

ORACLE 12C

sql&pl/sql

by

murali

naresh

MANOJ ENTERPRISES & XEROX

All soft ware institute materials, spiral-binding,

Printouts & stationery also available ..,

Contact:8125378496

Add: Plot No.40, Gayatri Nagar, Behind HUDA, mithrivannam, HYD.

- Oracle is a relational database which is used to store data or information permanently into secondary storage devices.
- If you want to operate oracle database then we are using Non-procedural language SQL and also we can use procedural language PL/SQL
- All organization stores data or information for future purpose.

Data :-

It is a collection of raw facts.

ex:- Student Marks

Customer names,

SRI SAI BHAVANI XEROX
Plot No. 40, Ground Floor,
Rama Reddy Layout, Gayathri Theatre,
Ameerpet, Hyderabad-500 016.
Ph: 9908195958

Information:-

Information is nothing but meaningful data or process data. When we are processing data then we are achieving meaningful result; this is called information.

ex:- Student Marksheets
Invoice of customer

Data Stores:-

It is a place where we can store data or information.

1. Papers & books
2. Flat files
3. Database

1. Flat files -

It is a traditional mechanism which is used to store data into permanently into secondary devices.

Drawbacks:-

- Data Retrieval
- Data Redundancy
- Data Integrity
- Data Security
- Data Indexing

SRI SAI BHAVANI XEROX
Plot No. 40, Ground Floor,
Rama Reddy Layout, Gayathri Theatre,
Ameerpet, Hyderabad-500 016.

Ph: 9908195958

• Data Retrieval —

If we want to retrieve data from flat file then we must develop application programs in high level language, whereas if we want to retrieve data from Database then we are using SQL.

SQL \Rightarrow Structured Query Language.

- Data Redundancy :-

- In flat files sometimes we are maintaining multiple copy of the some data in different location.
- This data is also called as duplicate data or redundant data.
- In flat file whenever we are modifying this data in one location it is not affected in other locations. This is called inconsistency. That's why flat files doesn't maintain consistency in data automatically.
- Databases automatically maintain consistent data through transactions.

Every transaction internally having four properties; these properties are also called as ACID property.

These properties only automatically maintain consistent data in database.

If you want to reduce duplicate data in database then we are using "Normalization process".

- Data Integrity :-

Integrity means to maintain proper data if we want to maintain proper data in database then we are using constraints (Primary key, foreign key, ...) whereas in flat files we must develop application program in high level language for maintaining valid data.

- Data Security -

Data stores in flat files can't be secured because flat files doesn't provide security mechanism.

Whereas Database provide role based security.

- Data Indexing -

If we want to retrieve data very fast from Database then we are using indexing mechanism on Database. whereas flat files doesn't support indexing mechanism.

Organization suffering from flat files mechanism for storing data or information in secondary storage devices.

That's why organization introduced a new software called DBMS. which is used to store data or information efficiently in secondary storage devices.

Data Base Management System

3

(DBMS)

It is a collection of programs (software) written to manage software.

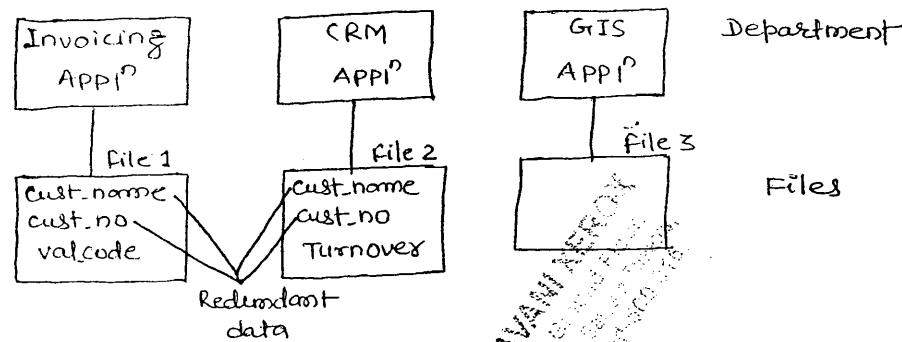
example of DBMS S/W -

- oracle
- teradata
- SQL server
- MySQL
- SQL-lite
- informix
- Sybase ... etc.

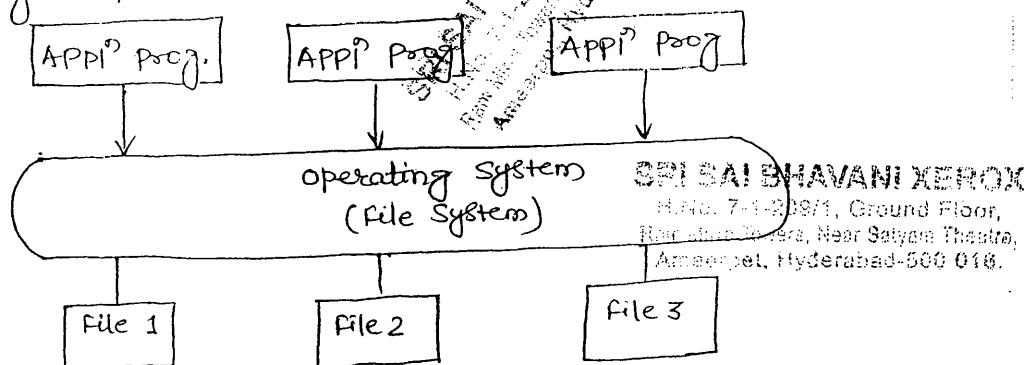
GRI SAIBHAVANI XEROX
H.No. 7-1-209/1, Ground Floor,
Ram Bhawan Towers, Near Satyam Theatre,
Amarpet, Hyderabad-500 016.

- In flat files mechanism every application program maintains its own file separate from other application programs in the organization.

ex:- example of flatfile for storing data

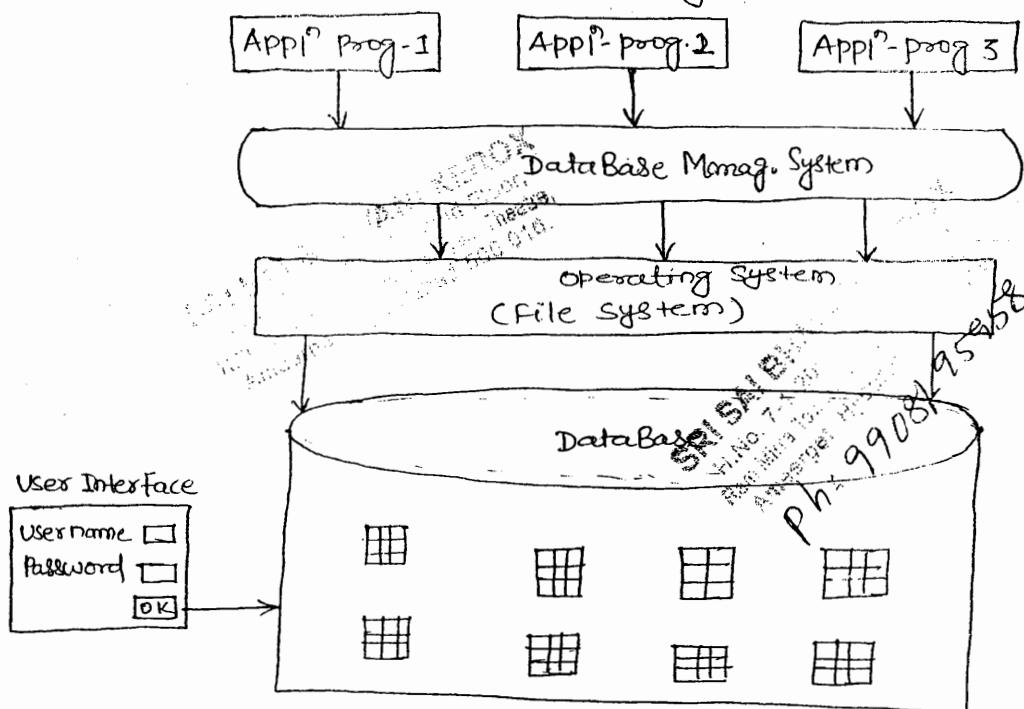


Data Management in flat files -



Whenever we install DBMS S/W into our system or automatically some places is created in Hard-disk; is called DataBase and also automatically an user interface is created through this user interface we can directly interacting with an database or through the application programs we are indirectly with DataBase.

ex:- DBMS Approach for Data management.



DataBase -

It is an organized collection of interrelated data.
(OR)

It is a collection of structured data.

- Every Database having two types of structures -

1. logical structure.
2. physical structure.

1. logical structure:-

A structure which is not visible in O.S. is called logical str.

Logical structure containing table, views, sequence, synonyms, etc

Logical str. is handle by database developer or either Database Administrator.

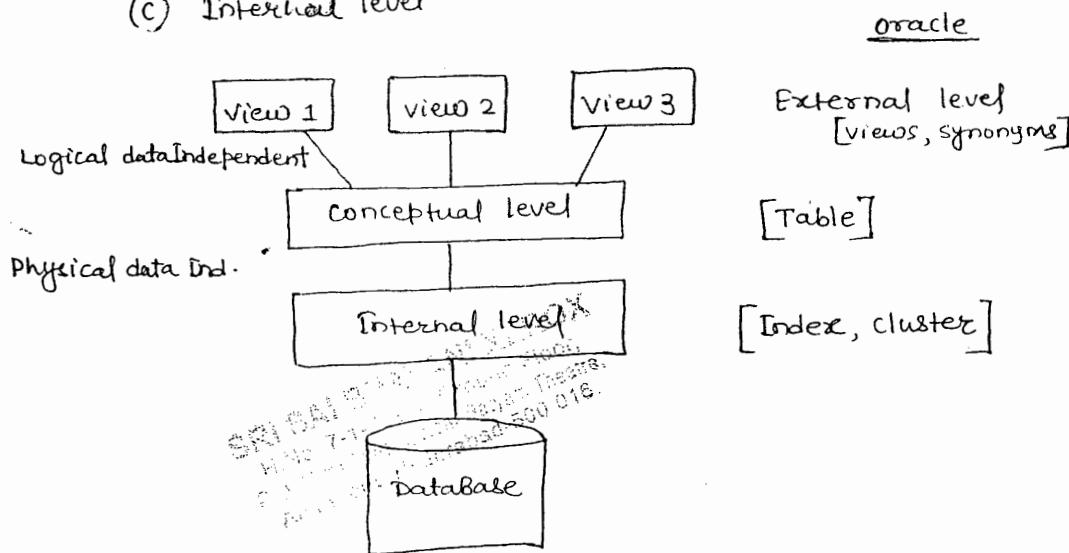
2. physical structure -

A structure which is visible in O.S. is called physical structure.

Physical str. handled by DBA only.

DBMS - Architecture:-

- ANSI (American National Standard Institute) has established three level architecture for DBMS.
- This Architecture is also called as ANSI/SPARC (std. planning & req.) committee architecture.
- Main objective of DBMS architecture is to separate user view of the database from the physically it is stored.
- DBMS architecture mainly consists of three levels -
 - Conceptual level
 - External level
 - Internal level



Three level architecture provides data independence.

Data Independence:-

Data Independence is nothing but upper level are unaffected by changes in the lower level.

DBMS architecture having two types of Data Independence -

1. logical Data Independence
2. physical Data Independence

1. Logical Data Independence

changes to the conceptual level do not require changes to the external level is called logical data independence.
ex - adding a new entity in conceptual level shouldn't affect in external level.

Q. Physical data Independence

Changes in the internal level, then don't require changes to the conceptual level is called Physical data independence.

ex:- Adding a new entity in internal level shouldn't affect in conceptual level.

1. Conceptual level :-

Conceptual level describes logical structure of database.

This level doesn't define how data physically stored in DB.

Conceptual level defines what type of data can be stored in database by specifying datatype and datatype signs.

- It defines what type of data can't be stored by specifying constraints (primary key, foreign key, ...)

This level also defines relationship between tables.

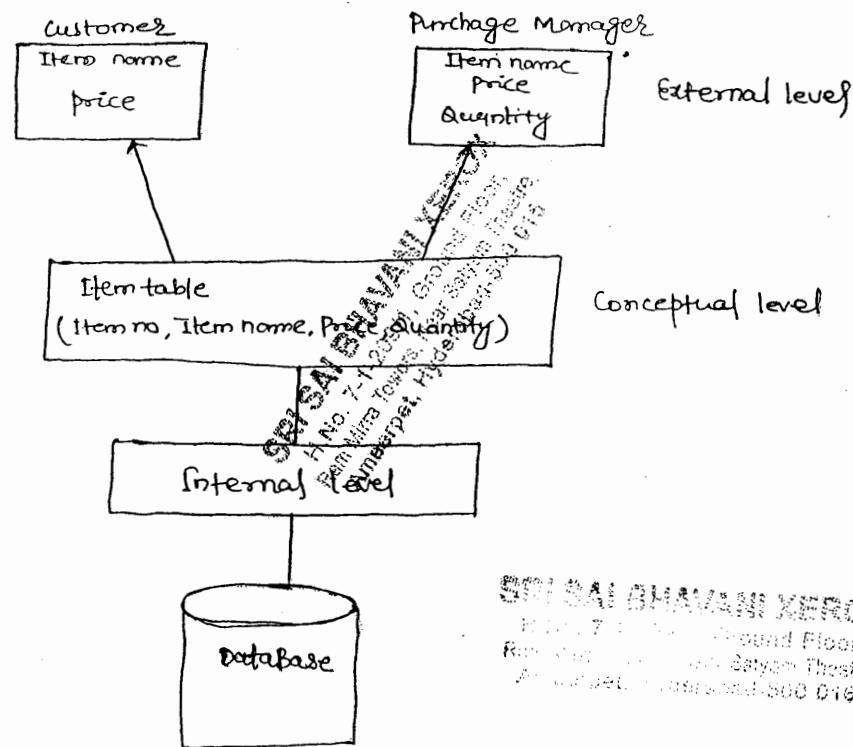
ex:- SQL > create table bank (accno (10) primary key, name varchar(10), balance number(10));

2. External level :-

External level describes user's view of database.

