start with
- 2) connect by

### Level:-

Level is a <sup>pseudo</sup> col which assign numbers to each level in a tree structure.



### 1) Start with :-

↳ it specified searching condition in tree structure.

#### Syntax:-

[startwith condition]

### 2) Connect by :-

↳ connect by clause by specified by relationship bet<sup>n</sup> Parent , child col using a prior operator.

#### Syntax:-

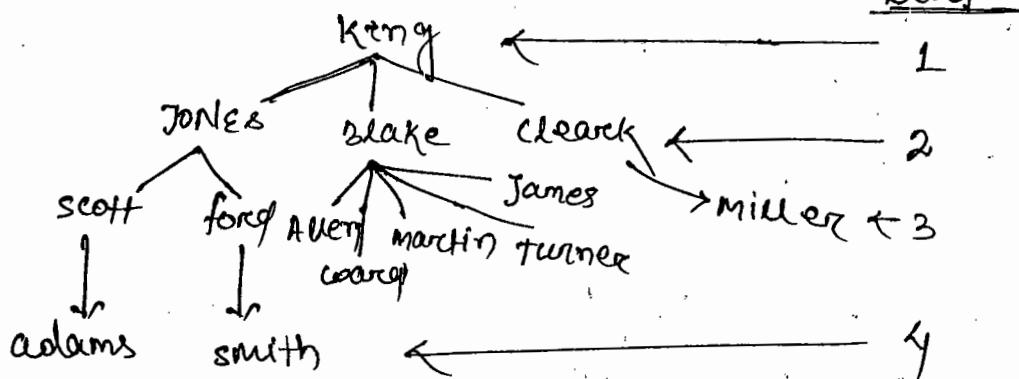
Connect by Prior Parentcol = childcolname

#### Syntax:-

Select colname1, level from tablename where condition start with condition connect by Prior Parentcol = childcolname;

Q: Write a hierarchical query to display hierarchy from emp table using empno, mgr cols.

Ans) Select level, ename from emp start with ename='KING' connect by prior empno=mgr;



N.B.1 :- Oracle 9i introduced sys-connect-by-Path() which returns Path of the hierarchy in tree structure.

Syntax:-

(specify symbol)

sys-connect-by-Path(ename, 'delimitername' )

Eg:- select level, sys-connect-by-Path(ename, '/')  
from emp startwith mgr is null connect by  
prior empno=mgr;

Q: Write a hierarchy query to display the emp who are working under blake from emp table.

Ans) Select level, sys-connect-by-Path(ename, '→')  
from emp start with ename='BLAKE' connect  
by prior empno=mgr;

Ans) Select level, sys-connect-by-Path(ename, '→')  
from emp start with ename='MILLER' connect by  
empno = prior mgr;

Op:- 1 → MILLER

2 → MILLER → CLERK

3 → MILLER → CLERK → KING

○ PRIOR:- PRIOR is a currency operator which specifies 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.

e.g:- select level, 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 column

↳ select level, ename, connect-by-root ename from emp start with mgr is null connect by prior empno=mgr;

Mail LOCK:

24/03/14

↳ Locking is a mechanism to prevent another user excess ~~other user~~ resources.

↳ All DB systems having 2 types locks:-

↳ Row level locks.

↳ 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 lock & another user query the data but can't perform DML operation and also when we are using commit & rollback automatically locks are released.

### Session 1

scott

sql> conn scott/Tiger ↴

sql> Select \* from emp where deptno=10 for update;

sql>commit; [for Releasing lock] [we cannot perform DMLs]

### No wait :-

↳ This is an optional clause, used along with for update clause.

↳ when we are using this clause Oracle server automatically returns control to the current session, if another user not releasing clause also.

↳ in this case Oracle server returns an error:-

**[ORA-00054 : resource busy]**

### Session 1 scott

sql> conn scott/Tiger ↴

sql> select \* from emp where deptno=10 for update nowait;  
ORA-00054: Resource busy

### Session 2 soma

sql> conn soma/soma ↴  
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 using this lock another user query the data, & that can't perform DML operations.

No. of users locks the resources.

Syntax :-

```
lock table tablename in share mode;
```

### Session 1

SCOTT

sql> conn sys scott/tiger ↴

sql> lock table emp in share mode;

### Session 2

SOMA

sql> conn sys soma/soma ↴

sql> select \* from scott.emp; ✓

sql> lock table scott.emp  
in shared mode; ✓

sql> update scott.emp set

sal = sal + 100; [we cannot  
x 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;
```

### Session 1

SCOTT

sql> conn sys scott/tiger ↴

sql> lock table emp in  
share mode;

### Session 2

SOMA

sql> conn sys soma/soma ↴

sql> select \* from scott.emp; ✓

sql> lock table scott.emp in  
shared mode; x

NB:- In all DB systems, when we are using cursor locking mechanism all database servers internally 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 using 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

(attribute<sub>1</sub> datatype (size), attribute<sub>2</sub> 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 (col<sub>1</sub> datatype (size),  
col<sub>2</sub> datatype (size), ...,  
col nestedtabletype) nested table colname store  
as anyname;

e.g:- say create or replace type g1 as object(bookid  
number(10), bookname varchar2(10), Price number(10));

/  
say create or replace type f1 as table of g1;  
/

say create table student (stno number(10), name varchar(10),  
cols fan) nested table cols store as abc;

O/P:- desc student;

name	type
STNO	NUMBER(10)
NAME	VARCHAR(10)
COLS	FAN

→ If we 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, 'merali', fan(g1(  
101, 'java', 300), g1(102, 'pgsql', 500)));

say select \* from student;

number varchar(10)  
fan

stno	name	cols

BookId	BookName	Price
101	java	300
102	pgsql	500

NB:- if we want to view nested table data then  
we are using table operator.  
Syntax:-

Select \* from ~~student~~ table (select nestedtable colname  
from actual table);

say select \* from table (select cols from student);

BookId	BookName	Price
101	java	300
102	pgsql	500

21/08/14

Ques What are synonyms:-

Synonym is a db object which is used to provide security of the table becoz synonym hides another user object name.

↳ Synonym is a referenced name or alias name for the original names.

↳ All database system having two types of 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 using following syntax.

Syntax:-

create public synonym synonymname for ~~username~~ username. objectname @database link;

scott/Tiger

sql> Grant all on emp  
to murali@, soma;

Murali/murali

sql> select \* from scott.emp;

sql> create synonym cooler for scott.emp;

sql> select \* from cooler;

sql> create public synonym ab for scott.emp;  
error: insufficient privileges

sql> conn sys as sysdba;  
password: \* \* \*

sql> grant create public synonym to murali;

soma/soma

sql> select \* from scott.emp;

sql> select \* from cooler;  
error: Table or view does not exist

sql> select \* from ab;

(✓)

say connect mcaral/murali  
say create Public synonyms  
as for scott.emp;  
say select \* from ab;

## TCL (Transaction Control Language) :-

### Transaction:-

- It is an logical unit of work bet<sup>n</sup> two points.
- All db system having two transactional commands
  - 1) Commit
  - 2) Save Point

Commit:- This command is used to save the transaction permanently to disk (DB).

Syntax:-

commit;

Save Point:- It is an logical mark bet<sup>n</sup> transactions

Syntax:-

Savepoint savepointname;

Rollback:- This command 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;

Commit;

## Partitions :-

Partitions & tables are created by dba in recovery and backprocess.

- ↳ This process automatically improves performance of the application.
- ↳ partitions tables are used in datawarehousing application.
- ↳ Oracle having 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  
(key colname) (Partition Partitionname values  
less than (value), ...);
```

### 2) to view Particular Partitions :-

Syntax :-

```
select * from tablename Partition (Partitionname);
```

Say create table test (empno number(10), name varchar(10))  
Partition by range (sal) (Partition P1 values less than  
(1000), Partition P2 values less than (2000), Partition  
P3 values (3000));

12

Say insert into . . .

Say select \* from test Partition (P1);

## 2) List Partitions:-

Oracle q; introduce list Partitions, using this Partitions we can also create Partition table based on char datatype column, in this Partitions Partition tables are created by reserving 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:-

```
say create table test (sno number(10), name varchar(10))  
Partition by list(name) (Partition P1 values ('india',  
'Pakistan'), Partition P2 values ('us', 'uk', 'canada'),  
Partition others values (default));
```

say insert into . . .

say select \* from test;

<u>sno</u>	<u>sname</u>
1	Japan
2	aus.

## 3) hash Partitions:-

In this Method of Partitions are created by db server base on hash algorithm.

Syntax:-

```
Create table tablename (col1 datatype(size), col2 datatype  
(size), ...) Partition by hash(keycol) Partition no.
```

Eg:- Create table test (name varchar(10), sno number(10))  
Partition by hash(sno) Partition 5;

All Partition are stored under user-tab-partition data dictionary.

eg:- desc user-tab-partitions;

sql> 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 in place of null.  
syntax:-

NVL2(exp1, exp2, exp3)

↳ if expression  $exp_1$  is null, then it returns  $exp_3$   
otherwise it returns  $exp_2$ .

Ex:- select nvl2(null, 10, 20) from dual;

O/P:- 20

sql> select nvl2(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.

sql> select emp.comm

sql> update table set comm = nvl2(comm, comm+500, 500);

coalesce():-

This is an ANSI SQL which returns 1<sup>st</sup> normal expression.

eg:- Select coalesce(null, 20, null, 20, null) from dual;

O/P :- 20

↳ nvl function internally uses implicit conversion  
whereas coalesce() not using 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

## PL/SQL

24/08/14

↳ 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 cursor life cycle.

\* cursor for loops.

\* parameterise of cursor.

\* for update, where current of clauses

\* implicit cursor attributes.

3) ↳ EXCEPTIONS

→ \* Predefined, user defined, named exception

\* sql code, sqlerror.

→ raise application\_error()

4) ↳ SUBPROGRAMS

→ \* stored procedures

\* in, out, inout modes {Parameter modes}

\* no copy

\* autonomous transactions

\* authid current\_user

→ \* stored function

\* dmls used in functions.

\* function execution methods.

\* In - Concat()

## 5) TRIGGERS

- \* row level triggers
- \* applications
- \* before/after
- \* statement level triggers
- \* trigger execution order
- \* follows clause
- \* mutation error introduction
- \* system triggers.

## 6) PACKAGES

- \* introduction
- \* serially - reusable
- \* Overloading procedures
- \* forward declarations.

## 7) TYPES USED IN PACKAGE

- \* PL/SQL record
- \* index by table
- \* nested table
- \* varray
- \* collection methods.

## 8) BULK BIND

- \* BULK COLLECTION
- \* forall statements
- \* indicates of clause
- \* SQL%bulk - rowcovert

## 9) UTL - file package

## 10) LOB ( 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

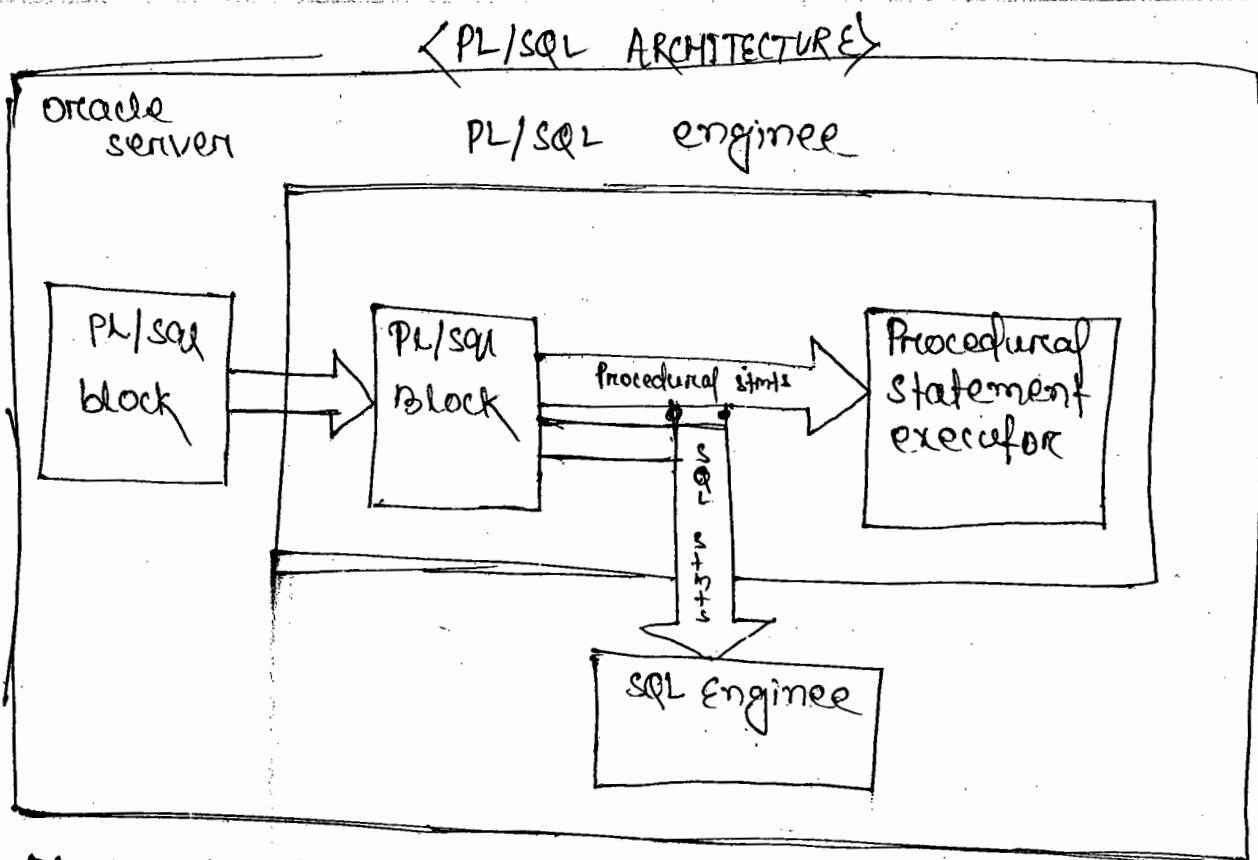
Oracle 7.0 → PL/SQL 2.0

Oracle 8.0 → PL/SQL 8.0

Oracle 11.2 → PL/SQL 11.2

↳ Basically PL/SQL is a block structure of Programming language.