External level provides a security mechanism through views i.e. Some type of users only access portion of the data from the Conceptual level.

ex:-

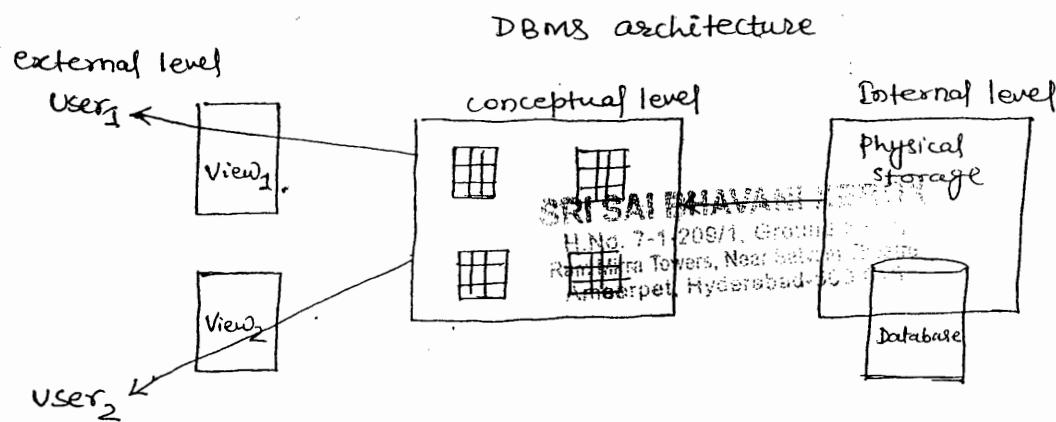


Generally, data security points of view database Administrator (DBA) creating views from conceptual level and then only those views given to the no. of the users,
Then only no. of users allowed to access portion of the data in conceptual level.

3. Internal level:-

Indexes and clusters are available to control and manage the way the data physically stored in the hard-disk.

Whenever we're adding indexes on the database then table structure does not effect in conceptual level this is called Physical data independence, But performance will be affected.



Data Model.

How data is represented at the conceptual level defined by means of data model.

In the History of Database design three data Model have been used. They are -

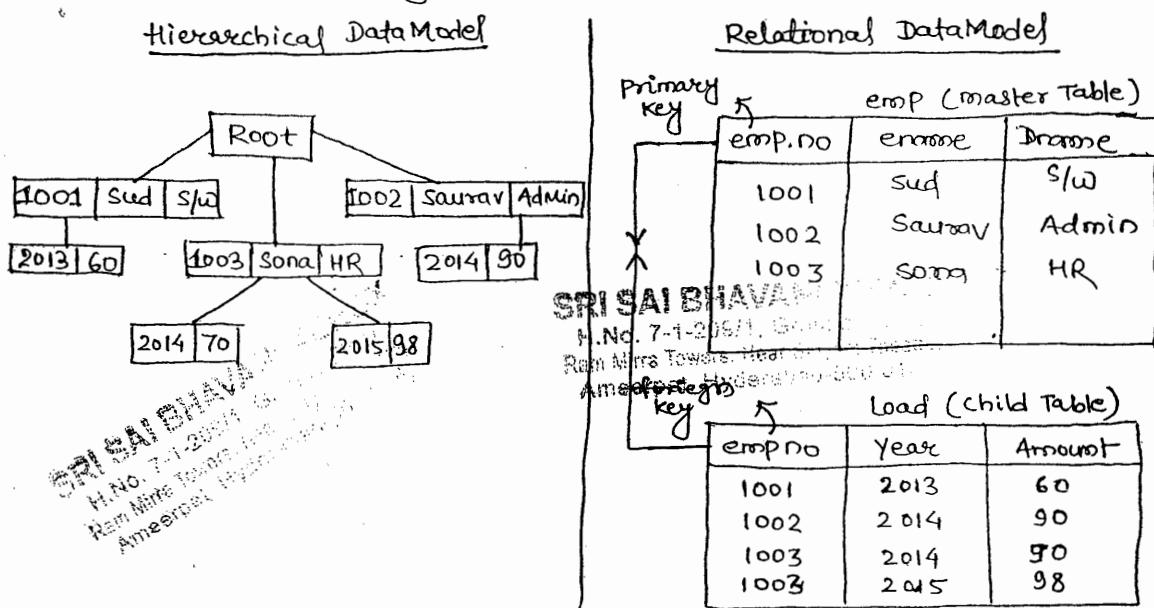
1. Hierarchical Data Model
2. Network Data Model
3. Relational Data Model.

I. Hierarchical Data Model :- (1960's onwards)

In Hierarchical Data Model organise data in tree like str in Hierarchical Data Model record type is also same as table in relational data model.

In this Data Model also data is represented in format of records.

- Hierarchical Data Model is implemented based on one to many relationship between Parent, child records based on this relation many child records having single parent records.
- That's why in this Data Model always child records are repeated that's why in this data Model having more duplicate data.
- In 1960's IBM introduced IMS (Product Name) Product Hierarchical Data Model if we want to operate Hierarchical Data Model Product then we are using procedural language.



2. Network Data Model

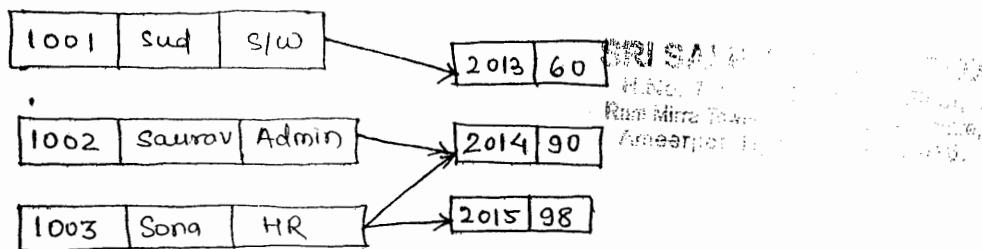
In 1970's CODASYL (Conference on Data System Language) committee introduced Network Data Model.

In Data Model having less duplicate data because in this data Model parent-child relationship implemented based on many to many relationship. In this data Model also data is represented in the formate of records, and also record Type is same as table in Relational Data Model.

- In 1970 IBM introduced IDMS (Information Data Management System) product based on N/W data Model.
If we want to operate N/W data Model Product also then we are using procedural language.

SRI SAI BHAVAN
M. No. 7-1-206/1, Gachibowli,
Ran Mitra Towers, Near Nizam Institute of
Management, Hyderabad - 500 032
Ameerpet, Hyderabad, Telangana, India

ex:-



3. Relational Data Model

In 1970 E.F.Codd introduced Relational Data Model.

This data Model consist of collection of relations, these relation is represented in the form of Table.

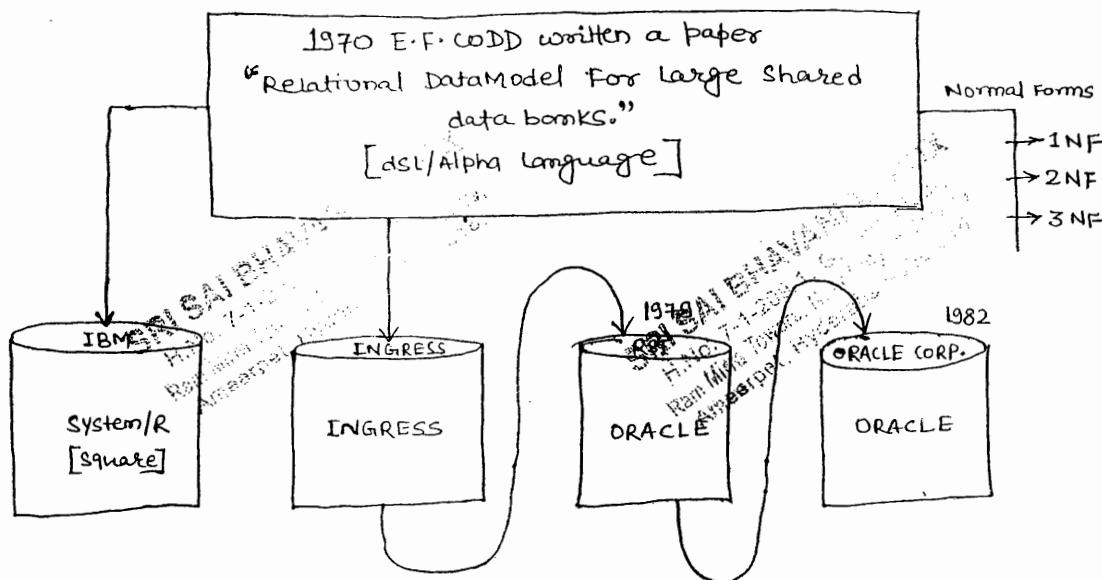
That's why in relational data Model we can store data in 2-D table.
Relational Data Model mainly consist of 3-component these are-

(a) collection of objects.

ex:- tables, views, indexes, synonyms, clusters etc.

(b) set of operators

(c) set of integrity rules



CEO ORACLE CORPORATION - Larry Ellison.

Timeline of ORACLE

oracle → (a) SQL (b) PL/SQL (c) Dynamic SQL

1977:

Larry Ellison, Bob Miner, Ed Oates started a new company SDL.
SDL (Software Development Laboratories)

1978:

(first version Oracle 1.0)

This version is never released.

1979:

SDL company name changed into RSI (Relational SW Inc)

1982:

RSI name changed into oracle corporation.

ORACLE VERSIONS

- ORACLE (2.0) - 1979
 - First public version
 - Basic SQL functionality, Joins.
- ORACLE (3.0) - 1983
 - Commit, rollback
 - Rewritten in C-language.
- ORACLE (4.0) - 1984
 - Read consistency
 - Import/Export utility programs
- ORACLE (5.0) - 1985
 - introduced client-server architecture
- ORACLE (6.0) - 1988
 - introduced PL/SQL
 - Two level locks
- ORACLE (7.0) - 1992
 - Datatype varchar changed into varchar2
 - Truncate table
 - Stored Procedures, stored function.
 - triggers.
 - Integrity constraints
 - roles
 - view compilation.

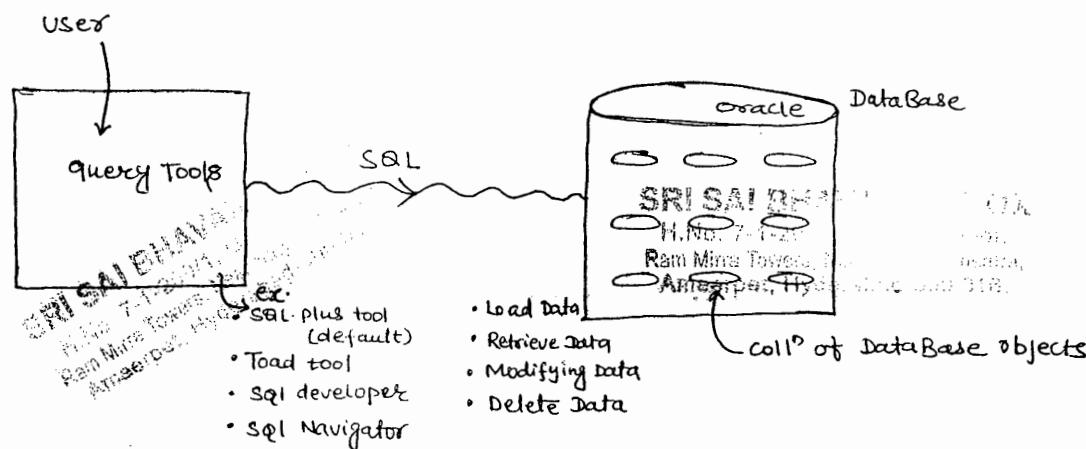
MANOJ ENTERPRISES
 Plot No: 40, Gayathri Nagar,
 Ameerpet, Hyderabad,
 Cell: 8125378496

- ORACLE (7.1) - 1994
 - Dynamic SQL
 - ansi/iso SQL92
- ORACLE (7.2) - 1995
 - inline view
 - ref. cursors (or) cursor variable
 - dbms-job package
- ORACLE (7.3) - 1996
 - Bitmap indexes
 - utl-file package
- ORACLE (8.0) - 1997
 - introduced object Technology
 - columns increased per table up to 1000.
 - Instead of triggers
 - Large objects (clob, blob, bfile, datatypes)
 - nested table, varrays
- ORACLE (8i) - i-internet - 1999
 - materialized view
 - rollup, cube
 - function based indexes
 - Bulk bind
 - analytical function
 - trim(c) function
- ORACLE (9i) - 2001
 - merge statement
 - flashback query
 - 9i Joins (or) ANSI Joins
 - multitable insert
 - renaming a column.
- ORACLE (10g) - 2003
 - f-grid Tech.
 - introduced recycle bin
 - flashback table
 - indices of clause
 - wmi-concat()
 - regular Expression.