↳ When we are submitting PL/SQL blocks into the Oracle server, all SQL statements are executed with in SQL enginee. and also all procedural statements are executed with in PL/SQL enginee.



Block Structured:-

Declare [optional]

- Variable declaration, 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 :-

↳ Using an assignment operator ( $:=$ ) we can store a value into variable

ex:-  $a \underset{\substack{\text{in oracle} \\ \text{number}}}{\text{number}}(10);$  |  $\overset{\substack{\text{in} \\ \text{int a;}}}{\text{a :=}}$  |  $a \boxed{50}$   
 $\downarrow$   
Assignment operator

a 100

$a = 50;$   
Assignment operator

Syntax :- variable\_name := value;

SQL declare

```
a number(10);
begin
  a := 50;
end;
```

Display a message or variable value :-

Syntax :- dbms\_output.put\_line('message');

Package name (or) Method name

dbms\_output.put\_line(variable\_name);

To view PL/SQL versions :-

↳ SQL Select \* from v\$version;

eg:- SQL set serveroutput on;

SQL begin

↳ if we don't write this then  
o/p doesn't show.

```
dbms_output.put_line('welcome to PL/SQL');
end;
```

/

O/P :- welcome to PL/SQL

Q:- ex:- 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 retrieve data from table and store 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. declare

```
vENAME varchar2(10);  
v-SAL number(10);  
begin  
select ename, sal, into  
v-ename, v-sal from emp  
where empno=8765;
```

25/03/14

dbms-output.Put-line

```
(v-name || ' ' || 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 declaring the variable in declare section of PL/SQL block.

e.g:- declare

```
a number(10) not null := 50;
```

```
b constant number(10) := &;
```

begin

```
dbms-output.Put-line(a);
```

```
dbms-output.Put-line(b);
```

end;

/

NB: we can also use default clause instead of assignment operator <sup>when we are assigning the value</sup> at the time of variable declaration in declare section of PL/SQL block.

e.g:- declare

```
a number(10) default 50;
```

begin

```
dbms-output.Put-line(a);
```

end;

/

O/P:- 50

Q.W.A. PL/SQL Program retrieve max salary from emp table and stored it in to PL/SQL variable and also display that salary.

Q) ~~A~~ declare

a number(10)

v\_ename varchar2(10);

~~me work~~

v\_sal number(10);

begin

Select ename, sal

& 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

↳ declare

v\_sal number(10);

begin

Select max(sal) into v\_sal  
from emp;

dbms-output. put-line(v\_sal);

/

O/P:- 5200

NB:- in PL/SQL expressions, we are not allowed to use group functions, decode conversion function within ~~PL/SQL expression~~ but we are allowed to use number functions, character functions, date functions in PL/SQL expressions.

e.g:- declare

a varchar2(10);

begin

a := upper('SOMA');

dbms-output. put-line(a);

end;

/

O/P:- SOMA

## Variable attributes :-

variable attributes are used instead of datatypes in variable declaration.

↳ These variable attributes are also called as anchor notations.

↳ When we are using these attribute Oracle server automatically allocates memory for the variables based on the column datatypes in a table.

↳ Oracle having 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 using percentage (%) type.

Syntax:-

variable name tablename.colname % type ;

↳ When we are using this attribute Oracle server automatically allocates memory for the variables based on corresponding col datatype from a table.

e.g:- declare

v-ename emp.ename%type,

v-sal emp. sal%type;

begin

select ename, sal into

v-ename, v-sal from emp

where empno=870;

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 <sup>row</sup> record within a table.
- This variable is also called as record type variable.
- it is also same as structures in C-language.
- These attributes are represented by using % rowtype syntax:-

variablename      tablename%rowtype;

e.g:- declare

```
i emp%rowtype;
begin
select ename , sal , hiredate , deptno
into i.ename , i.sal , i.hiredate ,
i.deptno from emp where empno=870;
dbms_output .put_line (i.ename || '|| '|| i.sal || '|| '
i.hiredate || '|| '|| i.deptno);
end;
```

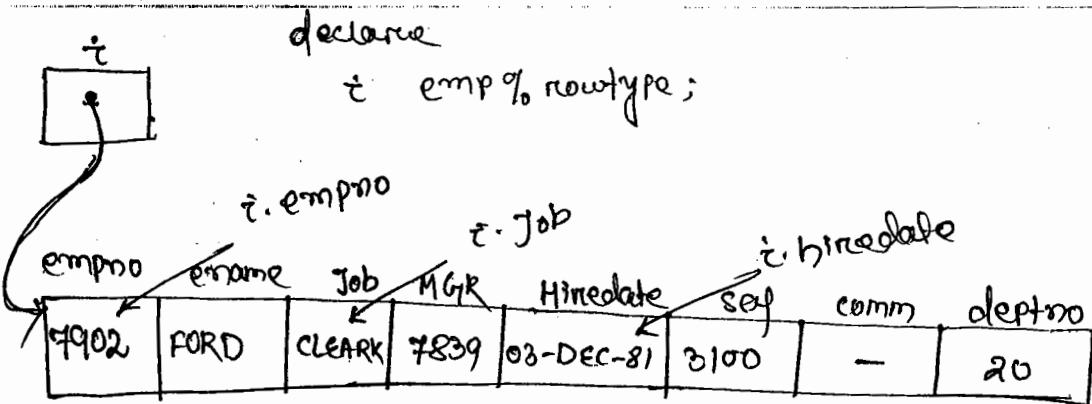
(alternative way)

declare

```
t emp%rowtype;
begin
select * into t from emp where empno=870;
dbms_output .out_line (t.ename || '|| '|| t.sal || '|| '
t.hiredate || '|| '|| t.deptno);
end;
```

O/P:- Enter value for no: 7902

FORD 3100 03-DEC-81 20



Maa

25/03/19

datatypes and variables :-

it supports all sql datatypes (also called as scalar datatypes). + boolean datatypes.

Composite datatypes

Ref obj.

Lobs (large objects) → clob, blob, bite

non Pl/sql variables (or) bind variables

Conditional statements :-

1) if

2) if-else

3) elseif

if :- syntax :- if condition then

    stmts;

    end if;

if-else :- syntax :- if condition then

    stmts;

    else

        stmts;

    end if;

elseif :- To check & more no. of conditions then we are using elseif.

syntax :- if condition then  
          stmts;

    elseif condition2 then

```
    stmts;  
elsef condition3 then  
    stmts;  
--  
--  
else  
    stmts;  
end if;
```

eg:- declare

```
    v-deptno    member(10);  
begin  
    select deptno into v-deptno  
    from dept where deptno=dno;  
    if v-deptno=10 then  
        dbms-output.Put-line('Ten'),  
    elsif 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;
```

op:- Enter value for no: 40  
others

sql/  
Enter value for no: 90  
error.

NB:- when PL/SQL block package select <sup>into</sup> clause  
is also of 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 doesn't return any error messages to handle these types of blocks we are using implicit cursor attributes.

e.g:- begin

```
delete from emp where  
ename = 'welcome';  
end;
```

or:- PL/SQL Procedure successfully completed.

NB3:- when ever select in to clause try to return multiple values at a time then oracle server returns an error:

ORA - 1422 : Exact fetch returns more than requested number of rows.

Control statements (Loops) :-

- 1) Simple loop
- 2) while loop
- 3) for loop

Simple loop :-

This loop is also called as infinit loop. Here what body of the loops statements are executed repeatedly. syntax

syntax:- loop  
" statements;  
end loop;

↳ To exit from infinit loop oracle provides two methods.

✓ 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 :- using 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 executed repeatedly until stmts is false.

syntax:- while condition  
 loop  
 stmts;  
 end loop;

eg:- declare  
 n number(10) := 1;  
 begin  
 while( $n \leq 10$ )  
 loop  
 dbms-output . put-line(n);  
 $n := n + 1$ ;  
 end loop;  
 end;

O/P :-  
 1  
 2  
 3  
 4  
 5  
 6  
 7  
 8  
 9  
 10

### for loop:-

syntax:- for index variable name in lower bound  
 .. upper bound loop  
 loop  
 stmts;  
 end loop;

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 10..1  
 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;

N.B. - ~~in~~ <sup>for</sup> one loops index variable internally behave like a integer variable.

26/03/14

Q. What are bind variables :-

- ↳ Bind variables are session variables.
- ↳ These variables are create a fast environment. That's why these variables are also called as 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.

Step 1:- creating a bind variable :-

Syntax:-

sql> Variable variablename datatype;

Step 2:- using bind variable :-

Syntax:-

: variablename.

Step 3:- display variable value :-

Syntax:-

sql> Print variablename;

e.g:- say variable g number;

say declare

a number(10) := 500;

begin

:g := a/2;

end;

say /  
say print g;

g

-----  
250

↳ PL/SQL having two types of blocks.

- 1) Anonymous blocks. [These blocks doesn't have a name.]
- 2) named blocks.

These blocks don't have a name.

- These blocks are not stored permanently into DB.
- Not to call in client applications.  
Ex:- declare  
begin

named blocks end;

These blocks having a name

• These blocks are automatically permanently stored in DB.

• These blocks are calling in client applications.

Ex:- Procedures, functions, Triggers, Packages.

\* 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 having two types of static cursors:-

1) implicit cursor.

2) explicit cursor.

implicit cursors :-

for SQL statements returns single record is called implicit cursor.

↳ implicit cursor memory area is also called as context area/SQL area.

↳ when an PL/SQL block contains select...into clause or DML statements then oracle server

① Automatically creates a memory area also called as context area or SQL area are implicit cursor.

- ↳ This memory area returns single record when PL/SQL block contains select... into clause.
- ↳ where as this memory area returns multiple records when PL/SQL block contains DML stmts. but these multiple records process at a time by SQL engine.

e.g) → SQL declare

```
v-ename varchar2(10);  
v-sal number(10);  
begin  
select ename, sal into v-ename, v-sal from emp  
where empno = 7902;  
dbms_output.out_line(v-ename || ' ' || v-sal);  
end;  
/
```

O/P:- Enter value for no : 7902.  
FORD 3000

② Explicit cursor

- ↳ For SQL statements 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 SQL area.

Explicit cursor lifecycle :-

1) declare    4) close.

2) open

3) fetch

1) declare :-

In declare section of the PL/SQL blocks they are defining the cursor using following syntax.

## Syntax :-

cursor cursorname is select statement;

ex:- declare

cursor c1 is select \* from emp where job =  
open :—

~~27 open :-~~

when ever we re open in the cursor then  
only oracle server transfer data from table  
into cursor memory area.

because in all database systems when we are open in the cursor then only cursor select stmts are executed.

## Syntax :-

Open [username](#);

↳ These statements used in executable section of the PL/SQL block.

INB @ whenever we open the cursor implicitly cursor pointers point to the 1<sup>st</sup> record of the cursor.

3) Fetch :- (fetching <sup>data</sup> from cursor)

using fetch stmts we are fetching data from cursor in to PL/SQL variables.

Syntax:-

~~fetch curusername into variablename<sub>1</sub>,variable  
name<sub>2</sub>,.....;~~

## 4) close:-

when ever we are close the cursor all the resources allocated from cursor memory area automatically released.

Syntax:- close cursorname;

e.g:- 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'||,
fetch c1 into v-ename,v-sal;
v-ename);
dbms-output.put-line(v-ename||','||high salary);
close c1;
end;
/
```

O/P:- SMITH 900

My second emp name is ALLEN  
WARD high salary.



Ques

## PL/SQL

07/06/19

Two types of blocks :-

### Types of blocks

1) Anonymous 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.

Declaring a Variable:-

Syntax:-

VariableName datatype(size);

ex:- PL/SQL

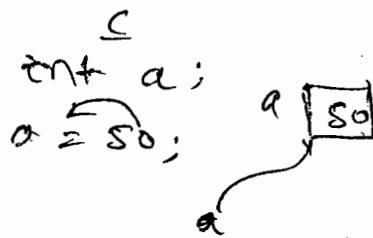
sql> declare

a number(10);

b varchar2(10);

    a := 50;

    b := a;



Storing a Value into Variable :-

Using assignment operator ( $:=$ ). we can stored a value in to 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 Package 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 retrieve data from table & stored into PL/SQL variables.

↳ Select....into clause always returns single record or single value at a time.

Syntax:-

Select col<sub>1</sub>, col<sub>2</sub> ... into var<sub>1</sub>, var<sub>2</sub> ...  
from tablename where condition;

↳ This clause is 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.

④ Set Serveroutput on;

say declare

```
V-ename varchar2(10);  
V-sal number(10);
```

begin

```
select ename, sal into V-ename, V-sal from  
emp where empno = <?>;
```

```
dbms_output.put_line(V-ename || V-sal);  
end;
```

/

O/P:- Enter <sup>Value</sup> for no: 7369  
SMITH 8000

emp		
empno	ename	sal
7369	SMITH	8000
7566	JONES	2000
-	-	-
-	-	-

Ans

Q My 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 || '||' || highsal);
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 cursor having 4 attributes

These are :-

- \* 1) cursor % notfound.
- 2) % found.
- 3) % isopen.

} (Returns boolean value  
True or false)

- \* 4) % Rowcount.

} (Returns number datatype)

When we are using these attributes in PL/SQL block then we must specify a cursor name along with these attributes.

Syntax:-

cursorname%attributename

↳ Except %Rowcount all other cursor attributes returns boolean value either true or false, whereas as %rowcount always returns number datatype.

1) % notfound :-

↳ This attribute always returns boolean values either true or false.

↳ This attribute always returns true if cursor does not have records after searching 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 using %notfound attribute ?

Q) declare  
 cursor c1 is select ename, sal from emp;  
 v-ename varchar(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  
 highest sal emp from emp table using  
%rowcount attribute.

Q) declare  
 cursor c1 is select ename, sal from emp;  
 order by sal desc;  
 v-ename varchar(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 using %rowcount attribute.

A) 

```
declare
cursor c1 is select ename,sal from emp;
v-ename varchar(10);
v-sal number(10);
begin
open c1;
loop
fetch c1 into v-ename,v-sal;
exit when c1%rowcount=0;
if mod(c1%rowcount,2)=0 then
dbms_output.put_line(v-ename||" "||v-sal);
end if;
end loop;
close c1;
end;
/
```

↳ whenever we are creating a cursor automatically 4 memory locations are created along with cursors memory area.

↳ These memory locations are viewed like a variable.

↳ These memory locations are identified through cursors attributes.

↳ These variables also stores single value at a time.

Ex:-

declare

a number(10);

b boolean;

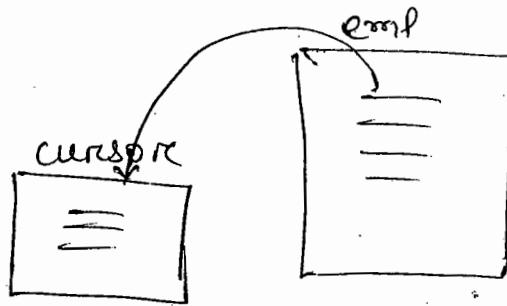
begin

a:=50;

b:= true;

a [50]

b [true]



c1% not found

True

c1% found

false

c1% is open

true

c1% row count

2

Notes

01/09/14

Q: W.A. PL/SQL cursor Program for Paking Job as a Parameter from emp table . display employee working as clerks or analyst and also display that report statically.

Q:- employee working as clerks

SMITH

ADAMS

JAMES

MILLER

employee working as analysts

SCOTT

FORD.

④ 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 working 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 ANALYST');  
loop  
fetch c1 into i;  
exit when c1%notfound;  
dbms-output.put-line(i.ename);

```
end loop;  
close q;  
end;  
/
```

N.B: Before we are reopening cursor then 1st we must close properly otherwise oracle server returns an error: **ORA-6511 : cursor already open.**

N.B 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 Practically using cursor for loop.

```
declare  
cursor c1(p-Job varchar) is select * from emp  
where Job = p-Job;  
begin  
dbms-output.put-line('employees working as Clark');
```

```

for i in g('clerk').
loop
dbms_output.put_line(i.ename),
end loop;
dbms_output.put_line('employees working as analysts clerks'),
for i in g('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 retrieve all dept no from dept table using a static cursor and also Passes these deptnos from this cursor in to another Parameterised cursor which returns employee details from emp table based on passed dept no.

④ 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;

```
end loop;  
end loop;  
end;  
/
```

N/B :- we can also pass default values in Parameterised cursor using default (or) ~~:=~~ :=. becoz by default cursor Parameters are inparameters.

Syntax :-

Parametername datatype default [:=] actualvalue

```
declare  
cursor c1(P-deptno number default 20) is select *  
from emp where deptno = P-deptno;  
type emp%rowtype;  
begin  
open c1;  
loop  
fetch c1 into t;  
exit when c1%notfound;  
dbms_output.put_line(t.ename||' ||t.deptno);  
end loop;  
close c1;  
end;  
/
```

Note

(RD-15) 15/06/14

percentage :-

This attribute always returns no. datatype i.e. it returns no. of records number fetched from cursor.

ex:-

```
declare
cursor c1 is selected ename, sal from emp;
v-ename varchar(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%row-
count);
close c1;
end;
/
```

O/P:- SMITH 1500

ALLEN 1200

number of records fetched from the cursor 2.

Note:-

Cursors also used to transform data from one oracle table in to another oracle table.

Q.W.A PL/SQL cursor program to transform employees who are getting more than 2000 salary in emp table.

④ declare

say create table target (sno number(10), name  
varchar2(10), salary number(10));

sql> 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 attribute 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||' is open');
```

```
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 fetching at least one row from the cursor.

Q) W.A. PL/SQL cursor Program to display all employee names & salary from emp table using %found attribute.

```
④ 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 (vENAME || ' ' || vSA);
            fetch c1 into i;
        end loop;
    close c1;
end;
/
```

O/P:- ~~923#~~      ename      sal  
          SMITH      8000  
          JOEWS      5000  
          ;  
          ;

Q) W.A. PL/SQL cursor program to display total salary from emp table without using ~~some~~ <sup>sum()</sup> function

P) ~~declare~~ cursor c1 is select \* from emp;  
~~type~~ v-sal number(10);  
v-name varchar2(10);  
v-saf 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

Rd-17 14/06/14

Q) max

eliminating explicit cursor life cycle (or) cursor  
for loop :-

when we are using cursors for loop internally  
Oracle server only open the cursor fetched  
records from the cursor & closed the cursor  
automatically.

This is also called shortcut method of the cursor.  
cursor for loop syntax :-

```
for indexvariablename in cursorname
loop
  statements;
end loop;
```

PL/SQL

This loop used in executable section of the  
blocks.

Note :-

\* in cursor for loops index variable internally  
behaves like a record type variable (%rowtype).

ex:-

```
declare
  cursor c1 is select * from emp;
begin
  for r in c1
  loop
    dbms_output.put_line(r.ename || ',' || r.sal);
  end loop;
end;
```

Note: we can also eliminate declare section of the cursor for loop. in this case we must specify cursor <sup>select stmts</sup> in place of cursor name within cursorforloop.

Syntax:

```
for indervarname 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;
n number(10) := 0;
begin
for i in 1
loop
n:=n+nvl(t.sal,0);
end loop;
dbms_output.put_line('total salary is
' || n);
end;
```

1

O/P:- total salary is 34075

Parameterised

cursor constructor :-

we can

also pass parameters into cursor

same like a subprograms to parameters.

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, functions then we are not allowed to use datatype size in formal parameter declaration.

Syntax :-

format Parameter

- 1) cursor cursorname (Parametername datatype)  
is select \* from tablename  
where columnname = Parametername
- 2) Open cursorname (actual Parameters);

ex:-

```
declare
cursor q (P_deptno number) is select * from emp
where deptno=P_deptno;
    t emp%rowtype;
begin
open q(10);
loop
fetch q into t;
exit when q%notfound;
dbms_output.put_line(''||t.empno|| ''|| t.ename|| ''|| t.sal|| ''|| t.deptno);
end loop;
close q;
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 statrical by.

ex:- employee working as clerks

SMITH

ADAMS

JAMES

MILLER

employee working as analyst

SCOTT  
FORD.

④ declare

cursor c1 (l-job varchar2)

is select \* from emp where job = l-job;

type emp%rowtype;

begin

open c1 ('CLERK');

dbms\_output.put\_line('employees working as 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('employees working as analyst');

loop

fetch c1 into :i;

exit when c1%notfound;

dbms\_output.put\_line(:i.ename);

end loop;

close c1;

end;

/

Note:-

Before we are reopen the cursor ~~it~~ it 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 trying to perform operations on the functions then oracle server returns an error ORA-1001: invalid space cursor.

May

RD-19

16/06/14

↳ In oracle we can also pass default values in parameterized cursor using default or operator.

Syntax:-

Parametername datatype default{:=} actual value;

ex:- declare

```
cursor q1(p_deptno number default 30) is
select * from emp where deptno = p_deptno;
begin
open q1;
loop
fetch q1 into i;
exit when q1%notfound;
dbms_output.put_line(i.ename||'-'||i.sal||'
'||i.deptno);
end loop;
close q1;
end;
/
```

ex:-

```
declare
cursor q1(p_deptno number) is select * from
emp where deptno = p_deptno;
begin
for i in q1(10);
loop
dbms_output.put_line(i.ename||'-'||i.sal||'
'||i.deptno);
end loop;
end;
```

Ex:-

```
declare
cursor c1(p_job varchar) 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.

Ex:- W.A PL/SQL program using a static cursor retrieve all department nos from dept table & pass these department nos to another parameterised cursor to retrieve employee details from emp table based on pass deptno.

By

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 1
    loop
      dbms_output.put_line ('my dept table dept no
                            is ' || i || ' i.deptno);
    for j in c2(i.deptno)
      loop
        dbms_output.put_line (j.ename || ' ' || j.sal
                            || ' ' || j.deptno);
      end loop;
    end loop;
  end;
/

```

update, delete statements using cursor  
(without using for update, where clause)  
 current of clauses) :-

Q) w.A PL/SQL cursor Program modifying salaries  
 of the employee in ~~employee~~ table  
 on following condition.

- 1) if job = 'CLERK' then increment sal → 100
- 2) if job = 'SALESMAN' then decreament sal → 200

R) declare  
 cursor c1 is select \* from emp;  
 type emp%rowtype;  
 begin  
 open c1;  
 loop  
 fetch c1 into r;  
 exit when c1%notfound;

```

④ if i.job = 'CLERK' then update emp set sal =
    i.sal+100 where empno = i.empno;
else if i.job = 'SALESMAN' then update emp set
    sal = i.sal-200 where empno = i.empno;
end if;
end loop;
end cursor;
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 areas returns single record.
- ↳ This memory areas returns multiple records when PL/SQL block contains select...into clause.
- ↳ This memory areas also returns multiple records when PL/SQL block contains pure DML stmts but these multiple records are processed at a time by using SQL engine.

↳ Along with SQL area oracle server automatically creates 4 memory location these memory locations are behaves like a variable and also these variables are identified by thorough implicit 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 member datatype.

Ex:-

```
begin
    delete from emp where ename='welcome';
    if SQL%found then
        dbms_output.put_line ('a record deleted');
    end if;
    if SQL%notfound then
        dbms_output.put_line ('no record does not
                                exists');
    end if;
end;
/
```

op:- no 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 it occurs use an appropriate exception name in exception handler under exception section.

Oracle having 3 types of exception,

- \* Predefined Exception
- \*\* Userdefined Exception
- \*\*\* Unnamed Exception.

### \* Predefined Exception :-

Oracle provides 20 predefined exception names for regularly occurring runtime errors.

whenever ever there runtime errors occurs use an appropriate predefined exception name in exception handler under exception section within PL/SQL blocks.

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 ends 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 provides '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;

/

Q/P :- Enter value for no : 111  
your employee does not exist.

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 return more-rows');
```

end;

/

3) zero-divide :-

20/06/14

In oracle when we are trying 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

ex:- begin

```
dbms_output.putline($10);
```

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;  
type emprowtype;
```

begin

loop

fetch c1 into r;

exit when c1%rownum=1;

dbms\_output.putline (r.ename || ' ' || r.sal);

end loop;

close c1;

exception

when invalid\_cursor then

dbms\_output.putline ('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.

```

① exr 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 errors,

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 date string into Date type then Oracle server returns an error, [ORA-01722 : invalid number.]

To handle this error oracle provides 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('enter 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:= '12x'+'y';

dbms-output.put-line(z);

exception

when value-error then

dbms-output.put-line('enter proper data only');  
end;  
/

## Note

In Oracle when we are trying to store more data than the Datatype size in varchar datatype  $\Rightarrow$  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 exception name.

declaration

```
z varchar(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 trying to store more data than of datatype size  $\Rightarrow$  number NUMBER datatype declared in variable declaration, then oracle server returns an error,

ORA-6502 : numeric or value error : number  
~~Precision~~ Precision too large

To handle this error also we are using Value-error exception name.

```

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 string buffer too small.

Exception propagation :-

In PL/SQL exception also ~~basef~~<sup>raised</sup> 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 exception 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;

1

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.

Step 1 :- declare

Step 2 : raise

Step 3 : handling exception.

Step 1 :- declare

In declare section of the PL/SQL blocks we are creating our own exception name using EXCEPTION predefined type.