MANOJ ENTERPRISES
 Plot No: 40, Gayathri Nagar,
 Ameerpet, Hyderabad.
 Cell: 8125378496

- ORACLE (11g) - 2007
 - Continue statement introduced in pl/sql loops.
 - Read only tables.
 - simple_integer datatypes in PL/SQL
 - Virtual column
 - Pivot () - function
 - follows clause in triggers
 - compound triggers
 - enable disable clauses are used in trigger specification.
 - named, mixed notation are used in sub-prog executed by using select statement.
 - sequences are used in pl/sql blocks without using dual table.
- ORACLE (12c) - 2013 c-cloud computing
 - invisible columns
 - identity columns
 - session specific sequences.
 - New top_N analysis (fetch first, fetch next clauses)
 - truncate table cascade
 - with clause used in pl/sql subprogram.
 - Accessible by clause used in procedures.

SRI SAI BHAVAN COLLEGE
H.No. 7-1-209/1, Gachibowli,
Rani Mirra Towers, Near Satyam IT Park,
Ameerpet, Hyderabad-500042.

Architecture -Oracle 10g, 11g, 12C Enterprise Edition

Username

Password

Error: Account locked

UserName

Enter Password: sys

SQL> Alter user scott account unlock;

SQL> conn scott/tiger

Enter Password: tiger

Confirm Password: tiger

Start --> programs --> oracle

11g-home --> Application

Development --> SQLplus

User Name: Scott

Password: tiger

SRI SAI BHAVAN
H.No. 7-1-200
Raja Mira Towers
Ameerpet, Hyderabad, India
Ph: 9908195958

clear screen -

SQL > clr scr;

(OR)

Shift + del (through keyboard)

To view all Tables -

SQL > select * from tab;

DEPT -----> master table

EMP -----> child table

To view Particular Table :-

Select * from TableName;

ex:- select * from Dept;

Select * from Emp;

CDL SAI BHAVAN
M. No. 7-1-205/1, C.G.
ECC Main Road, Secunderabad
Hyderabad - 500003

(Structured Query Language)

- SQL is a Non-procedural language which is used to operate all relational databases.
- In 1970's E.F.Codd introduced dsl/alpha language, which is used to operate relational Database.
- In IBM system/r team introduced Simplified version of dsl/Alpha language "square" again IBM changed "square" to "SEQUEL" (Structured English Query Language) and again IBM changed into "SEQUEL" to "SQL"

ansi - 1986

ISO - 1987

ansi/iso sql 86 --- → sql 89 (Compatibility Purpose)

ansi/iso sql 92 --- → sql 92

ansi/iso sql 99 --- → sql 99

ansi/iso sql 2003 --- → sql 03

Data definition Language (DDL):—

- Create
- Alter
- Drop
- Truncate
- Rename (Oracle 9i)

SRI SAI BHAVAN
H.No. 7-1-269/1, 2nd Floor, 100ft Road,
Ram Niwas Tower, Ameerpet, Hyderabad, 500012
Ameerpet, Hyderabad, 500012

ph : 9908195958

Data Manipulation Language (DML):—

- Insert
- Update
- Delete
- Merge (Oracle 9i)

Data Query Language (or) Data Retrieval Language :-

- SELECT

Transactional Control Language (TCL) :-

- Commit (Save)
- Rollback (like undo)
- Savepoint

Data Control Language (DCL) :-

- Grant (Giving Permission)
- Revoke (Cancel Permission)

SRI SAI BHAVAN
H.No. 7-1-269/1, 2nd Floor, 100ft Road,
Ram Niwas Tower, Ameerpet, Hyderabad, 500012
Ameerpet, Hyderabad, 500012

DATA-TYPES :-

Data-types identifies type of data within a table column.

Oracle having following datatypes ; these are -

1) Number (P,S)

2) Char

↳ varchar2 (maxsize)

3) Date

1) Number (P,S) -

P \Rightarrow precision

 ↑ Total no. of digits

S \Rightarrow scale

- It is used to store fixed, floating Point Number.

Syntax -

Column Name number (P,S)

Ex:-

SQL> create table test (sno number(7,2));

SQL> insert into test values (12345.67);

SQL> select * from test;

SNO
12345.67

- SQL> insert into test values (123456.7);

ERROR: value larger than specified precision allowed for this column.

Note:- whenever we are using number (P,S) format then we are not allowed to insert more than (P-S) no's of digits before decimal point.

example:- number (P,S) \Rightarrow (P-S) \neq Number (7,2)

$$7-2 \Rightarrow 5$$

- SQL> insert into test values (12345.6789);

SQL> select * from test;

SNO

12345.68

Whenever we are using Number(P,S) and also when we are try to insert more no's of digits after decimal point then Oracle server doesn't return any Error.

In this case Oracle server internally automatically rounds off number based on the maximum size specified in scale within declaration.

number (P) -

- It is used to store fixed Numbers.

Syntax:-

ColumnName number (P)

e.g- SQL>create table test1 (Sno number(6,2));

SQL> insert into test1 values (99.99);

SQL> select * from test1;

SNO

100

SQL> insert into test1 values (99.4);

SQL> select * from test1;

SNO

99

Note :-

In Oracle number datatype maximum precision is upto 38 digits.

2. Char :-

It is used to store fixed length alphanumeric data in bytes.

Maximum limit is upto 2000 bytes.

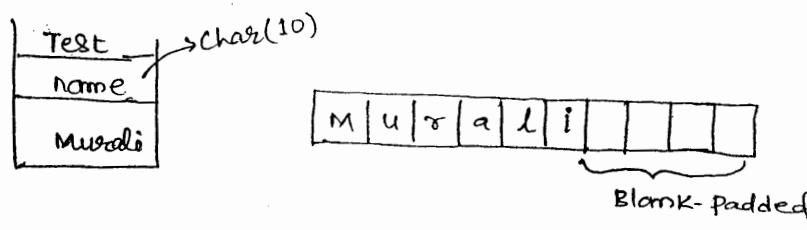
Syntax:-

ColumnName char (maxsize)

Note:- By default character datatype having 1-Byte size.

- When we are try to store less no. of bytes then the data-type size specified in character Datatype; then Oracle server automatically adding blank spaces in place of remaining bytes after end of the string; this called blank Padded Mechanism.

To overcome this problem Oracle provided varchar2 datatype.



- VARCHAR2 - (MaxSize)

Oracle 7.0 introduced Varchar2 datatype it is used to store variable length of alphanumeric data in Bytes.
Maximum size is 4000 bytes.

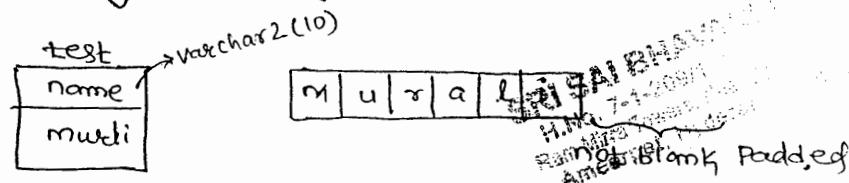
Syntax -

Columnname	VARCHAR2 (MaxSize)
------------	--------------------

Note:-

when we are try to store less no's of bytes then data-type size specified in varchar2 datatype . Then oracle server doesn't add blank spaces at the end of the String.

That's why this datatype doesn't waste disk space.



- VARCHAR (maxsize)

Prior to oracle 7.0 oracle having varchar Datatype.
It is also same as varchar2 Datatype . Varchar also stores variable length alphanumeric data in Bytes but Maximum size of the varchar datatype is upto 2000 bytes.

Syntax

Columnname	VARCHAR (maxsize)
------------	-------------------

3.) Date :-

It is used to store dates in oracle date format.

In oracle by default Date format is DD-MON-YY

Syntax

Columnname	'date'
------------	--------

ex - insert into emp values ('10-JAN-25');

1) Data Definition Language (DDL) :-

- create
- alter
- drop
- truncate
- rename (oracle 9i)

DDL Commands are used to define structure of the table.

a) create :-

It is used to create database objects like tables, views, synonyms, indexes... .

Creating a table -

Syntax -

```
create table tablename (columnname1 datatype(size),
                      columnname2 datatype(size), ...);
```

ex:-

```
SQL> create table first (sno number(10), name varchar2(10));
```

To view structure of the table :-

Syntax -

```
desc tablename;
```

ex -

```
SQL> desc first;
```

SQL> desc first;
 Name Type
 SNO NUMBER(10)
 NAME VARCHAR(10)

b) alter :-

It is used to change existing table structure.



- add -

It is used to add new columns into existing table.

Syntax -

```
alter table tablename add (col1 datatype(size), col2 datatype(size),
                           ...);
```

example -

alter table first add sal number(10);

SQL> desc first;

- Modify :-

It is used to change column datatype or datatype size only.

Syntax -

alter table tablename modify (col1 datatype(size), ...);

example -

① alter table first modify sno date;

② alter table first modify sno varchar2(10);

SQL> desc first;

5) - drop :-

It is used to drop columns from the table.

method 1 -

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

Syntax -

alter table tablename drop column columnname;

ex -

alter table first drop column sno;

SQL> desc first;

method 2 -

If we want to drop single or multiple columns at a time with using parentheses then we are using following syntax

Syntax -

alter table tablename drop (col1, col2, ...);

ex -

alter table first drop (name);

SQL> desc first;

Note:-

In all databases we can't drop all columns in the table.

Ex:- alter table first drop column sal;

Error:- cannot drop all column in a table.

drop:-

It is used to remove database objects from Database.

Syntax-

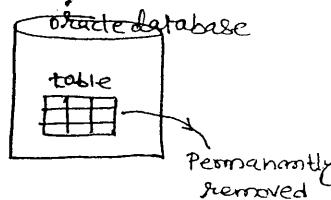
```
drop object objectname;
(or) drop table tablename;
(or) drop view viewname;
```

dropping a table

before oracle 10g

Syntax-

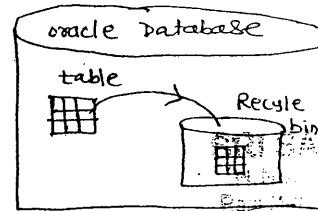
```
drop table tablename;
```



oracle 10g, 11g, 12c (Enterprise Edition)

Syntax-

```
drop table tablename;
```



SHAVANI XEROX
Sant Ghandi Nagar,
Rajkot - 360 002
Phone: 079 246 600 0194

get it back from recyclebin

Syntax-

```
flashback table tablename to before drop;
```

To drop permanently -

Syntax-

```
drop table tablename purge;
```

ex:-

```
SQL> desc first;
```

```
SQL> drop table first;
```

testing -

```
SQL> desc first;
```

Error:- Object first does not exist.

get it back from recyclebin -

```
SQL> flashback table First to before drop;
```

testing -

```
SQL> desc first;
```

To drop permanently

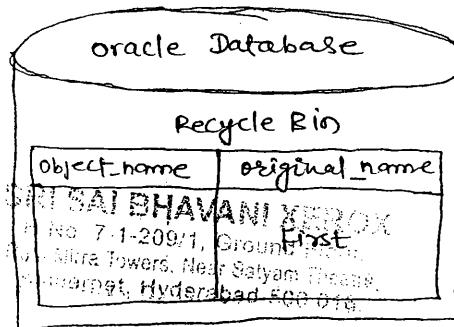
SQL> drop table first purge;

Testing -

SQL> flashback table first to before drop;
error: object not in RECYCLE BIN.

RECYCLE BIN

- Oracle 10g introduced RECYCLE BIN which is used to store dropped tables.
- RECYCLE BIN is a predefined Read-only table.
Whenever we're installing oracle server then automatically so many predefined read only tables are created; these tables are also called as Data Dictionary.



Ex:- SQL> create table first (sno number(10));
SQL> drop table first;

Testing -

SQL> desc recyclebin;

SQL> select ORIGINAL_NAME from recyclebin;

Showing :-

ORIGINAL_NAME
FIRST

NOTE:-

In oracle we can also drop tables from recycle bin by using purge command.

to drop single table from recyclebin -

Syntax -

Purge table tablename;

To drop all tables from recyclebin -

Syntax -

Purge recyclebin;

d) truncate :-

Oracle 10.0 introduced truncate table command it is used to delete all rows permanently from the table.

Syntax -

truncate table tablename;

example:-

SQL> create table first as select * from emp;

SQL> select * from first;

SQL> truncate table first;

testing -

SQL> select * from first;

no row selected.

SQL> desc first;

e) rename :-

- It is used to renaming a table.

Syntax -

rename oldtablename to newtablename;

ex:-

SQL> rename first to second;

Renaming a column - (oracle 9i)

Syntax -

alter table tablename ^{column} rename^ Oldcolumnname to newcolumnname;

ex:-

SQL> alter table emp ^{column} rename empno to sno;

SQL> select * from emp;

NOTE - In all databases by default all DDL commands are automatically committed (Save).

SRI SAI SHARMA
H.No. 7/1 APT. 101
Ram Niwas Layout,
Anandpuri, Hyderabad
ph: 9908195958

Data Manipulation Language:- (DML)

DML commands are used to manipulate data within a table.

These are—

- 1.) Insert
- 2.) Update
- 3.) Delete
- 4.) Merge

1.) Insert :-

It is used to insert data into in the table.

Method 1 —

Syntax —

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

example—

SQL> select * from first;

SQL> No rows is selected

SQL> insert into first values (1,'murali');

SQL> insert into first values (2,'gokul');

SQL> select * from first;

SNO	Name
1	Murali
2	gokul

Method-2 (Using substitutional operator) —

& ≠ Enter values for

Syntax —

```
insert into tablename values (&columnname1, &columnname2, ...);
```

Ex:- SQL> insert into first values (&Sno, '&name');

Enter the value for sno: 3

Enter the value for name: abc

SQL> /

[This gives the previous command]

Enter the value for sno: 4

Enter the value for name: sudh.

SQL> select * from first;

25

SNO	Name
3	abc
4	suf

Ex:- SQL> insert into first values (&1, '&2');

Enter values for 1: 5

Enter values for 2: pqx

Method 3 (skipping columns) :-

Syntax -

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

Ex:- SQL> insert into first (name) values ('zzz');

1 row created

SQL> select * from first;

SNO	Name
3	abc
4	suf
	zzz

2. update :-

- It is used to change data in a table.

Syntax -

update tablename set Columnname = NewValue where
Columnname = OldValue;

Ex:-

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

1 row updated

SQL> select * from emp;

Note - In all databases we can also use update statement for inserting data into particular shell.

SQL> alter table first add address varchar2(10);

SQL> select * from first;

sno	name	address
3	abc	SAI BHAVANI NERWA
4	sud	

SQL> update first set address = 'mumbai' where name = 'sud';

sno	name	address
3	abc	
4	sud	mumbai

To delete particular row of data like mumbai.

SQL> update first set address = null where name = 'sud';

1 row updated

SQL> select * from first;

sno	name	address
3	abc	
4	sud	

3> delete:-

- It is used to delete rows or particular rows from a table.

Syntax - all rows deleted from table

delete from tablename;

(or) delete from tablename where condition;

particular rows deleted.

ex:-

SQL> delete from first;

rows deleted

get it back data -

SQL> rollback;

SQL> select * from first;

Diff. between delete and truncate?

27

- whenever we're using delete from tablename then automatically deleted data internally stored in buffer we can also get it back this data by using rollback.

whenever we're using truncate table tablename then all rows are permanently deleted we can't get it back this data by using rollback also because truncate is a DDL command and also DDL command transaction are automatically committed.

- DQL (Data Query Language) or (Data retrieval Language)

- Select -

Select command is used to fetch data from Database.

Syntax -

```
Select * from tablename;
```

Syntax

```
Select col1, col2, ... from tablename where condn group by columnname  
having condn order by columnname [asc/desc];
```

1. Select all columns & all rows.

*

where condⁿ

↳ true (or) false

2. select all columns & particular rows.

col₁, col₂, ...

3. Select particular columns & all rows.

4. select particular columns & particular rows.

- Creating a new table from existing table -
(or)

- Copying a table from another table -

```
Create table tablename as select * from existingtablename;
```

ex:- SQL> create table diwali as select * from emp;
SQL> select * from diwali;

Note -

In all databases whenever we are copying a table from another table internally constraints (Primary key, Foreign key..) are never copied.

- Creating a new table from existing table without copying Data:-

Syntax -

create table newtablename as select * from existtablename
where falsecondition;

SQL> create table abc as select * from emp where 1=2;
SQL> select * from abc;
no rows selected
SQL> desc abc;

Operators Used in Select Statement :-

- select \downarrow col₁, col₂ 1) Arithmetic operators (*, +, /, -)
 \downarrow 2) Relational operators (=, <, >, =>, <=, $\boxed{<> \text{ or } !=}$)
where \downarrow cond 3) Logical operators (And, Or, Not)
 \downarrow 4) Special operator

- Arithmetic operators are used in number, date, datatype column.

Q. Write a Query to display ename, sal, annsal from emp table.

SQL> select ename, sal, sal*12 ^{space} $\underbrace{\text{annsal}}$ _{column aliasname} from emp

Ename	Sal	ANNSAL
SMITH	2200	26400
ALEN	300	3600

Q. WAP to display the employee except JOB as CLERK from emp table.

SQL> select * from emp where Job <> 'CLERK';

Syntax

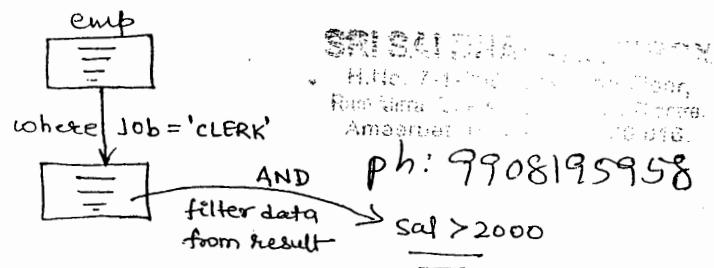
select * from tablename where columnname operator value;

Q. WAP to display the employee who are getting more than 2000 from emp table.

SQL> select * from emp where SAL > 2000;

Q. WAP whose CLERK having more than 2000 SAL from emp table.

SQL> select * from emp where Job = 'CLERK' and SAL > 2000;

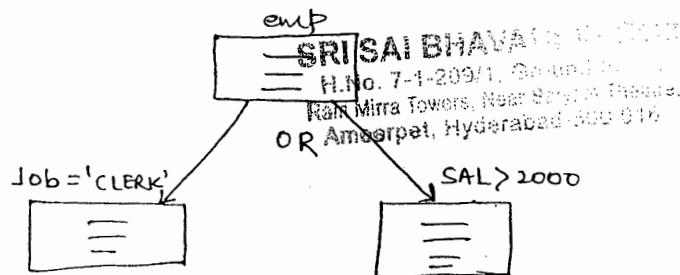


AND

whenever we are using logical operator "AND" then database serves filter the data from result set.

OR

select * from emp where Job = 'CLERK' OR SAL > 2000;



whenever we're using logical operator "OR" then DB server returns data based on each & every individual cond' from table.

Note - In all DB's if you want to retrieve multiple values from single column based on equality operator then we must use logical operator "OR".

ex:-

SQL > select * from emp where job = 'CLERK' OR job = 'SALESMAN'.

Q. WAP to display a employee whose Dept.No 20, 30, 50, 70, 80 from emp.table ?

SQL > select * from emp where deptno = 20 or deptno = 30
or deptno = 50 or deptno = 70 or deptno = 80;

* Special Operators -

i) in	not in
ii) between	not between
iii) is null	is not null
iv) like	not like

i) in :-

- It is used to pick the value one by one from list of values.
- generally we can also use "in" operator in place of "OR" operator when we are retrieving multiple values from single column. In this case in operator performance is very high compare to "OR" operator.

Syntax -

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

ex:-

SQL > Select * from emp where deptno in (20, 30, 50, 70, 80);

SQL > Select * from emp where ename in ('SMITH', 'ALLEN', 'MILLER');

Note -

In all DB's "not in" operator doesn't work with null values.

Ex:-

SQL> Select * from emp where deptno not in (10, 20, null);
 NO rows selected

null :-

Def - null is an undefined, unknown, unavailable value it is not same as zero.

- In all DB whenever we're using arithmetic operation and null value then it will becomes null.

Ex -

$$\text{null} + 50 = \text{null}$$

Q. WAP to display ename, sal, comm, sal+comm from emp whose ename is SMITH ?

SQL> select ename, sal, comm, sal+comm from emp where ename='SMITH';

ENAME	SAL	COMM	SAL+COMM
SMITH	2200

- Generally any arithmetic operation performed null value again its value is null to overcome this problem oracle provided NVL().

• NVL() -

Def - NVL is a predefined function which are used to replace (or) substitute user-defined value in place of null.

Syntax -

nvl(exp₁, exp₂)

Here exp₁, exp₂ must belongs to same datatype.

If exp₁ is null then it will return exp₂ otherwise it returning exp₁.

Ex - nvl(null, 20) → 20

nvl(10, 20) → 10

SRI SAI BHAVANI MARGA

H.No. 7-1-209/1, Opp. to Durgam
Ram Mirra Towers, Near Yediyur Bank,
Amarapet. Hyderabad-500 071.

Ph: 9908195958

Ex:-

SQL> Select ename, sal, comm, sal+nvl(comm, 0) from emp where
ename='SMITH';

ENAME	SAL	COMM	SAL+COMM
SMITH	2200	---	2200

$$\begin{aligned}
 &\Rightarrow \text{Sal} + \text{nvl}(\text{comm}, 0) \\
 &\Rightarrow 2200 + \text{nvl}(\text{null}, 0) \\
 &\Rightarrow 2200 + 0 \\
 &\Rightarrow 2200
 \end{aligned}$$

nvl2() -

- Oracle 9i introduced nvl() Function.

This function are also used to replace or substitute undefined value in place of null. This function accepts three parameters -

Syntax —

`nvl2(exp1, exp2, exp3)`

- Here if exp_1 is null then it will return exp_3 otherwise it will return expression_2 .

example —

① $\text{nvl2}(\text{null}, 10, 20) \rightarrow 20$

② $\text{nvl2}(30, 10, 20) \rightarrow 10$

Q) update the employee COMM within emp table by using following conditions through nvl2().

i) if comm is null then update comm to 500.

ii) if comm is not null then update comm to (comm+500).

SQL> update emp set comm = nvl2(comm, comm+500, 500);

2) between:-

It is used to retrieve range of value.

Syntax -

```
Select * From tablename
```

```
where columnname between low value and high value;
```

Q. WAP to displays the employees who are getting SAL between 2000-5000 from emp table?

```
SQL> Select * from emp  
      where sal between 2000 and 5000 ;
```

- When we want to sal except 2000 and 5000 between those then -

```
SQL> select * from emp where sal not between 2000 and 5000 ;
```

3) is null , is not null -

These two special operators are used in where condition only.
These operators are used to test whether a column having null value or not.

Note -

In all DB we're not allowed to use relational operators ; along with null value within with where condition.

Syntax -

- ① select * from tablename where columnname is null ;

- ② select * from tablename where columnname is not null ;

Q. WAP to display the employee's who are not get COMM from emp by using is null operator ?

```
SQL> Select * from emp where comm is null ;
```

Q. WAP to display the employee's who are getting Comm from emp by using is not null operator ?

```
SQL> select * from emp where comm is not null ;
```

4) Like :-

This operation is used to such data based on character pattern. Like operator performance is very high compare to predefined searching function.

-
- ```
graph TD; Like[Like] --> Percent["%
[string or group of character]"]; Like --> Underscore["- (underscore)
[single character]"]
```
- along with like operator we are using two special operators these are % and - (underscore).
  - These operators are also called as wildCard characters

#### Syntax :-

```
Select * from tablename where columnname Like
'character Pattern';
```

ex:-

- Q. WAP to display the employee's whose ename start with M from emp table by using like operator ?

SQL) select \* from emp where ename like 'M%' ;

O/P :- MARTIN  
MILLER

- Q. WAP to display the employee's whose ename having M in any position within ename column from emp table ?

SQL) select \* from emp where ename like '%.M.%.' ;

O/P:  
SMITH  
MARTIN  
ADAMS  
JAMES  
MILLER

- Q. WAP to display the employee's whose ename second letter is L from emp table by using like operator ?

SQL) select \* from emp where ename like '\_L%.' ;

ALLEN  
BLAKE  
CLARK

Q. WAP to display the employee's whose ename 4th letter is M from emp table.

SQL> select \* from emp where ename like '\_\_\_M%';

Q. WAP display the employee who are joining in the month of Dec from emp table by using like operator.

SQL> select \* from emp where hiredate like '%DEC%';

Q. WAP to display the employee who are joining in the year 81 from emp table by using like operator.

SQL> select \* from emp where hiredate like '%.81';

- SQL> insert into emp(empno,ename) values (1,'S-MITH');

SQL> select \* from emp;

Q. WAP to display the emp whose ename starts with "S\_" from emp table by using like operator?

SQL> select \* from emp where ename like 'S-%';

O/P: SCOTT [wrong result]  
SMITH  
S-MITH

- In oracle when we have column data having wild card characters and also when we're trying to retrieve data based on these characters by using like operators then oracle server returns wrong result. Because here oracle server treated as wild card character have special meaning. To overcome this problem oracle provided escape function along with like operator which escapes special meaning of the wild-card characters and also "\_"(underscore) treated as "-"(underscore) and also percent (%) sign, Percent sign (%) only.

#### Syntax

Select \* From tablename where columnname like 'character Pattern'  
escape 'escape character';

Note -

Escape character length must be 1-byte.

✓ SQL> select \* from emp where ename like 'S?\_%' escape '?' ;

S-MITH

✓ SQL> select \* from emp where ename like 'S1\_%' escape '1' ;

S-MITH

? → means after escape character underscore treated as underscore  
not treated as wild card character.