Syntax :-

userdefinedname exception;

27

S. 100

~~stop~~ declare

a exception;

## Step 2 :- raise

using raised 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 : handling exception

We can also handle user defined exception as same as predefined exception using exception handlers.

Syntax :-

when userdefinedexception name1 then

stmts;

when redefined exception name 2 then

Stamps 3

when others then

Sammels;

Q) In PL/SQL Program raise userdefinedexception  
of SATURDAY?

A) 

```
declare
  a exception;
begin
  if to_char(sysdate,'DY') = 'SAT' then
    raise a;
  end if;
exception
  when a then
    dbms_output.put_line('my exception raised
      on saturday');
end;
```

O/P:-  
my exception raised on saturday.

Ex:-

```
declare
  v-sal number(10);
  a exception;
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 (t%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;

## nesting of exception :-

declare

$z_1$  exception;

$z_2$  exception;

begin

    begin

        raise  $z_1$ ;

    exception

        when  $z_1$  then

            dbms\_output.put\_line(' $z_1$  is handled');

        raise  $z_2$ ;

    end;

    exception

        when  $z_2$  then

            dbms\_output.put\_line(' $z_2$  is handled');

    end;

end;

/

O/P:-  $z_1$  handle

$z_2$  handle

PL/SQL Procedure successfully completed

### 3) Renamed Exceptions :-

In Oracle if we want to handle other than Oracle 20 predefined exceptions then we are using renamed method.

↳ In this method we are <sup>Creating</sup> ~~for~~ our own exception name and associates this exception name with appropriate error no. using 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.

↳ whenever we are using this Pragma Oracle server associate error no. with exception name at compile time only.

e.g :-

```

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.

```
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 with error messages then we are using 2 predefined error trapping functions.

These are :-

- \* SQLCODE
- \* SQLERRM

### \* SQLCODE & \* SQLERRM

it returns numbers whereas SQLERRM returns error numbers with error messages.

These two functions are used in 'where', 'otherwise', 'then', 'else' clause or used in our own exception handlers.

Note:-

\* if we want to catch error numbers at runtime then we are using SQLCODE function.

Note:-

SQLCODE always returns numbers.

### SQLCODE returns values

### Meaning

0	→ no oracle errors.
-ve	→ oracle errors
100	→ no-data-found
1	→ user defined exception

Ques

24/06/14

Note:-

In Oracle we can also Predefined, userdefined function in INSERT stmts. But we are not allow to pass SQL code, sqleerrm 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 renamed exception using save 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);
```

(-20,000 to -20,999) 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, 'my exception raised
on tuesday');
end;
```

O/P:- ORA:- 20123 : My exception raised on tuesday.

Note:-

In oracle this function also used in triggers because this function stop invalid data entry based on the condition when condition is true, whereas dbms-output 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 return a value)
  - 2) Functions (must returns a value)

### 1) Procedures

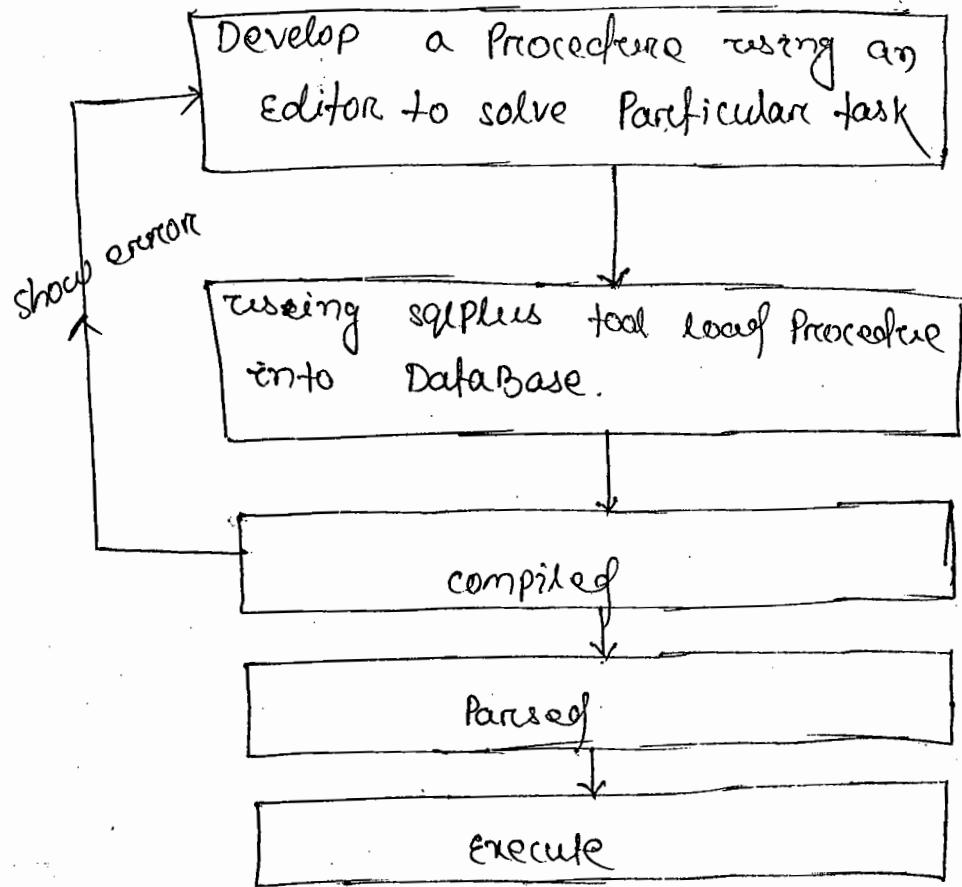
- Procedure is a named PL/SQL block used to solve particular task. Procedures may or may not return values.
- In oracle whenever we are using create or replace keyword in front of the procedure, these procedures are permanently stored in DB that's why that procedures are also called as stored procedure.

Ques?

RD - 29

25<sup>th</sup>/JUL/14

- ↳ Generally Procedures are used to improves performance of the application becoz procedure internally having one time compilation.
- ↳ In all improved DB one time compilation Program units Performance of the application.



Procedure having two parts

1) Procedure specification

2) Procedure module

4) In Procedure specification we are specifying name of the Procedure and type of the Parameters whereas as Procedure body we are solving actual task.  
syntax :-

create [or replace] Procedure procedurename(  
Parameter)  
IS/ AS  
Variable declarations, cursors,  
were defined exceptions;  
Begin  
[Exception]  
End [Procedurename];

Syntax of (formal Parameter) :-

Parametername [mode] datatype

in  
out  
inout

To view errors :-

sql> show errors;

executing a Procedure :-

Method 1 :-

Syntax :-

sql> exec Procedurename (actual Parameters);

Method 2 :- (using anonymous blocks):

Syntax :-

sql> begin

Procedurename (actual Parameters);

end;

/

Method 3 :-

Syntax :-

sql> call Procedurename (actual Parameters);

Q) If 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)

as

```
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 :-

Method 1 :-

```
sql> exec p1(7369);
```

SMITH 8000

Method 2 :- (Using anonymous blocks)

sql> begin

```
    p1(7566);
```

end;

JONES 2000

Method 3 :-

```
sql> call p1(7902);
```

FORD 3000

↳ In Oracle all Procedure information stored in user-procedures, user-source data dictionary.

Note :-

\* if we want to view code of the procedure then we are using user-source data dictionary

ex:- desc user\_source;

```
sql> 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.

P) create or replace Procedure

P2(P-dept number)

is

cursor c1 is select \* from emp  
where deptno = p-deptno;

is emp%rowtype;

begin

open c1;

loop

fetch c1 into i;

exit when 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 8600 10

MILLER 4100 10.

Mac

26/06/14

```
sql) CREATE OR REPLACE PROCEDURE P3(P_emphno Number, P_incnumber)
IS
begin
    UPDATE emp SET
        sal = sal + P_incr
    WHERE empno = P_emphno;
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.

↳ Formal Parameters :-

Formal Parameters must be specified in Procedure specification.

↳ Formal parameters specifies name of the parameter, mode of the parameter, type of the parameter.

Note:- In Oracle we are not allowed to use datatype declaration.  
size in formal Parameter when we are declaring cursors, procedures, functions.

ParameterName [Mode] datatype

### Mode :-

- Mode specifies purpose of the Parameters.
- 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 pass the value into Procedure body.
- This Parameter behaves 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?

R)

```
CREATE OR REPLACE PROCEDURE  
P1 (P_DeptNo in NUMBER, P_Dname in VARCHAR2, P_Loc in  
VARCHAR)  
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  uninitialized 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 anonymous 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 taking empname as 'in' Parameter from Emp table and return salary of the employee using 'out' Parameter.

④ CREATE OR REPLACE Procedure

```
P1 (P-Ename IN varchar2, P-sal OUT NUMBER)
IS
begin
SELECT sal INTO P-sal
FROM emp
WHERE empname = 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;
/
```

Note

27/06/14

Q) write a PL/SQL , stored Procedure for taking dept no as in Parameter and return dname, location using out Parameter from dept no.

④ Create or replace procedure

```
P1 (P-deptno IN number, P-dname OUT varchar2,
    P-loc OUT varchar2)
```

18

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 pl(10,:a,:b);

sql> print a b;

O/P :- A

ACCOUNTING

B

NEWYORK

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

SQL create or replace Procedure

```
    pi(p_deptno in number, p_count out number)
is
begin
    select count(*) into p_count
    from emp
    where deptno = p_deptno;
end;
```

### Execution

Say variable a Number;

Say exec (10, :a);

Say Print a;

Opp:- 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 specifies when we are  
modifying formal Parameter then actual Parameters  
are affected or not.

9 In Pass by value method actual value does not  
change in main Program, when we are modifying  
formal Parameter also because internally copy of the  
values are passed into executing calling program.

If we want to change actual value according to formal Parameter modification then we are using Pass by reference method.

### Nocopy :-

Oracle also support these two passing Parameter mechanism when we are using subprogram Parameters in oracle by default all in Parameters internally uses Pass by reference method. whereas as by default all out Parameters internally - uses Pass by value method.

In Oracle whenever 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 , l-sal out nocopy number)

is

begin

select sal into l-sal from emp where ename = p-name;

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's body?

#### Syntax:-

Parametername in out datatype

Ex— create or replace procedure P1(4 in out number)  
is  
begin  
a := a \* a;  
end;

#### Execution

method 1: (bind variable)

SQL> variable z number

SQL> exec P1(:z);

SQL> print z;

O/P:- z  
81

method 2: (using anonymous block)

SQL> declare

x number(10) := 8;

begin

P1(x);

dbms\_output.put\_line(x);

end;

O/P:- enter value for x:3

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  $P_1$  ( $P_x$  is not member)

is

begin

select sal into  $P_x$  from emp where empno =  $P_x$ ;

end;

Execution :- (using bind variable) :

SQL> variable  $Z$  number;

SQL> exec : $Z$  := 7902;

SQL> exec  $P_1$  (: $Z$ );

SQL> print  $Z$ ;

$Z$

3850

Autonomous concept / transaction :-

Autonomous transactions are independent transactions used in procedures, triggers.

Generally Autonomous transactions are used in child procedures

when ever we are calling autonomous procedures in PL/SQL block & also when we are using rollback in PL/SQL block then autonomous 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 [procedure-name]  
(formal parameter)

is/as

Pragma autonomous-transaction;  
begin

commit;

[exception]

end [procedure-name];

Ex:-

SQL> create table test (name varchar(10)),  
using autonomous transaction;

SQL> create or replace procedure P,

is

Pragma autonomous-transaction;  
begin

```
insert into test values ('india');
commit;
end;
/
```

```
SQL> begin
      insert into test values ('hyd');
      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 in

```
SQL> Create or replace Procedure p1
```

as

begin

```
      insert into test values ('india');
```

```
      commit;
```

```
      end;
      /
```

Procedure created.

SQL begin

insert into test values ('hybdi')

insert into test values ('mumbai');

PI;

rollback;

end;

/

SQL select \* from test;

Name

hybdi

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 in Procedures.  
authid current-user :-

→ Data Procedure having authid current-user clause then that Procedure not executed by another user if you are giving Privileges 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

- ~~on~~ Procedures.
- ~~by~~ 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 [privilegation] to another user:-

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> conn murali/murali

set server output on;

SQL> exec scott.P1(7902);

Error: Table or view does not exists.

25

30/06/14

Maan

Handled (or) unhandled 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 :-

say 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 :-

say create or replace procedure

P2

is

begin

P1 (5,0);

exception

when others then

dbms\_output.put\_line ('any error');

end;

/

say exec P2;

b can not be zero;

### Execution Methods :-

#### IN Parameter :-

Procedure IN Parameter having 3 types of execution method.

1) Positional Method notations. (ex:-

2) Named notations. (ex:- exec P1(1, 'a', 'b'), )

3) Mixed notations:- { SQL exec P1(P-dname → 'x',  
P-Loc → 'y', P-depth → 2), }

it is a combination of Positional and named notations.

#### Note:-

After positional there can be all named notations but after named there can not be positional.

say exec

P1(3, P-Loc → 'y', P-dname → 'x'),

We can also drop procedure using drop procedure function.

function.

## FUNCTIONS

- ↳ function is a named PL/SQL blocks which is used to solve particular tasks & also functions must return a value.
- ↳ Function also having two parts.
  - 1) function specification.
  - 2) function body.
- ↳ In function specification we are specifying name of the function & type of the parameter whereas in the function body, we are start solving actual tasks.