X SQL> select \* from emp where ename like 'S

diwali

\_%' escape '

diwali

' ;

More than 1 byte

SQL> insert into emp(empno, ename) values (2, 'S--MITH') ;

{ SQL> select \* from emp where ename like 'S1\_1\_%' escape '1' ;

Q. WAP to display the emp whose ename start with "S--" from emp table using like operator?

(OR) SQL> select \* from emp where ename like 'S2\_2\_%' escape '2' ;

### Concatenation Operator :- (||)

If we want to display column data along with literal string along with concatenation operator.

ex:-

SQL> select 'my employee names are' || ename from emp

O/P my employee names are : SMITH

my employee names are : ALLEN

:

:

Note - If we want to display our own space in bet<sup>n</sup> the

- columns and also if we want to display literal strings in bet<sup>n</sup> the columns then also we're using concatenation operator.

7) Select ename || ' ' || sal from emp;

## FUNCTIONS

14-Nov-15

- Functions are used to solve particular task.
- Functions also return a value.
- Oracle having two types of functions —
  - 1> Predefine functions
  - 2> Userdefine functions

### 1> Predefine Functions:-

The predefined functions are characterized into following types —

- (a) Number Functions
- (b) Character Functions
- (c) Date Functions
- (d) Group Functions (or) Aggregate Functions.

### a) Number Functions:-

These functions operate over number data.

#### (i) abs() -

It is used to convert -ve sign into +ve sign.

ex:- SQL> select abs(-50) from dual;

o/p : 50

dual → dual is a predefined virtual table which contains one column and one row.

- dual table is used to test predefined, Userdefine Functions functionality.

ex:- SQL> select nvl(null,10) from dual;

10

SQL> select nvl(20,10) from dual;

20

## SRI SAI BHAVANI VEROX

H.No. 7-1-209/1, Ground Floor,  
Ram Niwas Colony, Near Satyam Theatre,  
Ameerpet, Hyderabad-500 018.

SQL> select \* from dual;

O/P: 
$$\begin{array}{c} D \\ \times \end{array}$$

SQL> desc dual;

O/P: 

|       |             |
|-------|-------------|
| Name  | datatype    |
| Dummy | varchar2(1) |

Note -

By default dual table columns datatype is varchar2.

Ex:- SQL> select ename, comm, sal, abs(comm-sal) from emp where  
comm is not null;

(ii) \* mod(m,n) :-

. It will gives remainder after m/n

ex:- SQL> select mod(10,2) from dual;

O/P: 0

(iii) \* round(m,n) :-

It rounds given floated value number f based on n.

ex:- select round(1.7) from dual.

O/P: 2

select round(1.23456, 3) from dual

O/P: 1.235

Execution :-

Step-1      1.234      56  
                          above 50%  
                          (out of 100)

Step 2      1.234  
                  + 1  
                  1.235

Note - round always checks remaining number, if remaining no. is  
above 50% then automatically 1 added to the rounded no.

Q. SQL> Select round (1475.789, -1) From dual ;

O/P : 1480

execution: 1470      5 replaced with 0.

5 is above 50% (out of 10)

$$\begin{array}{r} \text{Step-2} \\ 1470 \\ \hline 1 \\ \hline 1480 \end{array}$$

SQL> select round(1295,-2) from dual;

O/P : 1300

Q: select ename, round(sal, -3) from emp;

|        |        |      |
|--------|--------|------|
| O/P :- | MILLER | 2000 |
|        | TURNER | 0    |
|        | ALLEN  | 7000 |

Q. Select ename, round(Sal/7, 2), round(Sal/7, 3). From emp;

(iv)  $\text{trunc}(m,n)$

If truncates gives floated value number m based on n.

SQL> select sumc(1.8) from dual;

O/P : 1

eg:- SQL> select trunc(1.23456,3) from dual;

0/R : 1.234

$$1 \cdot \underline{23456} \quad \overline{1}$$

(v<sub>0</sub>) (ell(), floor()) :-

These functions always return integer, ceil returns nearest greatest integer whereas floor returns nearest lowest integer.

SQL> select ceil(1.4) from dual;

O/P : 2

$$\textcircled{1} \quad \underline{\underline{1 \cdot 1 \quad 1 \cdot 2 \quad 1 \cdot 3 \quad 1 \cdot 4 \quad 1 \cdot 5 \dots}} \quad \textcircled{2}$$

SQL> select ceil(1.9) from dual;

O/P: 2

SQL> Select floor(1.9) from dual ① 1.1 1.2 1.3 . . . ②

O/P : 1

(vi)  $\text{greatest}(\text{exp}_1, \text{exp}_2, \dots, \text{exp}_n)$ ,  $\text{least}(\text{exp}_1, \text{exp}_2, \dots, \text{exp}_n)$  :-

$\text{greatest}$  returns maximum value among given expression where  
 $\text{least}$  returns minimum value among given expression.

eg - SQL> select greatest(5,8,9,7) from dual;

eg - SQL> select greatest(12,15,20) from dual;

eg - SQL> select ename, sal, comm, greatest(sal, comm) from emp  
where comm is not null;

### b. Character Functions :-

#### 1. UPPER() -

It is used to convert a string into uppercase and column  
into uppercase.

eg - SQL> select upper(ename) from emp;

eg - SQL> select upper(abc) from dual;  
O/P : ABC

#### 2. LOWER()

It is used to convert a string into lowercase.

eg - SQL> select lower(ABC) from dual;  
O/P : abc

SQL> select lower(ename) from emp;

O/P : miller  
allen  
King  
turner  
clerk

Permanently

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

14 rows updated

SQL> select \* from emp;

### 3. Initcap() :-

If returning 1st letter is capital and remaining letters are small.

eg:- SQL> select initcap(ename) from emp;

O/P: Smith

Allen

Turner

eg:- SQL> select initcap('ab cd ef') from dual;

O/P: Ab Cd Ef

### 4. Length() :-

Always it returns number datatype it returns total length of string including spaces.

eg:- select length ('AB CD') from dual;

O/P: 5

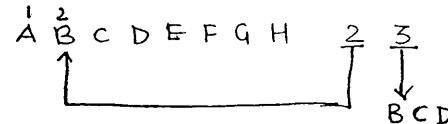
\*\*

### Substr() :-

It will extract portion of the string within given string based on last two parameters.

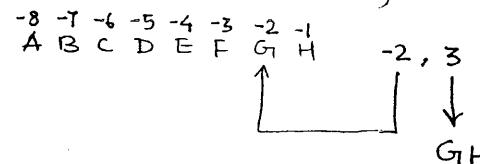
eg:- SQL> select substr('ABCDEFGHI', 2, 3) from dual;

O/P: BCD



eg:- SQL> select substr('ABCDEFGHI', -2, 3) from dual; (3-character)

O/P: GH



eg:- SQL> select substr('ABCDEFGHI', -4) from dual; (character start from -2 level)

O/P: EFGH

Syntax -

substr (Columnname, Starting Position, no. of characters from Position)  
 OR  
 substr (Columnname, Position, Number of characters)  
 OR  
 substr (Columnname, Position, Number of characters)  
 OR  
 substr (Columnname, Position, Number of characters)  
 OR  
 substr (Columnname, Position, Number of characters)

Column name / Variable / Constant / Expression / Function / Subquery / Anywhere, Hyphenated word is used

-ve      +ve

number

Position  
always left to right  
↳ always left to right

Q. WAP to display the employees whose ename 2nd letter would be 'LA' from emp table by using substr().

SQL> select \* from emp where substr(ename,2,2) = 'LA';

Q. WAP to display the employee whose ename length is 5.characters from emp table by using Length().

SQL> select \* from emp where length(ename) = 5;

O/P :- SMITH

CLARK

Note:- In all databases we're not allowed to use group function in "where" class. But we're allowed to use number(), character(), date() in "where" class.

e.g:- SQL> select \* from emp where sal = max(sal);

error: group function is not allowed here.

● instr() :-

instr always returns number datatype i.e. it returns position of the delimiter, position of the character, position of the string within given string.

ex:-

SQL> select instr('ABC\*D', '\*') from dual;

O/P : 4

Q: SQL> select instr('ABC<sup>1</sup>DEF<sup>2</sup>G<sup>3</sup>H<sup>4</sup>C<sup>5</sup>D<sup>6</sup>I<sup>7</sup>K<sup>8</sup>L<sup>9</sup>C<sup>10</sup>D<sup>11</sup>M<sup>12</sup>N<sup>13</sup>P', '(D', -6, 2)  
②      ①  
              . From dual;

O/P : 3

searching from right to left  
and print output left to right.

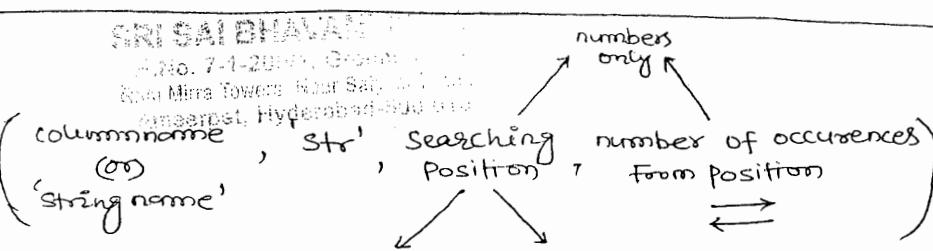
SQL>

Select instr ('ABCDEF<sup>1</sup>G<sup>2</sup>H<sup>3</sup>C<sup>4</sup>D<sup>5</sup>I<sup>6</sup>K<sup>7</sup>L<sup>8</sup>C<sup>9</sup>D<sup>10</sup>M<sup>11</sup>N<sup>12</sup>P', '(D', -5, 2) from dual;

O/P : 9

Syntax -

Instr



- always Instr() returns position based on last two Parameter but start counting number of characters left side first character onwards.

eg:- select Instr('AB\*CD\*EF', '\*', -3, 1) from dual;

O/P : 6

↑ Searching

Print whether 1st position occurrence  
of \*.

and point from left to Right.

1 2 3 4 5 6  
A B \* C D \* E F  
                 ↑

- Lpad()

It will fill remaining spaces with specified character on the left side of given string. Here always second Parameter returns total length of string.

Syntax -

Lpad

(column name  
(or)  
'string name') , total length , 'filled character') From dual;

eg - select Lpad('ABCD', 10, '#') from dual

O/P :-

----- A B C D

✓ #####ABCD

**SRI SAI BHAVAN**  
H.No. 7-1-2039, Ground Floor,  
Ram Mira Towers, Near S.S.J. Hospital,  
Ameerpet, Hyderabad-500 013.

Left side padding

- Rpad()

select Rpad('ABCD', 10, '#') from dual

A B C D -----

✓ ABCD#####

Right side padding.

eg:- select lpad ('ABC', 2, '\*') from dual;

O/P : AB

SQL> select rpad ('ABC', 2, '\*') from dual;

O/P : AB

Note:-

Whenever we are passing more no. of characters than the total length specified in second parameter then lpad() & rpad() returning some output because when we're submitting lpad() & rpad() into oracle server then oracle server always execute left side 1st character onwards within 1st parameter.

### ● LTRIM() :-

If removes specified character or the left side of the given string.

Syntax -

ltrim ( Column name  
(or)  
'String Name' , {Set of characters} )

eg - SQL> select ltrim ('SSMMESSHSS', 'S') from dual;

O/P : MISSTHSS

eg:- SQL> select job, ltrim (job, 'CSM') from emp;

| JOB      | LTRIM (JOB) |
|----------|-------------|
| CLERK    | LERK        |
| SALESMAN | ALESMAN     |
| MANAGER  | ANAGER      |

eg - SQL> select job, ltrim (job, 'CL') from emp;

| JOB      | LTRIM (JOB) |
|----------|-------------|
| CLERK    | ERK         |
| SALESMAN | SALES       |

- Rtrim() -

SQL> select rtrim ('SSMISSTHSS', 'S') from dual;  
O/P :- SSMISSTH

- Trim() -

oracle 8i has introduced TRIM(). It is used to remove left and right side specified character ; it is also used to remove leading, trailing spaces.

Syntax -

Trim ('character' from 'Stringname')

SQL> select trim ('S' FROM 'SSTHSSMISS') from dual;

O/P : THSSMI

TRIM() can also used as Ltrim() and Rtrim().

for Ltrim()

SQL> select trim(leading 'S' from 'SSMISSTHSS') from dual;  
O/P : MISSTHSS

- we can also convert Trim() to ltrim() by using leading class and also convert Trim() to Rtrim() by using trailing class.

For rtrim()

SQL> select trim(trailing 'S' from 'SSMISSTHSS') from dual;  
O/P : SSMISSTH

- Translate(), replace()

- Translate() is used to replaces character by character whereas as Replace() is used to replaces character by String (or) String by String.

SQL> select translate('india', 'in', 'xy') from dual;  
O/P : xydia

SQL> select replace('india', 'in', 'xy') from dual;  
O/P : xydia

Syntax -

H.No. 7-1-2027  
Roshni Mira Towers, New Road,  
Amarpet, Hyderabad-500003  
translate ('Str1', 'Str2', 'Str3');

eg -

SQL> select translate ('ABCDEF', 'FEDCBA', '123456') from dual;  
O/P: 654321

SQL> select replace ('A B C', ' ', 'PQR') from dual;  
O/P: A PQR B PQR C

SQL> select job, replace (job, 'SALESMAN', 'MARKETING') from emp  
O/P:- JHON SALES MAN MARKETING  
MILLER SALESMAN MARKETING

SQL> select replace ('SSMISSSTHSS', 'S') from dual  
O/P: MITH

- In oracle whenever we're not specifying third Parameter within replace function then automatically Second Parameter specify value permanently removed from the given string.