### \* Syntax:-

```
create or replace function Functionname(  
return datatype  
is/as
```

variable declarations, cursors, userdefined 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 having IN Parameters then  
those type of function then they are executing.  
using select stmts.

#### Syntax:-

```
Select functionname(actual Parameters) from deaf;
```

### Method 2 (Using anonymous blocks) :-

#### Syntax:-

```
sql begin
```

```
variablename := functionname(actual Parameters);  
end;
```

#### Ex:-

```
Create or replace function f1(a varchar)
```

```
return varchar
```

```
is
```

```
begin
```

```
return a;
```

```
end;
```

```
/
```

#### Execution:-

### Method 1 :- (using select statement) :-

```
sql> select f1 ('hi') from deaf;
```

```
hi
```

### Method 2 :- (using anonymous block) :-

```
sql>
```

```

SQL declare
  x varchar2(10);
begin
  x := f1('welcome');
  dbms_output.put_line(x);
end;
/
QP:- 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.

R) 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;
 /

Method 1 Execution :-

Method 1 (using select statement) :-

SQL select f1(5) from dual;

odd

Method 2 (using anonymous block) :-

SQL declare

x varchar2(10);

01/08/14

begin  
z := f<sub>1</sub>(4);  
dbms\_output.Put\_line(z);  
end;  
/

o/p:- even.

Method 3 (Using bind variable) :-

say variable x varchar<sub>2</sub>(10);

say begin  
:x := f<sub>1</sub>(8);  
end;

say Print x;

x

o/p:- even.

Method 4 :-

SQL> exec dbms\_output.Put\_line(f<sub>1</sub>(7));

o/p: odd

~~REVERSE~~

Method 5 :-

say begin  
dbms\_output.Put\_line(f<sub>1</sub>(8));  
end;

o/p:- even

Note:-

we can also call user-defined functions into  
insert stmts.

say create table test (emp varchar(10));  
say insert into test values (f1(6));  
say select \* from test;  
o/p:- even.

DML stmts used in functions :-

Generally when a function having DML stmts those functions are not executed using Select stmts, but we are executing those function using anonymous blocks.

If we want to execute those select stmts then we must use ~~an~~ autonomous transactions with in functions.

Ex:-

```
say 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)

say select f1(1) from dual;

Error: cannot perform a DML operation inside a query.

Method 2 (using anonymous block):-

say declare

a number(10);

begin

a := f1(1);

dbms\_output.put\_line(a);

end;

/

o/p:- 0

Using autonomous transactions:-

Ex:-

say create or replace function

f1(p\_empno number)

return number

as

v\_count number(10);

Pragma autonomous\_transaction;

begin

delete from emp where empno = p\_empno;

v\_count := sql%rowcount;

commit;

return v\_count;

end;

/

Execution:-

say 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 along with non-aggregate functions.

ex:-

```
create or replace function f,
return number
is
v-sal number(10);
begin
select max(sal) into v-sal from emp;
return v-sal;
end;
```

/

Execution:-

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.

R)

```
create or replace function f,
return number
is
z number(10);
begin
select month_between
(P-date,hiredate)/12 into z
from emp
where empno = P-empno;
return round(cx);
end;
```

### Execution :-

Select empno, ename, hiredate, f1(empno, sysdate)  
 "11' 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 subProcedure executed using select stmts.

### Oracle 11g:-

Select empno, ename, hiredate, f1 (P-empno  $\Rightarrow$  empno,  
 P-date  $\Rightarrow$  sysdate) "11' years" "exper" from  
 emp where empno = 7566;

Max

(37) 02/07/14 :

Q) Write a PL/SQL stored function for passing empno as a parameter then return job of the emp from emp table based on passed parameter.

Ans)

```

create or replace function f1 (P-ename number)
return number varchar2;
is
v-empno number varchar2(10)
begin
select job into v-job from emp where
ename = P-ename;
return v-job;
end;
/

```

Execution :-

say select si ('SMITH') from dual;  
O/P - CLERK.

Q) w.A. PL/SQL Stores 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 > 15000 then  
tax → 20% of annsal

3) if annsal > 20000 then  
tax → 30% of annsal

else

tax → 0

B) 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 > 10000 and annsal  $\leq$  15000 then

z : = annsal \* 0.1 ;

else if annsal > 15000 and annsal  $\leq$  20000 then

z : = annsal \* 0.2 ;

```

        elseif annsal > 20000 then
            z := annsal * 0.3;
        else
            z := 0;
        end if;
        return z;
    end;
    /

```

execution :-

sav>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 having Out Parameters in Out Parameter those functions are not allowed to execute using select stmts.

Ex w.A. PL/SQL stored function for Passing Deptno as a parameter then returning dname, location using out parameters from dept table,

④ 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;

exception :— (using bind variable)

say variable a varchar2(10);

say variable b varchar2(10);

say variable c varchar2(10);

say begin

:a := f1(10, :b, :c);

end;

/

say 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 aggregate functions same like a Predefined aggregate function and also those aggregate fun<sup>n</sup> returns multiple values then Only we are using cursors with in user-defined functions.

we can also used these user-defined aggregate function same like a Pre-defined aggregate fun<sup>n</sup> in group by clause.

Ex:-

```
create or replace function f1(p_deptno number)
returns 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 , nm\_concat(hiredate) emp  
group by deptno

↳ In oracle, all <sup>stored</sup> 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 usersource

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

/

↳ We can also drop function by using:

drop function function\_name;

Note

## TRIGGERS

(39) 03/07/14

↳ Triggers is also same as stored procedure &  
it will automatically invoke when ever  
DML ~~operation~~ performed on table or view.

↳ All database system having two types of  
triggers:

- 1) statement level triggers.
- 2) row level triggers.

↳ In statement level trigger, trigger body  
executed only once per DML stmts.

↳ whereas as a row level triggers, triggers  
body executed for each row for DML stmts.

↳ Triggers also having two parts

1) triggers specification.

2) triggers body.

Syntax: — Trigger Timing → Trigger events

create or  
before / after

replace Trigger Triggername

insert / update / delete on tablename

Trigger specification. [for each row] → for row level Trigger.  
[when condition] →

[Declare]

variable declaration, cursors, user defined exception

Begin

—

—

end;

Difference bet<sup>n</sup> stmts level, row level trigger :-

stmts level trigger :-

say create or replace trigger t1  
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 trigger body is executed for each row DML ~~statements~~ that's 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 ~~column~~ in front of the qualifier name.

Syntax:-

:old . columnname

Syntax:-

:new . columnname

operator	insert	update	delete
:new	✓	✓	✗
:old	✗	✓	✓

Q) W.A. PL/SQL row level trigger on emp table  
when ever user insert in salary, then  
salary should be more than 5000.

(P) 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);  
o 1 row created.

Q) w-A. PL/SQL row level trigger using emp, dept table implement on delete cascade concept with out using on delete cascade clause.

R) create or replace trigger t4  
after delete on dept  
for each row

```
begin  
delete from emp where deptno = :old.deptno;  
end;
```

testing:-

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.

R) create or replace trigger t5

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 whenever user modifying deptnos, then automatically those modification are effected in emp table.

④ create or replace trigger t6

after update on dept

for each row

begin

update emp set deptno := new.deptno where  
deptno = :old.deptno;

end;

/

testing:-

say update dept set deptno=1

where deptno=10;

say select \* from dept;

say select \* from emp;

now

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 commission of the ANALYST, then automatically user entered commission set to null.

Q) 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, whenever we're inserting data into employee no column, that column doesn't accept duplicate values. i.e. it acts like a primary key behaviour.

Q) 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 ('a record inserted');
end if;
end;
/
```

### testing :-

say insert into emp(empno) values(7902);

ORA-20123 : 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.

```
④ Create or replace trigger tks
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 modified data in a  
column those transaction  
another table is called auditing a column.  
↳ auditing a column app must be implemented  
using rowlevel triggers.

### Ex:-

say create table target (oldempno number (10),  
oldsalary number (10),  
newsalary number (10),  
coly date, reser name  
username varchar(10)),

SQL> create or replace trigger tkg before update  
of sal on emp for each row  
for each row

begin

insert into target (oldempno, oldsalary,  
newsalary, coly, username) values (  
:old.empno, :old.sal, :new.sal, sysdate, reser);

end;

1

Testing:-

say update emp set sal=sal+10 where dname='INN';  
SQL> select \* from target

In Oracle we are not allowed to use  
:new, :old qualifiers in front of the sysdate,  
use pseudo columns.

What auto increment :-

(43) 05/07/24

If we want to generate primary key values  
automatically, then all database system uses  
autoincrement concept.

In Oracle we are implementing autoincrement  
concept by 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 try

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 occurred)  
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 in DB. before this process these values are effected in either in trigger specification or trigger body then only those values are effected in DB. That's why in Oracle when ever we are modifying :new qualifier values in row level triggers 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 transform data <sup>we are</sup> one table into another table then only using after timing.

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.

R) Create or replace trigger try  
before insert on emp  
for each row  
begin  
:new.ename := upper(:new.ename);  
end;  
/

Testing:-

SQL> insert into emp(empno,ename) values(1,'soma');

SQL> select \* from emp;

empno	ename
1	SOMA

08/07/14

Q Mac

Q ?

Ex:-

SQL> create or replace trigger t1  
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 & deleting clauses.

These events are used in either in  
stmts level triggers or in row level  
triggers.

syntax:—

```
if inserting then  
stmts;  
else if updating then  
stmts;  
else if deleting then  
stmts;  
end if.
```

Q) W.A. PL/SQL stmts level trigger on emp table  
not to perform DML operations in any  
days.

A) create or replace trigger tJ2  
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 update');  
elseif deleting then  
raise\_application\_error(-20749,'we can  
not perform deletion');  
end if;  
end;

## testing :-

SQL> delete from emp where empno=7902;  
ORA-20749: we cannot perform deletion.

## ex:-

```
SQL> create table test (msg varchar2(20));  
SQL> create or replace trigger t3  
after insert or update or delete on emp  
declare  
    z varchar2(20);  
begin  
    if inserting then  
        z := 'rows inserted';  
    elsif updating then  
        z := 'rows updated';  
    elsif deleting then  
        z := 'rows deleted';  
    end if;  
    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

SQL> W.A.P/PLSQL row level trigger on emp table.  
whenever user inserting data, on emp table  
then those values are automatically stored  
in another table, whenever user modifying  
data those transactional data automatically  
stored in another table, whenever user  
deleting data those deleted data also  
stored in another table.

SQL> create table target<sub>1</sub> (empno number(10),  
name varchar<sub>2</sub>(10),  
salary number(10));

SQL> create table target<sub>2</sub> (empno number(10),  
name varchar<sub>2</sub>(10),  
salary number(10));

SQL> create table target<sub>3</sub> (empno number(10),  
name varchar<sub>2</sub>(10),  
salary number(10));

SQL> create or replace trigger t5  
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);
else if updating then
insert into target2(empno, name, salary) values
(:new.empno, :old.ename, :old.sal);
else if deleting then
insert into target3(empno, name, salary) values
(:old.empno, :old.ename, :old.sal);
end if;
end;
```

1

testing:-

```
SQL> delete from emp where sal > 4000;
SQL> select * from target3;
      1 row deleted.
```

Trigger Execution Order :-

- 1) before statement level.
- 2) before row level.
- ~~3)~~ after row level.
- 4) after statement level.

Ex:-

```
SQL> create table test(sno number(10));
SQL> create or replace trigger tr1
after insert on test
for each row
begin
```

```
dbms_output.put_line('after row level');
end;
/
```

SQL> create or replace trigger tr2

before insert on test

for each row

begin

```
dbms_output.put_line('before row level');
end;
/
```

SQL> create or replace trigger tr3

after insert on test

begin

```
dbms_output.put_line('after statement
level');
end;
```

SQL> create or replace trigger tuy

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.

- Q What follows clause (Oracle 11g) :-
- ↳ In Oracle whenever we are using same table or same level when we can not control execution out of the trigger to overcome 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 explicitly we are providing guarantee execution order of the trigger.

Ex:-

```

SQL> create table test(sno number(10));
SQL> create or replace trigger tbs
      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 ~~create~~ trigger tb1  
 after insert on test

begin

```
dbms_output.put_line('trigger1 fired');
end;
/
```

④ 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, whenever user deleting data records on emp table, then automatically display remaining no. of records no. in bottom of the delete stmts.

A // Using stmts level trigger

create or replace trigger t1

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.

Q // Using Row Level trigger.

create or replace trigger t1

after delete on emp

~~for each row~~

declare

x number(10);

```
begin
    select count(*) into x from emp; /* here
    dbms_output.put_line(x);
end;
/
```

// Trigger created.

→ ~~The~~ In this Prog no compile time error, but if error in runtime.

Testing:-

SQL> delete from emp where empno=7902;  
ORA-04091 : table SCOTT.EMP is mutating.

↳ Mutating error :-

defn:- If there row level trigger based on a ~~row level~~ table then trigger body ~~comes of~~ 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 any 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 occurred in row level triggers.

↳ This error doesn't occurred of stmts level triggers.