Q. WAP which counts no. of occurrence of the within given string SLEEP by using length/replace().

SQL> select length('SLEEP') - length(replace('SLEEP', 'E')) from dual;  
O/P: 2

• Concat() —

It is used to concatenate given two strings.

SQL> select concat ('wel', 'come') from dual;

O/P: welcome

### 3) Date

In oracle by default date format is DD-MON-YY  
oracle having following date() —

- 1) sysdate
- 2) add\_months()
- 3) last\_day()
- 4) next\_day()
- 5) months\_between()

#### 1) sysdate

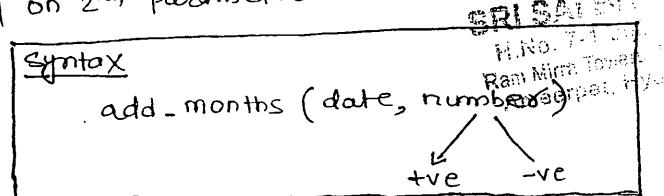
It returns current date of the system in oracle date format.

SQL> select sysdate from dual;

O/P: 17-NOV-15

#### 2) add\_months()

It is used to add or remove no. of months from the specified date based on 2nd parameter.



eg - select add\_months(sysdate, 5) from dual;

select add\_months(sysdate, 1) from dual;

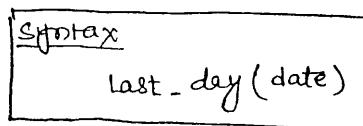
O/P: 17-DEC-15

Select add\_months(sysdate, -1) from dual;

O/P: 17-OCT-15

#### 3) last\_day()

It returns last date of the specified Month.



eg:- SQL> last\_day (sysdate) from dual;

20-NOV-15

#### 4) next\_day()

It returning next occurrence day from the specified date based on 2nd parameter.

Syntax:

next\_day (date, 'day')

eg :- select next\_day (sysdate, 'Monday') from dual;

O/P : 23-NOV-15

sysdate  $\rightarrow$  17-NOV-15

(TUESDAY)

#### 5) months\_between()

This function always returns number data type i.e. it returns no. of months between two specified date.

Syntax

months\_between(date1, date2)

Note :- Here date1 > date2 otherwise this fxn returns -ve sign.

select ename, round (months\_between (sysdate, hiredate)) from emp

### Date Arithmetic

(1) Date + number ✓

(2) Date - number ✓

(3) Date<sub>1</sub> + Date<sub>2</sub> ✗ not allowed.

(4) Date<sub>1</sub> - Date<sub>2</sub> ✓

SQL> select sysdate + 1 from dual;

SQL> select sysdate - 1 from dual;

SQL> select sysdate - sysdate from dual;

O/P: 0

Q. WAP to display first date of the current month by using sysdate, add\_months, last\_day().

SQL> select last\_day (add\_months (sysdate, -1)) + 1 from dual;

O/P : 01-NOV-15

Date Conversion Function:-

Oracle having following date conversion functions are-

- 1) TO\_CHAR()
- 2) TO\_DATE()

1) TO\_CHAR() :-

It is used to convert oracle date type into character type i.e.  
it converts date type into date string.

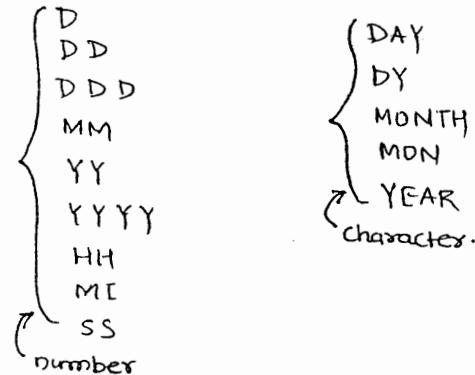
SQL> Select to\_char(sysdate, 'DD/MM/YY') from dual;

O/P: 18/11/15

- Basically To\_Char is caseSensitive function i.e. here To\_Char fxn character format are case sensitive format.

e.g. SQL> select to\_char(sysdate, 'DAY') from dual;  
O/P:- WEDNESDAY

SQL> select to\_char(sysdate, 'day') from dual;  
O/P:- wednesday



SQL> select to\_char(sysdate, 'DY') from dual;

O/P: WED

e.g.- SQL> select to\_char(sysdate, 'D');  
O/P: 4 ↑ day of the week (Sunday-1, Monday-2, ...)  
SQL> select to\_char(sysdate, 'DDD');  
O/P: 322 ↑ day of the year  
SQL> select to\_char(sysdate, 'DD');  
O/P: 18 ↑ day of the month.

SQL> select to\_char(sysdate, 'DDTH') from dual;

O/P: 18TH

SQL> select to\_char(sysdate, 'DDSPTH') from dual;

O/P: EIGHTEENTH

SP - spell out

SQL> select to\_char(sysdate, 'HH:MI:SS') from dual;

O/P: 02:51:33

SQL> select to\_char(sysdate, 'HH24:MI:SS') from dual;

O/P: 14:51:33

- The default oracle Time format has 12 hours.

Select to\_char('25-JUN-05', 'DD/MONTH/YY') from dual;

error

Note - whenever we're using To\_Char fxn always 1st Parameter must be oracle date type otherwise oracle server returns an error.

## 2) TO\_DATE() :-

It is used to convert date string into oracle date type.  
[oracle date format]

SQL> select to\_date('12/JUNE/05') from dual;

O/P: 12-JUN-05

SQL> select to\_date('12/06/05') from dual;

O/P: not a valid month

- whenever we're using TO\_DATE always Passes Parameter return values Matched with the default date format return values otherwise oracle server returns an error to overcome this problem use a second Parameter as same as 1st parameter format then only oracle server automatically converts date string into date-type.

ex:- SQL> select to\_date ('12/06/05', 'DD/MM/YY') from dual;  
 O/P :- 12-JUN-05

Q. What will add 5 days to the given date 09-feb-05 ?

SQL> select '09-FEB-05' +5 from dual;  
 error

SQL> select to\_date ('09-FEB-05') +5 from dual;  
 O/P : 14-FEB-05

SQL> select to\_date ('09-02-05') +5 from dual;  
 Error: not a valid month

SOL:-

SQL> select to\_date ('09-02-05', 'DD-MM-YY') +5 from dual;  
 14-FEB-05

Q. What to display given date into character format by using to\_char function ? [given date - 05-jun-1994] display into 'DAY/JUNE/1994'

SQL> select to\_char ('05-Jun-1994', 'DAY/MONTH/YY') from dual;  
 error

SQL> select to\_char (to\_date ('05-Jun-1994'), 'DAY/MONTH/YY')  
 from dual;

O/P :- THURSDAY/JUNE/1994

NOTE:- whenever we're using To\_char function formats MONTH, DAY then oracle server returns spaces passed Month, day occupies less no. of bytes than the maximum size to overcome this problem oracle provided fillMode within To\_char function i.e. fill mode used to suppress leading zero, trailing spaces.

ex:- ① SQL> select to\_char (to\_date ('12-June-05'), 'DD/MONTH/YY')  
 from dual;

O/P : 12/JUNE / 15

SQL> select to\_char (to\_date ('12-June-05'), 'DD/FMMONTH/YY')  
 from dual;

O/P : 12/JUNE/2015

Q. Write a query to display the employee who are joining Dec month from emp table by using To\_char function?

SQL> select \* from emp where to\_char(hiredate, 'MM') = '12';

(OR)

SQL> select \* from emp where to\_char(hiredate, 'MON') = 'DEC';

- SQL> select \* from emp where to\_char(hiredate, 'MONTH') = 'DECEMBER';  
NO row selected

SQL> select \* from emp where to\_char(hiredate, 'FMMONTH') = 'DECEMBER';

Q. Write a query to display the employees who are joining in the year 81 from emp table by using to\_char function.

SQL> select \* from emp where to\_char(hiredate, 'YY') = '81';

SQL> select hiredate, to\_char(hiredate, 'YYYY') from emp;

| O/P:- | Hiredate  | To-char |
|-------|-----------|---------|
|       | 17-dec-80 | 1980    |
|       | 20-feb-81 | 1981    |
|       | .....     | .....   |

Rule - In oracle whenever we're passing date string into pre-defined date function. Then oracle server automatically converts date string into date-type. That's why in this case to\_date function explicitly, but here passed parameter must be in oracle date format otherwise oracle server returns an error.

eg - AUTOMATIC CONVERSION -

SQL> select last\_day('15-AUG-05') from dual;

O/P: 31-AUG-05

explicit conversions -

SQL> select last\_day('15-08-15') from dual;

error

SQL> select last\_day(to\_date('15-08-15', 'DD-MM-YY')) from dual;

31-AUG-15

## Inserting dates into oracle table:-

53

Syntax  
columnname date

SQL> create table test (col1 date);

SQL> desc test;

SQL> insert into test values(sysdate);

1 row created

SQL> select \* from test;

col1

19-NOV-15

SQL> insert into test values ('15-aug-05');

1 row created

col1

19-NOV-15

15-AUG-05

SQL> insert into test values ( to\_date ('15-07-15', 'DD-MM-yy'));

1 row created

col1

19-NOV-15

15-AUG-05

15-JUL-15

## round(), trunc() functions are used in date:-

In oracle date datatype contains both date and time whenever we're using round, trunc() then automatically date part can be changed based on time portion and also time portion automatically set to zero.

- whenever we're using round() function oracle seize automatically adds one day to the given date if time portion is greater than (or) equal to 12 noon ( $\geq$  12 noon) and also time portion is automatically set to zero's.

e.g:- at 2:30 pm

→ SQL> select to\_char(sysdate, 'DD-MON-YY HH24:MI:SS') from dual;

20-NOV-15 14:33:17

→ SQL> select to\_char(round(sysdate), 'DD-MON-YY HH24:MI:SS') from dual;

21-NOV-15 00:00:00

- whenever we're using truncate Function then oracle server doesn't add one date to the given date if time portion is  $\geq$  12 noon also but internally time portion set automatically to zero.

eg - SQL> select to\_char(trunc(sysdate), 'DD-MON-YY HH24:MI:SS') from dual;  
O/P: 20-NOV-15 00:00:00

eg - SQL> insert into emp(empno, ename, hiredate) values (1, 'murali', sysdate);  
SQL> select \* from emp;

Q. WAP to display the employee's who are joining today from emp?

SQL> select \* from emp where hiredate = sysdate;  
no row selected

Internally oracle server compare dates based on date part, time portion.

SQL>. SQL> select \* from emp where trunc(hiredate) = trunc(sysdate);

Note :- In oracle round(), trunc() function also returns first date of the year, first date of the month based on following syntax.

Syntax :-  
round (date, 'year')  
Syntax :-  
round (date, 'month')  
Syntax :-  
trunc (date, 'year');  
Syntax :-  
trunc (date, 'month');

Note :- whenever we are rounding year oracle server checked out given month is above 50% or below 50%. (Jan-Jun). If it is above 50%, then it adds one year and also rounds to the Jan 1.

SQL> select :?; round(sysdate, 'year') from dual;

01-Jan-16

SQL> select round(sysdate, 'month') from dual;  
01-DEC-15

SQL> select round(sysdate, 'day') from dual;  
22-NOV-15

SQL> select trunc(sysdate, 'year') from dual;  
01-JAN-15

Hint:  
sysdate  
20-NOV-15

SQL> select trunc(sysdate, 'month') from dual;  
01-NOV-15

SQL> select trunc(sysdate, 'day') from dual;  
15-NOV-15

## d.) Group () function :-

Date  
20-NOV-15

In all databases group () functions operate over no. of values in a column and returns single value.

oracle server having following group function these are —

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

SRI SAI BHAVAHINI  
H.No. 7-1-209/1, Gopinathnagar,  
Rani Mirra Towers, New Sanjeevani  
Ameerpet, Hyderabad - 500042  
Tel: 040-23511111

### 1) max () -

It returning maximum value from a column table.

SQL> select max(sal) from emp;  
O/P : 7000

SQL> select max(hiredate) from emp;  
O/P : 23-MAY-87

SQL> select max(ename) from emp;  
O/P : WARD

### 2) min ()

SQL> select min(sal) from emp;  
O/P : 500

SQL> select min(hiredate) from emp;  
O/P : 17-DEC-80

SQL> select min(ename) from emp;

SQL> select \* from emp where sal = min(sal);  
error:- group function is not allowed here.

Note:- In all DB's we are not allowed to use group function in where cond.

### 3) avg() :-

It refers average from number datatype column

SQL> select avg (sal) from emp;

O/P: 2848.21429

SQL> select avg (comm) from emp;

O/P: 550

note: In all databases by default all group functions ignores null values except count(\*)

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

O/P: 157.14287

### 4) sum() :-

It returning total from number datatype column.

SQL> select sum(sal) from emp;

O/P: 39875

### 5) count(\*) :-

It counts no. of a rows in the table.

SQL> select count(\*) from emp;

O/P: 14

### 6) count(columnname) :-

It counts no. of not null values in a column.

SQL> select count(comm) from emp;

O/P: 9

SQL> select count(mgr) from emp;

O/P: 13

select distinct(deptno) from emp;  
O/P      deptno

10

20

30

select count(distinct(deptno)) from emp;  
O/P: 3

### Group by :-

group by clause is used to arrange similar data items into set of logical groups. whenever we're using group by clause Database Server selects similar data items from a table column and then reduces nos of data items in each group.

#### Syntax:-

```
select columnname, ... from table group by columnname;
```

Q. WAP to display no. of employee's in each department from emp table by using group by ?

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

| <u>deptno</u> | <u>count(*)</u> |
|---------------|-----------------|
| 10            | 3               |
| 20            | 5               |
| 30            | 6               |

Q. write a Query to display no. of employee's in each job from emp table by using group by ?

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

| <u>Job</u> | <u>count(*)</u> |
|------------|-----------------|
| Clerk      | 4               |
| Salesman   | 4               |
| President  | 1               |
| Manager    | 3               |
| Analyst    | 2               |

SRI SAI BHAVAN  
H.No. 7-1-209/1, Ground Floor,  
Rani Mirra Towers, Near Raja Bazaar,  
Ameerpet, Hyderabad - 500 011

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

| <u>Dept no</u> | <u>min(sal)</u> | <u>max(sal)</u> |
|----------------|-----------------|-----------------|
| 10             | 4100            | 7000            |
| 20             | 2200            | 4500            |
| 30             | 200             | 2950            |

Note:- In every databases we can also use group by class without using group functions .

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

Q.P:- DEPTNO

10  
20  
30

Rule :- Other than group function columns specified after select those all columns must be specified after group by otherwise oracle server returning an error "not a GROUP BY expression".

eg :- SQL> select deptno, sum(sal), job from emp group by deptno;  
error: not a GROUP BY expression.

SQL> select deptno, sum(sal), job from emp group by deptno, job;

eg :- SQL> select deptno, job from emp group by deptno, job;

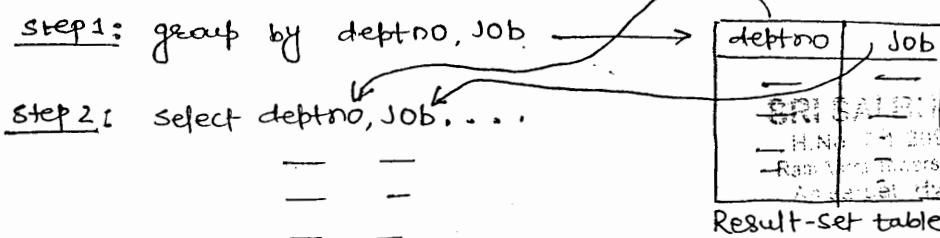
Note :- In all databases after group by class all specified column not required to specified after select.

eg - SQL> select deptno from emp group by deptno, job, sal;

Execution :-

In all Databases whenever we're submitting group by classes then database server first executes after group by class column and also then results are stored in result-set table then only Database servers select that result-set table data based on specified columns after select.

SQL> select deptno, job from emp group by deptno, job;



Note : In all Databases we are trying to display group function with normal function then Database server return errors to overcome this problem we must use group by class.

Step 1    eg - select max(sal) from emp;  
                        39875

Step 2    SQL> select deptno, sum(sal) from emp;  
error: not a single-group group function

Solution: SQL> select deptno, sum(sal) group by deptno;  
O/P:-

| Deptno | Sum SAL |
|--------|---------|
| 10     | 8450    |
| 20     | 15875   |
| 30     | 15450   |

Q. W&Q to display those dept having more than three employee from emp table ? using group by () . 59

SQL) select deptno, count(\*) from emp group by deptno where count(\*) > 3;  
exscr.

SQL) select deptno, count(\*) from emp group by deptno having count(\*) > 3;

| O/P: | DEPT NO | COUNT (*) |
|------|---------|-----------|
|      | 30      | 6         |
|      | 20      | 5         |

Having :-

after group by clause we are not allowed to use where class in place of this one ANSI/ISO SQL provided another clause having. which is like restrict after the clause.

- generally if we want to restrict rows in a table then we are using where clause. whereas if we want to restrict groups after group by then we must use having clause.
- generally we aren't allowed to use group function in where clause whereas in having clause we can also use group function.

Q. W&Q to display those dept. those of sum(sal) is more than 9000 from emp table by using group by ?

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

| O/P: | DEPTNO | SUM(SAL) |
|------|--------|----------|
|      | 20     | 15975    |
|      | 10     | 15450    |

Q. W&Q to display year, no. of employees per year in which more than 1 employee was hired. from emp table by using group by ?

SQL) select to\_char(hiredate, 'YYYY') "YEAR", COUNT(\*) from emp  
group by to\_char(hiredate, 'YYYY') having  
count(\*) > 1;

| O/P:- | YEAR | COUNT (*) |
|-------|------|-----------|
|       | 1987 | 2         |
|       | 1981 | 10        |

Note -

In all Databases whenever we are using group function within group by clause then oracle server internally stores all group function data that's why we can also restricting group after group by then we are using invisible function within having clause.

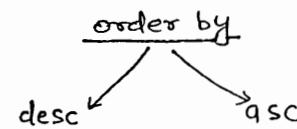
Q. WAP to display those deptno., sum(sal) from emp table those dept having more than 3 employee's by using having clause.

SQL> Select deptno, sum(sal) from emp group by deptno  
having count(\*) > 3;

- Order by

- Order by clause is used to either arrange in ascending order or descending order.

Along with order by clause we are using two keyword these are asc and desc



- In all databases by default order by clause having ascending order -

|        |
|--------|
| Syntax |
|--------|

|                                                            |
|------------------------------------------------------------|
| select * from tablename<br>order by columnname [asc/desc]; |
|------------------------------------------------------------|

eg. select sal from emp order by sal desc;

select \* from emp order by ename asc;

Syntax -

```

select col1, col2, ...
from tablename
where condition
group by columnname
having Condition
order by col [asc/desc];

```

SRI SAI DPP COLLEGE  
 H.NO. 744/200/200  
 Ram Niwas Pura, Agra  
 Uttar Pradesh, India

eg - select deptno, count(\*) from emp  
 where sal > 1000  
 group by deptno  
 having count(\*) > 3  
 order by deptno desc

| <u>DEPT NO</u> | <u>COUNT(*)</u> |
|----------------|-----------------|
| 30             | 4               |
| 20             | 5               |

• rollup, cube :-

oracle 8i introduced rollup, cube clauses these clauses are used along with group by clause only.

These clauses are used to calculate subtotal, grand total automatically.

Syntax :

```

select col1, col2, ...
From tablename group by Rollup (col1, col2, ...);

```

Syntax :

```

Select col1, col2 ...
From tablename
group by cube (col1, col2, ...)

```

SRI SAI DPP COLLEGE  
 H.NO. 744/200/200  
 Ram Niwas Pura, Agra  
 Uttar Pradesh, India

- If we want to calculate subtotal based on the single column then we are using rollup whereas if we want to cal. subtotal based on no. of column then we are using cube.

SQL> select deptno, job, sum(sal)  
from emp group by  
rollup(deptno, job);

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

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

Note - In all databases we can also use Multiple columns within order by clause. In this case database servers arrange data in shorting order based on first column after order by clause and then only second column is shorted based on each sub-group of the first column.

e.g. SQL> select deptno, sal from emp order by deptno, sal desc;

| <u>DEPTNO</u> | <u>SAL</u> |
|---------------|------------|
| 10            | 8350       |
|               | 7000       |
|               | 6000 ←     |
| 20            | 3300       |
|               | 3000       |
|               | 1800       |

SP15A1 DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING  
Rajiv Gandhi Technological University,  
Kukatpally, Hyderabad - 500 046.

## JOINS

- Joins are used to retrieve data from multiple table.
- In all databases if we are joining 'n' table then we are using  $(n-1)$  joining condition.
- Oracle Server having following types of joins:-

- 1) Equi Join (or) Inner Join
- 2) Non Equi Join
- 3) Self Join
- 4) Outer Join

- These 4 Joins are also called as 8i Joins.

### 9i Join (or) ANSI Joins -

- i) inner join
- ii) left outer join
- iii) right outer join
- iv) full outer join
- v) Natural join

**Note :** In oracle we can also retrieve data from multiple table without using joins i.e. whenever we are specifying no. of tables within from clause of the select stat. then oracle server internally uses cross join but cross join is implemented based on cartesian product. That's why this joins returns more no. of rows.

Eg - `select ename, sal, dname, loc from emp,dept;`

### 1) Equi Join or Inner Join -

- Based on equality condition we are retrieving data from multiple tables. here joining conditional columns must belongs to same datatype.
- whenever tables having common columns then only we are allowed to use Equi Joins and also these common columns must belongs to same data-type.

#### Syntax :-

```
Select col1, col2 ...
From tablename1, tablename2, ...
where tablename1.commoncolumn = tablename2.commoncolumn;
 join condition
```

eg:- select

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

- error: column ambiguously defined

Note - for avoiding ambiguity in oracle then we must specify tablename along with common column name by using dot(.) operator after select.

eg - select

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

Note - Always Equi joining returns matching rows only.

[here Deptno 10 doesn't display when we're using dept.deptno also]

Note: Generally to avoid future ambiguity then must specify tablename along with every columns within select list by using (.) dot operator.

-p- Using alias name:-

In joining we can also create alias names for the table within from clause. These alias names are also called as reference names for the table. These alias name must be different name. Once we are creating alias name in from clause then we must use those alias names in place of tablename within select list, within joining condition.

Syntax :-

```
from tablename1 aliasname1, tablename2 aliasname2;
```

eg:- select ename, sal, d.deptno, dname, loc

From emp e, dept d

where e.deptno = d.deptno ;

Q. What to display the employee who are working in the location "chicago" from emp & dept table by using Equi join?

SQL> select ename, loc from emp,dept  
where emp.deptno = dept.deptno  
AND loc = 'CHICAGO';

| <u>ENAME</u> | <u>LOC</u> |
|--------------|------------|
| WARD         | CHICAGO    |
| TURNER       | CHICAGO    |
| MILLER       | CHICAGO    |
| BLANKS       | CHICAGO    |
| MARTIN       | CHICAGO    |

SRI SAI BHAVAN  
H.No. 7-1-209/1, Gachibowli  
Ram Mirra Towers 11th Floor  
Ameerpet, Hyderabad - 500082

#### Note:-

If you want to filter data after joining condition then we're using AND operator in Bi Joining . whereas Gi Joining we are using either AND (or) where clause also.

Q. What to display the dname, sum(sal) from emp, dept table by using Equi join?

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

| <u>DNAME</u> | <u>SUM(SAL)</u> |
|--------------|-----------------|
| ACCOUNTING   | 15450           |
| RESEARCH     | 20150           |
| SALES        | 22100           |

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

| DeptNo | DName      | sum(sal) |
|--------|------------|----------|
| 10     | Accounting | 20000    |
| 20     | Sales      | 32010    |
| 30     | Research   | 42015    |

Q. Write to display loc, no. of employee's, Min. sal, max.sal from emp, dept table by using Equi Joining?

SQL) select loc, count(\*), min(sal), max(sal) from emp e, dept d  
where e.deptno = d.deptno  
group by loc;

O/P:

| LOC      | COUNT(*) | MIN(SAL) | MAX(SAL) |
|----------|----------|----------|----------|
| New York | 3        | 4100     | 7000     |
| Chicago  | 6        | 200      | 3000     |
| Dallas   | 5        | 2200     | 4500     |

SQL) select loc, count(\*), min(sal), max(sal) from emp e, dept d  
where e.deptno = d.deptno  
group by loc  
having sum(sal) > 10000;

RESEARCH            15450  
ACCOUNTING        15975

SQL) select dname, sum(sal) from emp e, dept d  
where e.deptno = d.deptno  
group by rollup(dname)

| DNAME      | SUM(SAL) |
|------------|----------|
| ACCOUNTING | 10000    |
| RESEARCH   | 22000    |
| SALES      | 32000    |
|            | 64000    |

← Grand total automatically  
when using rollup,

## 2) Non-Equi Joins.

Based on other than Equality Condition ( $<$ ,  $\leq$ ,  $>$ ,  $\geq$ ,  $\neq$ , between, etc.) we are retrieving data from multiple table.

Ex:- SQL> create table test1(deptno number(10));  
SQL> insert into test1 values (...);  
SQL> select \* from test1;

### DEPT NO

10

20

SQL> create table test2(deptno number(10));  
SQL> insert into test2 values (...);  
SQL> select \* from test2;

### Dept no

10

20

30

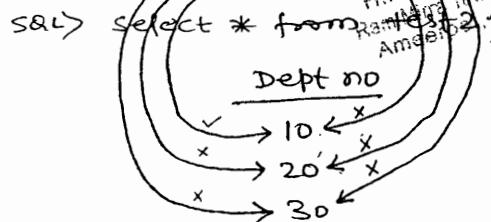
SQL> select \* from test1;

### Deptno

10

20

30



SQL> select \* from test1, test2  
where test1.deptno > test2.deptno;

| Deptno | Dept no |
|--------|---------|
| 20     | 10      |

Q: Note :- In oracle using Non-Equi join we can also retrieve data from multiple tables when table doesn't have common column but in this case one table one column values tie w/ another table to columns.

eg- SQL> select \* from emp;  
SQL> select \* from salgrade;  
SQL> select ename, sal, lsal, hsal from emp, salgrade  
where sal between lsal and hsal;

(OR)

SQL) select ename, sal, lsal, hsal from emp, salgrade  
where sal >= lsal and sal <= hsal;

### 3.) Self Join

- Joining a table itself is called self join.
- Here joining conditional columns must belongs to same date type.
- generally if we want to compare two column values from different tables and also if we want to retrieve matching rows then we are only using Equi Join whereas we want to compare two different column values within a same table then we must use self join; but here these column must belongs to same data type.
- Before we are using self join we must create table alias name in from clause. These alias name must be different names. These alias names are also known as reference name.
- These alias names internally behaves like a exact table when query execution time.