- ↳ whenever we are using stmts level triggers those transactions are automatically committed into DB. That's why using trigger body we can read committed data without any problem.
- whenever that's why stmts level triggers doesn't have mutating errors.
- ↳ whenever we are using row level triggers, DML transaction values are not committed automatically; use trigger body, if we are trying to read without committed data then all DB servers returns mutating errors.
- ↳ To overcome this Problem, we are using autonomous transactions with in triggers, but autonomous transactions always forgets previous result.

Ex:-

```

create or replace trigger fd2
after delete on emp
for each row
declare
x number(10);
Pragma autonomous_transaction;
begin
select current(*) into x from emp;
commit;
dbms_output.put_line(x);
end;
/

```

|| Trigger Created.

## Testing :-

SQL> delete from emp where empno > 902;

Ans

10/07/14

### ↳ when clause :-

We can also specify condition in trigger specification using when clause.

↳ when clause is not allowed to use instead 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 type

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);  
opr display message when empno is 1 only  
// 1 row created.  
SQL> insert into emp(empno) values(2);  
// 1 row created.

↳ 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.

④ SQL create table test (col1 date);  
SQL create or replace trigger tr2  
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

\* Autoincrement 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.nextval, 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 call 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 t-m2

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 system 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/tablename schema
```

```
declare
```

```
---
```

```
begin
```

```
---
```

```
end;
```

### Ex:-

```
SQL> conn sys as sysdba;
```

```
Enter Password: sys
```

```
SQL> create or replace trigger t1
```

```
after create on database
```

```
begin
```

```
dbms_output.put_line('somebody create database object');
```

```
end;
```

```
SQL> set serveroutput on;
```

```
SQL> create table t1(sno number(10));
```

```
somebody create database object.
```

Q) w.a dfl d trigger on <sup>spfct</sup> schema not to drop emp table.

A) create or replace trigger t1

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

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  
instate of triggers.

↳ All triggers information stored under user-triggers  
data dictionary.

SQL> desc user-triggers;

↳ we can also drop triggers.

drop trigger triggername.

## ~: PACKAGES ~

↳ Package is a DB objects. which encapsulates global variables, constants, Procedures, functions, types, cursors into single units.

↳ Generally Packages doesn't accept 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 subprograms 1st time they <sup>automatically</sup> load 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 having two parts

\* Package specification.

\*\* Package body.

↳ By default Package specification objects are Public. whereas Package body objects are Private.

↳ Generally in Package specification we are

declaring objects whereas package body we are implementing Pack those objects.

Syntax:-

Package Specification :-

Create or replace package Packagename.

is/as

global variable declarations, constant declarations;

cursors declarations;

types declarations;

procedure declarations;

function declarations;

end;

Package Body :-

Create or replace package body Packagename

is/as

procedure implementations/definitions;

functions implementations/definitions;

end;

Calling Packaged Procedures :-

Method 1 :-

SQL> exec Packagename. Procedurename(actual Parameters);

Method 2 :- (using anonymous block) :-

SQL begin

Packagename. Procedurename(actual Parameters);

end;

/

Calling Packaged functions:-

Method 1 :- (using select stmts)

SQL> select Packagename.functionname (actual Parameters)

Method 2 :- (using anonymous block)

SQL> begin

    variablename := Packagename.functionname

    (actual Parameters);

end;

Example:-

SQL> create or replace Package Pk\_1

is

    Procedure P1;

    Procedure P2;

end;

/

SQL> create or replace Package body Pk\_1

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 · Pk2;  
// 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 • ~~f(3)~~ from dual;

300

Ques Pragma serially reusable:-

12/07/14

- Y To maintain state of global variable or state of global cursor then we are using serially reusable pragma in Packages.
- Y 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;

/

OP:- 4

State :-

## State of the cursor:-

SQL> create or replace package Pf<sub>1</sub>  
cursor q is select \* from emp;  
Pragma serially\_reusable;  
end;

SQL> begin  
open Pf<sub>1</sub>.q;

end;

SQL> begin  
open Pf<sub>1</sub>.q;

end;

## Overloading Procedures :-

overloading refers to same name can be used for different purpose.

In oracle, we can also implement overloading procedure using package. These procedure having same name with diff types or diff no. of parameters.

Ex:-

SQL> create or replace package Pf<sub>2</sub>

is

Procedure P<sub>1</sub>(a number, b number);

Procedure P<sub>2</sub>(x number, y number);

end;

SQL> create or replace package body Pf<sub>2</sub>

is

Procedure P<sub>1</sub>(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:- 16

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 Pf<sub>3</sub>

is

Procedure P<sub>2</sub>;

Procedure P<sub>1</sub>;

is

begin

P<sub>2</sub>;

end P<sub>1</sub>;

Procedure P<sub>2</sub>

is

begin

dbms\_output.put\_line('Private Procedure');

end P<sub>2</sub>;

end;

/

SQL> exec Pf<sub>3</sub>.P<sub>1</sub>;

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 Preo: Types are created in a procedure/steps:-

1) 1st we are creating type using after/print Syntax then only creating variable of Datatype.

2) PL/SQL having following types:

Composite datatype. {  
• PL/SQL Record.  
• Index by table (or) PL/SQL table (or)  
• Associative array. }  
is also called as collection

- ~~is also called collection~~
- Nested table.
  - Varray.
- composite datatypes.
- \* • Ref cursor.

### 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 structures in 'C' language
- ↳ 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 that type.

Syntax:- (of step 1)

Type typename is record (attribute, datatype (size), -);
---

Syntax:- (of step 2)

Variablename      Typename;
-----------------------------

Ex:-

sql> declare

Type t1 is record (a1 number(10), a2 number(10),  
a3 number(10));

v -> t1;

begin

v.t.a1 := 101;

v.t.a2 := 'Some';

v.t.a3 := 2000;

14/07/14

Ques 7

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

end P1;

|| '' || v\_t.a3);

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 store no. of data items in a single unit.

↳ Basically this is an embedded table.

↳ Basically index by table having key-value

Pairs. i.e here value field stored actual data and key field stored indexes.

↳ These indexes are either nos or characters and also these indexes are either +ve or -ve nos.

↳ This key internally behaves 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. that's why these tables are also called as Memory tables.

↳ For improving performance of the App Oracle provides 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)

This is a user-defined type so we are creating a 2 step process. i.e 1st we are creating type then only we are creating a variable of datatype.

Syntax :- (of step 1)

Type Typename is Table of datatype(size)  
Index by binary\_integer;

Syntax :- (of step 2)

VariableName Typename;

Ex:-

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);

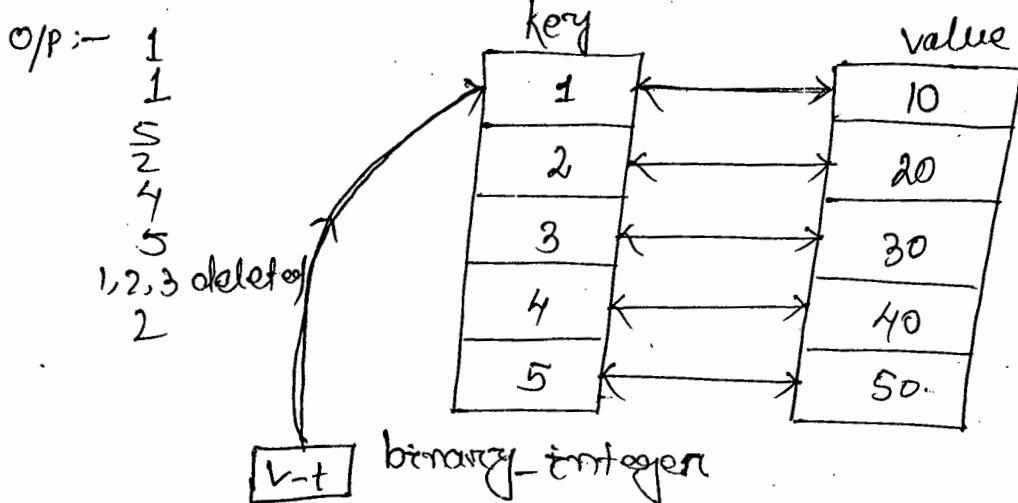
" " (v-t.current);

" " (v-t.delete(1,3));

```

dbms_output.Put_line(v_t.count);
end;
/

```



Q) W.A. PL/SQL Prog to ~~read~~ retrieve all emp names from emp table & store <sup>all</sup> these emp names in index by table & also display content from index by table.

④ 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

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. bcoz cursors internally uses Record by Record process.
- To overcome this problem if we want to improve performance of the application oracle 8i introduce bulk collect clause.

Syntax:-

```

select * bulk collect into collectionvarname
from tablename where condition;

```

we are

- when ever using bulk collect clause in oracle server select columns at a time & stored data into collections, This process automatically improves the 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)

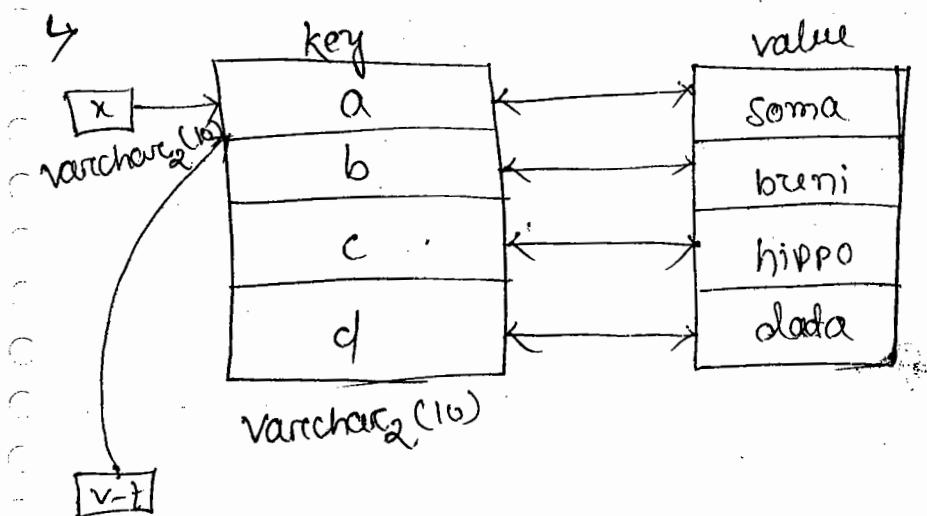
```

declare
type t1 is table of date
index by binary_integer;
v-t t1;
begin
for i 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<sup>n</sup> to transform all employees joining dates from emp table & store those dates in to index by table & display content from index by table.

○ A) 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:-

```

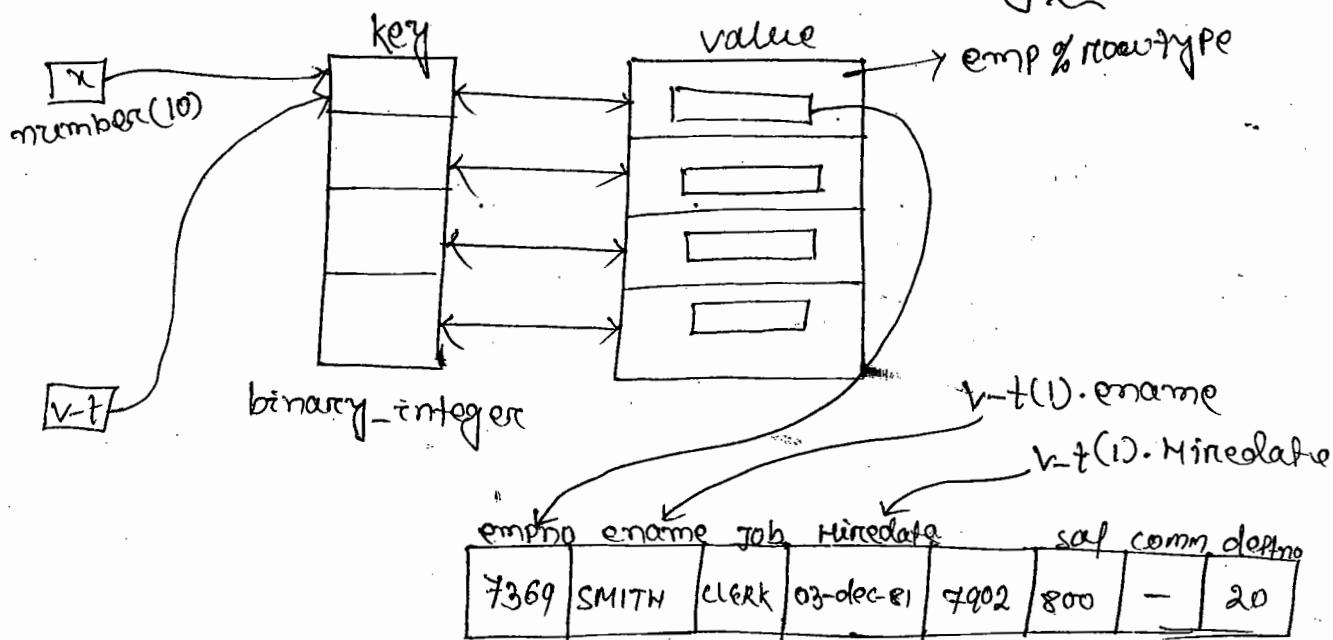
declare
  type t1 is table of varchar2(10)
  index by binary_varchar2(10);
  v-t t1;
  x varchar2(10);
begin
  v-t('a') := 'soma';
  v-t('b') := 'breni';
  v-t('c') := 'hippo';
  v-t('d') := 'dada';
  
```

```

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) or



Ex:-

```

declare
type t1 is table of emp%rowtype
index by binary_integer;
v-t t1;
x number(10);
begin
Select * bulk collect into v-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;
```

```
/
```

ex:-

(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;
```

```
/
```

Ques

? Return Result Sets :-

16/07/14

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 endor 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
vt t1;
begin
select * bulk collect into vt from emp;
return vt;
end f1;
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 whether requested data available in collection or not using indexed.

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('there 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 varchar(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.

**Note:-**

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.

'ORA-01403: no data found'

To overcome this Problem we must use EXISTS collection method.

Soln:-

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;

/

Ques

17/07/14

Nested table, varray:-

Oracle 8.0 introduced nested table, varray.

These are also 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 constraint 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 are consecutive.
- ↳ Generally index by table are not stored permanently into 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 extend, trim 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();

<sup>2</sup>  
    Constructor.

25/07/14

Ques

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 reserve 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;
/
OP:- 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;
/
OP:- 10
```

Note:- we can also specify actual value within constructor itself in nested table, varargs 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;
/
OP:- 10
```

Q) SQL declare

type t<sub>1</sub> is table of Number(10);

v-t t<sub>1</sub>;

v-t<sub>2</sub> t<sub>1</sub> := t<sub>1</sub>( );

begin

if v-t<sub>1</sub> is null then

dbms-output.Put-line('v-t<sub>1</sub> is null');

else

dbms-output.Put-line('v-t<sub>1</sub> is not null');

end if;

if v-t<sub>2</sub> is null then

dbms-output.Put-line('v-t<sub>2</sub> is null');

else

dbms-output.Put-line('v-t<sub>2</sub> is not null');

end if;

end;

/

O/P :- v-t<sub>1</sub> is null,

v-t<sub>2</sub> is not null.

1) declare

type t<sub>1</sub> is table of number(10);

v-t<sub>1</sub> t<sub>1</sub>;

v-t<sub>1</sub> → null

2) declare

type t<sub>1</sub> table of number(10);

v-t<sub>2</sub> t<sub>1</sub> := t<sub>1</sub>( );

v-t<sub>2</sub> → -----

3) declare

type t<sub>1</sub> is table of Number(10);

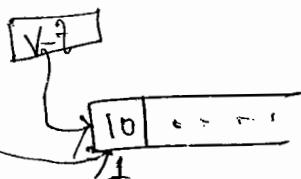
v-t t<sub>1</sub> := t<sub>1</sub>( );

begin

v-t.extend;

v-t(f) := 10

dbms-output.Put-line(v-t(f)); end;



Mac

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 employee names from emp table & store it in to next table & also display content from next table.

(old concept)

```
declare
  type t1 is table of varchar2(10);
  v-t t1 := t1();
  cursor c1 as select ename from emp;
  n number(10) := 1;
  begin
    for i in c1
```

```

    loop
    v-t.extend();
    v-t(n) := t.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;
    /

```

N.B:-

when we are using bulk collect clause for transferring data in to collections, then we are not allowed to use extend collection clause in nested table, becoz internally bulk collect clause only reserves memory for those collections.

{new concept}

so 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 <sup>storing</sup> data in to Varray then we use constraint of this is an user-defined 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();

<sup>constructor</sup>

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) Write PL/SQL Prog to transform 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 bet<sup>n</sup> index by table, nested table & varray:-

- | <u>Index by table</u>   | <u>Nested table</u>   | <u>Varray</u>  |
|---|---|--|
| 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.   | Nested table permanently into DB using SQL.   | we can store varray Permanently into DB using SQL.                                 |
| 3) we can not add, remove indexes.  | indexes using extend, remove indexes trim collection methods.   | we can add, remove indexes using extend, trim collection methods.                  |
| 4) Index values are either numbers or characters and also tve & -ve numbers.  | By default indexes are stored with 1.   | By default indexes are stored with 1.  |
| 5) Index by table having exists, first, last, prior, next, count, delete(index), delete(index, index), delete collection methods. | Nested table having exists, extend, trim, First, last, Prior, Next, count, delete(index), Prior, next, count delete collection methods. | Varray having exists, trim, limit, extend, first, last, delete collection methods. |

- o -

- o -

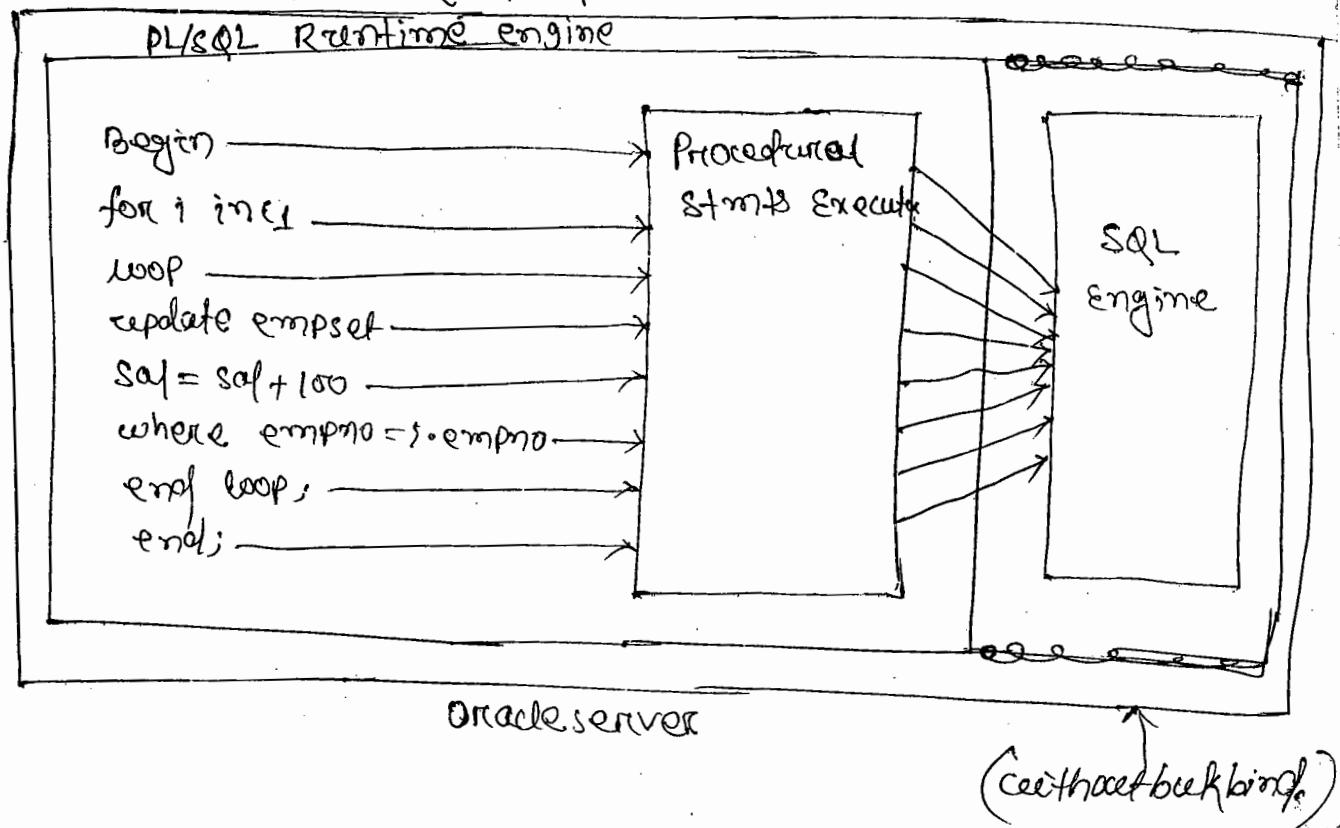
- o -

Ma

## BULK BIND

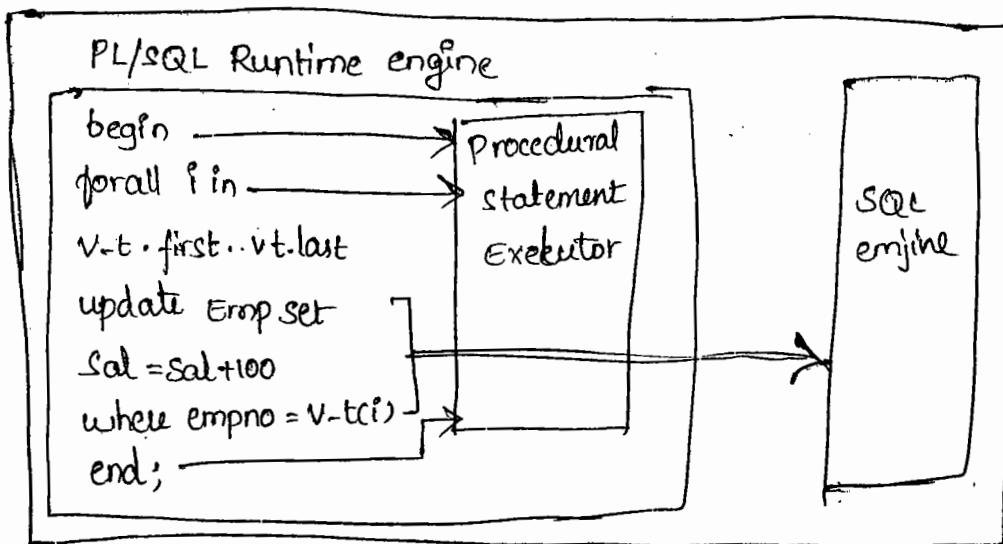
19/07/14

- ↳ When a PL/SQL block contains more number of SQL Procedural statement then all procedural statement are executed in procedural statement executor which within PL/SQL engine and also all SQL stmts are executed within 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 application Oracle introduced Bulk bind Process using collection.
- ↳ In this process we can execute all values in a collection at a time using forall statements.
- ↳ In bulk bind process we are using bulk update, bulk delete, bulk inserts.



using bulk bind (improve performance because of less over-head)

Oracle Server



forall syntax:-

forall indentvarName in Collection Var.first .. CollectionVar.last  
DML statement where

ColName = CollectionVarName (indentvarname);

→ Before we are using bulk bind process (forall statement) we must take data from resource into collection using Bulk collect clause, that's 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 statement's (actual bulk bind).

Step 1:-

bulk collect clause res of 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;  
/
```

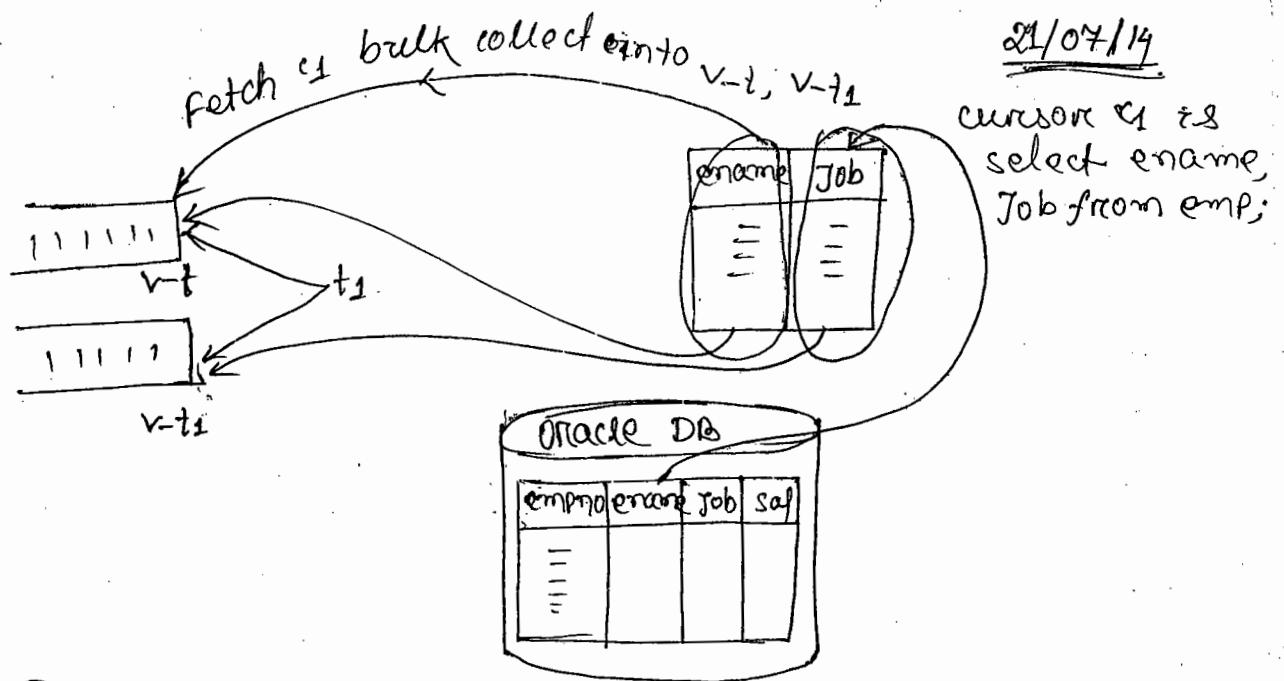
2) bulk collect clause used in cursor ... fetch ... statement

Syntax:-

```
fetch cursorname bulk collect  
into collectionvarname [limit anynumber]
```



Ques



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\_rthility package:

↳ In Oracle if we want to calculate elapsed time for each process then we are using get\_time

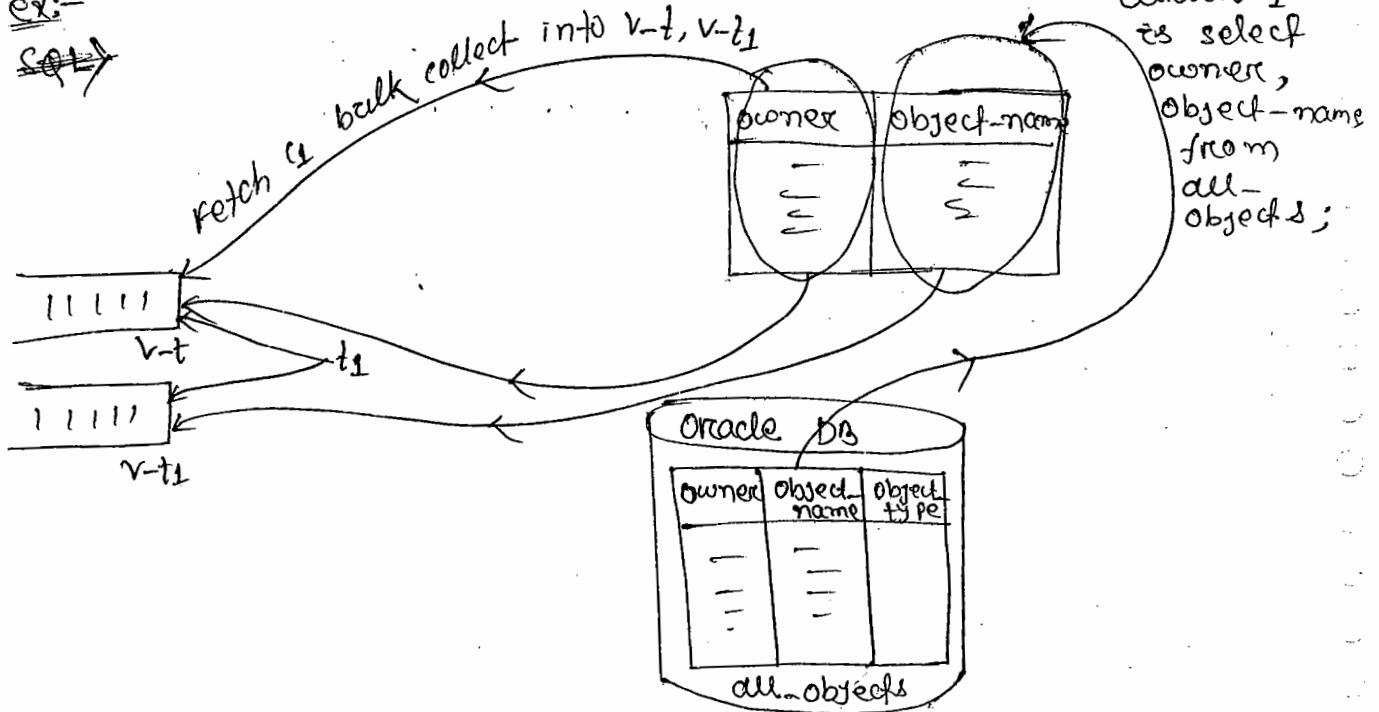
method from dbms\_utility package.  
↳ This methods always returns number datatype.

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\_utility.get\_time;

dbms\_output

for i in 1 ..

```
loop  
null;  
end loop;
```

$Z_2 := dbms_output.Put_line$

$Z_2 := dbms_utility.get_time;$

$dbms_output.Put_line('Elapsed time for normal$

$fetch' || '|| (Z_2 - Z_1) || ' hsecs');$

$Z_1 := dbms_utility.get_time;$

$open c_1;$

$fetch c_1 bulk collect into \del{all objects} v-t, v-t_1;$

$close c_1;$

$Z_2 := dbms_utility.get_time;$

$dbms_output.Put_line('Elapsed time for bulk fetch'$

$|| '|| (Z_2 - Z_1) || ' hsecs');$

end;

O/P:- Elapsed time for normal fetch 104 hsecs.

Elapsed time for bulk fetch 81 hsecs.

Note

22/07/14

Bulk collect clause used to DML returning into clauses  
we can also store DML transaction value into  
variables using returning into clauses.

That's why Returning ... into clauses are used only in  
DML statements.

e.g:-

SQL> variable z varchar(10);

update emp set sal = sal + 100,

where ename = 'King' returning job into :z;

Print z;

z

precedent.

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;
```

Step 2: 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 forall statement.

Syntax:-

```
forall columnname in collectionvarname::first .. collection
variablename::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 program to retrieve 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.

PL/SQL 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;
```

SQL 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] doesn't exist.

Indices of

when ever index by table or nested table having gaps then we are not allowed to use bulk bind process.

To overcome this problem oracle log introduced indices of clause in forall stmts.

Syntax:-

forall indexvarname in indices of collectionvarname  
DML stmts where  
collectionname = collectionvarname (indexvarname);

before oracle log

7369
7902
7566
7829
7788

update all  
rows based  
on  
array testing

oracle log (using indices of clause)

first

7369
7566
7788

v-t(1)=7369
v-t(3)=7566

last

update all  
rows based  
on array  
indexes

SOLN:-

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;

/

Q) Mac  
?

DT 23/07/14

C) 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 (empeno number(10));

SQL> declare

```
type t1 is table of number(10)
index by binary_integer;
vt t1;
begin
select empno bulk collect into vt from emp;
forall i in vt.first .. vt.last
insert into target values(vt(i));
end;
/
```

SQL> select \* from target;

SQL%bulk\_rowcount :-

↳ SQL%bulk\_rowcount attribute 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);

forall i in v-t.first .. v-t.last

loop

dbms\_output.Put\_line ('affected number of rows in

deptno '||' || v-t(i) || '||' || 'is' || '||' sql%break\_

rowcount(i));

end loop;

end;

Output:- 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 having 2 methods :-

1) comma-to-table,

2) table-to-comma.

1) comma-to-table :-

↳ This method is used to convert comma separated  
string into index by table values.

Syntax:-

dbms\_utility.comma-to-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:-

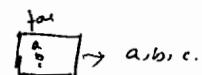
```
dbms_utility.table_to_comma(indexbytablevarname,  
binary_integervarname, stringname);
```

↳ Before we are using these methods it declare section of the PL/SQL block, we are creating a index by table variable using varclarray type from dbm-utility package.

Syntax:-

```
indexbytablevarname dbms_utility.varcl_array;
```

Q-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.



Q SQL declare

```
v_t dbms_utility.varcl_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 retrieve 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.cursor_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.

~~Temporary cursor~~

\* ref cursor (or) cursor variable (or) dynamic cursor :-

↳ Oracle 7.2 introduce ref cursor.

↳ 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. whereas as ref cursor DB servers executes no. of select stmts dynamically for a single active set area.

24/07/14

Q What is ref cursor?

A That's why this cursor is also called as dynamic cursor.

Q 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 cursor is a user-defined type.

Q Generally static cursor doesn't return multiple values in to the client applications, whereas ref cursor returns multiple values in to client applications.

Q 2 types of ref cursors in all DB.

1) strong ref cursor.

2) weak ref cursor.

Q Strong refcursor is ref cursor which having return type, whereas weak ref cursor is a ref cursor which doesn't have return type.

Q 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 cursor is also called as cursor variables.

### 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 executable 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;

t emp%rowtype;

begin

open v-t for select \* from emp where sal>2000;

loop

fetch v-t into t;

exit when v-t%notfound;

dbms\_output.put\_line(t.ename || ',' || t.sal);

end loop;

close v-t;

/

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%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.sof||' '||i.deptno);
end loop;
elsif 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 emp details from emp tables.

④ SQL> declare

Q) SQL declare create or replace Procedure

PL(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 ?

PL(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;
t emp%rowtype;
begin
  P1(v-t);
  loop
    fetch v-t into t;
    exit when v-t%notfound;
    dbms_output.put_line(t.ename||'/'||t.sal);
  end loop;
  close v-t;
end;
/
```

Mode  
Exe

25/07/14

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)
  begin
    open v-t1 for select * from emp;
  end P1;
```

Procedure P<sub>2</sub>(V-t<sub>2</sub> out t<sub>2</sub>)

is

begin

Open V-t<sub>2</sub> for select \* from dept;

end P<sub>2</sub>;

end;

/

Execution:~ (using Bind variable)

SQL> variable a refcursor;

SQL> variable b refcursor;

SQL> exec PK<sub>1</sub>.P<sub>1</sub>(:a);

SQL> exec PK<sub>1</sub>.P<sub>2</sub>(:b);

SQL> Print a b;

\* Note:~

→ In all DB systems we are not allowed to used  
refcursors directly in Packages. Becoz Bar ~~cannot~~ only  
refcursor is a user-defined type.

Utt 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 create 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/tiger;  
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 to the file we must declare a filepointer variable in declaration section of the PL/SQL block, using file\_type from utl-file package.

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
```

Step 3: For writing data into file then we must use Putf() from utl-file package.

Syntax:-

```
utl_file.Putf(filepointervar, content);
```

Step 4: After data into file we must close the file using fclose() from utl-file package.

Syntax:-

```
utl_file	fclose(filepointervar);
```

Example:-

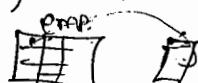
SQL declare

```
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) Write A PL/SQL Program write all employee names from emp table in to external file using utl-file Package.

Ans) declare

```
fp utl_file.file_type;
begin
fp := utl_file=fopen ('EMP', 'emp.txt', 'w');
```



④ 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

26/07/14

### Note

Whenever we are using utl-file package to write column data into file then DB server automatically write data horizontally 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.

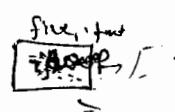
Syntax:-

`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;
      /
```

↳ A PL/SQL Prog read multiple values from file 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','file1.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:-

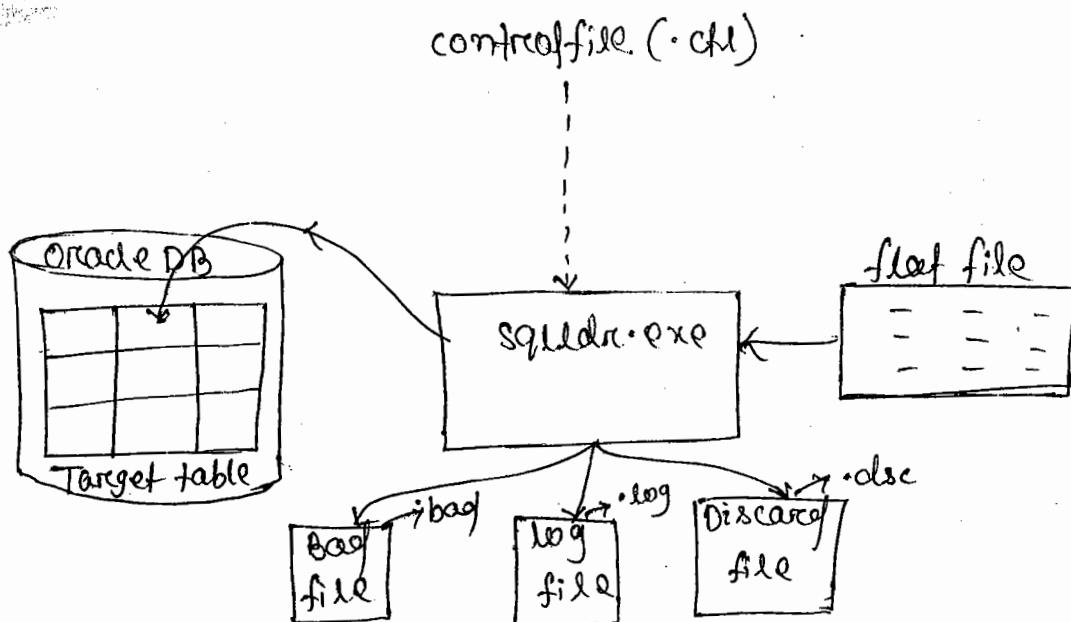
- ↳ In Oracle when we are reading multiple values from a file using ref-file Package, then oracle server returns an error ORA-01403 no-data-found the when control reach to the end of file.
- ↳ To overcome this Problem we must use no-data-found exception name.

### SQL loader :-

SQL loader is a utility program which is used to transform data from flat file into Oracle DB.

SQL loader is also called as bulkloader.

Always SQL loader executes control files. This file extension is .ctl.



↳ Based on 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 log file, and also this file stored or other file interaction & also stored Oracle error msg & also stored loaded, rejected no. of record nos.

↳ 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 user id = scott/tiger & control = Path of the control file

### flat file :-

↳ flat file is an structured file which contains no. of records.

↳ Oracle having two types of flat files i.e

- 1) Variable record flatfile.
- 2) fixed record flatfile.

### variable record flatfile :-

A flatfile having delimiter & each field having diff length then those type of flatfile are called variable record flatfile.

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:- 10LabC2000

202CC7000.