Syntax:-

From tablename aliasname1; tablename aliasname2;

- Q. WAP to display ename and mgrname from emp table by using self-join?

Select e1.ename "employees", e2.ename "manager"  
from emp e1, emp e2,  
where e1.mgr = e2.empno ;

Q. WAP To display the employee's who are getting more salary than their manager salary from emp table by using self join?

SQL> select e1.ename "employees", e2.ename "manager" From emp e1, emp e2  
where e1.mgr = e2.empno  
and e1.sal > e2.sal;

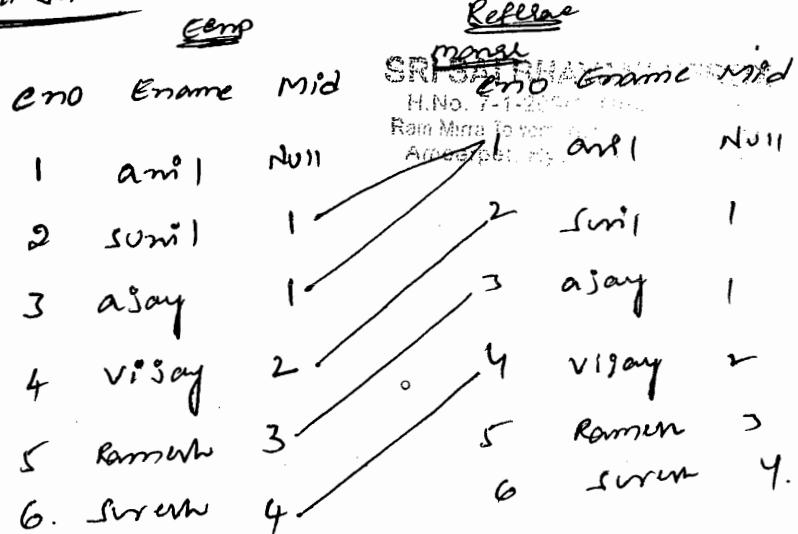
O/P:-

| <u>employees</u> | <u>sal</u> | <u>sal</u> | <u>manager</u> |
|------------------|------------|------------|----------------|
| FORD             | 3600       | 3475       | JONES          |
| SCOTT            | 4600       | 3475       | JONES          |

Q. WAP to display the employees who are joining before their managers from emp table by using self join?

SQL> select e1.ename "employees", e1.hiredate, e2.ename "manager", e2.hiredate  
From emp e1, emp e2  
where e1.mgr = e2.empno  
and e1.hiredate < e2.hiredate ;

Self join



O/P

| <u>ename</u> | <u>manager</u> |
|--------------|----------------|
| anil         | -              |
| sunil        | anil           |
| ajay         | anil           |
| vijay        | sunil          |
| Ramulu       | ajay           |
| Sreenu       | vijay          |

select e.ename, m.ename from emp e as employee self join  
select e.ename, m.ename from emp e as employee self join  
emp m as manager on m.empno = e.mid.

## 4) Outer join:-

- This join is used to retrieve all rows from one table and matching rows from another table.
- generally using Equi join we are retrieving Matching rows only if we want to retrieve non Matching rows also then we're using JOIN operator (+) within joining condition of the Equi join ; this is called oracle 8i outer join.

Note -

This join operator can be used only one side at a time within joining condition.

SQL select ename, sal, d.deptno, dname, loc

from emp e, dept d  
where e.deptno (+) = d.deptno  
↓                      ↑  
Matching rows        all rows

O/P:-     40   operation BOSTON

Note :- In oracle if we want to retrieve Matching or non-matching rows from all tables then we are using full outer join.

But full outer join is a Gi join. Prior to oracle 9i or if we want to retrieve all data then we are using join operator (+) within joining condition one time left side and another time right side.

ex:- Select

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

Union

select ename, sal, d.deptno, dname, loc  
from emp e, dept d  
where e.deptno = + d.deptno ;

## Gi Joins (or) ANSI Join :-

71

- 1> inner join
  - 2> left outer join
  - 3> Right outer join
  - 4> full outer join
  - 5> Natural join
- Full Outer Join :- Ram Maria from Dept 100, 101, 102.  
Natural Join :- Ram Maria from Dept 100, 101, 102.  
Amesgpt. 100, 101, 102.

### 1> inner join:-

- This join also returns matching rows only, here also join conditional column must belongs to same datatype. when tables having common columns then only we are allowed to use inner join.
- Inner join performance is very high compare to oracle 8i Equi join.

eg:- select  
 ename, sal, d.deptno, dname, loc  
 from emp e join dept d  
on, e.deptno = d.deptno;

Q. WAP to display the employee's who are working in loc "Chicago" from emp, dept table using inner join (gi).

eg. SQL> select ename, loc from emp join dept  
 on emp.deptno = dept.deptno  
 where loc = 'CHICAGO';

(or)  
 AND loc = 'CHICAGO';

| ENAME  | LOC     |
|--------|---------|
| WARD   | CHICAGO |
| TURNER | CHICAGO |
| FORD   | CHICAGO |
| ALLEN  | CHICAGO |

### \* using clause :-

In Gi Joins we can also use using clause in place of on clause generally using clause performance very high compare to on clause and also using clause returns common column one time only.

#### Syntax :-

```
Select * From tablename1 join tablename2
using (commoncolumnname1, commoncolumnname2);
```

eg - SQL> create table z<sub>1</sub>(a varchar2(10), b varchar2(10), c varchar2(10));

SQL> insert into z<sub>1</sub> values(...);

SQL> select \* from z<sub>1</sub>;

| A | B | C |
|---|---|---|
| x | y | z |
| p | q | r |

SQL> create table z<sub>2</sub>(a varchar2(10), b varchar2(10));

SQL> insert into z<sub>2</sub> values(...);

SQL> select \* from z<sub>2</sub>;

| A | B |
|---|---|
| x | y |
| s | t |

SQL> select \* from z<sub>1</sub>, z<sub>2</sub> where z<sub>1</sub>.a = z<sub>2</sub>.a and z<sub>1</sub>.b = z<sub>2</sub>.b;

| A | B | C | A | B |
|---|---|---|---|---|
| x | y | z | x | y |

gi inner join -

SQL> select \* from z<sub>1</sub> join z<sub>2</sub> on z<sub>1</sub>.a = z<sub>2</sub>.a and z<sub>1</sub>.b = z<sub>2</sub>.b;

| A | B | C | A | B |
|---|---|---|---|---|
| x | y | z | x | y |

using clause

SQL> select \* from z<sub>1</sub> join z<sub>2</sub>

using (a,b);

| A | B | C |
|---|---|---|
| x | y | z |

note - whenever we using using-clause then we are not allowed to use alias name in joining conditional columns.

eg - select e.ename, sal, d.deptno, d.dname, loc

from emp e join dept d

using (deptno)

error

SQL select

ename, sal, deptno, dname, loc from emp e join dept d  
using (deptno);

2) Left Outer Join :-

This join always return 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 - SQL> select \* from z<sub>1</sub> left outer join z<sub>2</sub> on z<sub>1</sub>.a = z<sub>2</sub>.a and  
z<sub>1</sub>.b = z<sub>2</sub>.b;

| A | B | C | A | B |
|---|---|---|---|---|
| x | y | z | x | y |
| p | q | r | - | - |

3) Right Outer Join:-

This join always return all rows from Right side table and matching rows from left side table and also returning null values in place of non-matching rows in another table.

eg :- SQL> select \* from z<sub>1</sub> right outer join z<sub>2</sub> on z<sub>1</sub>.a = z<sub>2</sub>.a and  
z<sub>1</sub>.b = z<sub>2</sub>.b;

| A | B | C | A | B |
|---|---|---|---|---|
| x | y | z | x | y |
| - | - | - | s | t |

4) full Outer join -

- This join returns all rows from all tables because it is combination of left, right-outer joins.

This join also returning null values in place of not matching rows in other table.

eg - select \* from z<sub>1</sub> full outer join z<sub>2</sub> on z<sub>1</sub>.a = z<sub>2</sub>.a and  
z<sub>1</sub>.b = z<sub>2</sub>.b;

| A | B | C | A | B |
|---|---|---|---|---|
| x | y | z | x | y |
| p | q | r | - | - |
| - | - | - | - | - |

H.No. 1/1/2021  
Rama Mira Institute of  
Amritsar, Punjab

## 5. > natural join -

This join also returns Matching rows only, This join performance very high compare to inner join.

In this join we are not required to use joining condition explicitly. But in this case resource tables must have a common column name based on this common column oracle server only internally automatically established joining condition.

Syntax -

```
select * from tablename1 natural join tablename2;
```

Note:- Whenever we are using Natural join oracle server always returns all common columns one time only because natural join internally use using clause.

eg:- Select \* from z<sub>1</sub> natural join z<sub>2</sub>;

O/P :-

| A | B | C |
|---|---|---|
| x | y | z |

Note - whenever we using natural join also then we aren't allow to use alias name to joining conditional columns because natural join internally uses using clause.

eg - SQL> select ename, sal, deptno, dname, loc from emp e  
natural join dept d;

## \* Cross Join

select ename, sal, dname, loc from emp cross join dept;

when emp - 14 rows  
Dept - 4 rows

Then output - 14\*4  
56 rows selected

## Joining More than two tables :-

Ex:- Join 3 tables

### 3 i joining

#### • Syntax

Select col<sub>1</sub>, col<sub>2</sub> ... from table<sub>1</sub>, table<sub>2</sub>,  
table<sub>3</sub>  
Where table<sub>1</sub>.commoncol = table<sub>2</sub>.commoncol  
And  
table<sub>2</sub>.commoncol = table<sub>3</sub>.commoncol

### 3 i joining (or) ANST join

#### Syntax -

Select col<sub>1</sub>, col<sub>2</sub> ...  
From table<sub>1</sub> Join table<sub>2</sub>  
On table<sub>1</sub>.commoncol = table<sub>2</sub>.common  
col  
Join table<sub>3</sub>  
On table<sub>2</sub>.commoncol = table<sub>3</sub>.common  
col

**SRI RAM COLLEGE**  
H.No. 7-4-420/1/2/3/4  
Ram Bhava Towers, 4th Floor  
Ameerpet, Hyderabad, Telangana.

**SRI RAM COLLEGE**  
H.No. 7-4-420/1, 4th Floor,  
Ram Bhava Towers, Ameerpet, Hyderabad-500 016.

# CONSTRAINTS

Date  
28/Nov/15

- Constraints are used to prevent invalid data entry into our tables.
- Generally constraints are created on table columns.
- Oracle Server having following types of constraints -
  - 1> not null
  - 2> unique
  - 3> Primary key
  - 4> Foreign key
  - 5> check
- In all databases all above constraints are defined into two ways -
  - (i) column level
  - (ii) table level

## (i) column level :-

In this method we are defining constraints in individual columns i.e. whenever we're defining column then only we are specifying constraints type.

### Syntax -

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

## (ii) Table level :-

In this method we are defining constraints on group of columns i.e. first we specifying all columns and last only we're specifying constraints type, along with group of columns.

### Syntax -

```
create table tablename (col1 datatype(size), col2 datatype(size),...
 constrainttype (col1, col2, ...));
```

SRI SAI  
H No 741, 2nd  
Rani Market, Gopalganj  
Amesha Patel Vidyapeeth

## 1. NOT NULL :-

- In all databases NOT NULL doesn't support table-level.
- NOT NULL doesn't accept NULL value but it accepts duplicate values.

### Column level :-

e.g. SQL> Create table z<sub>1</sub> (sno number(10) not null, name varchar2);

SQL> insert table z<sub>1</sub> values(null, 'a');

Error: cannot insert Null into sno

SQL> insert table z<sub>1</sub> values(1, 'X');

SQL> insert table z<sub>1</sub> values(1, 'Y');

| SNO | Name |
|-----|------|
| 1   | X    |
| 1   | Y    |

## 2. Unique :-

- In all databases unique constraints defined on column level, table level.
- Unique constraints doesn't accept duplicate value but accept Null values.

Note:- whenever we are creating unique constraint internally oracle server automatically creates B-tree indexes for those columns.

### Column level :-

SQL> Create table z<sub>2</sub> (sno number(10) unique, name varchar2(10));

### table level :-

SQL> Create table z<sub>3</sub> (sno number(10), name varchar2(10), unique(sno, name));

SQL> select \* from z<sub>3</sub>;

| SNO | Name   |
|-----|--------|
| 1   | Murali |
| 1   | abc    |

SQL> Insert into z<sub>3</sub> values(1, 'abc').

Error: unique constraint violated.

SRI SAI INSTITUTE OF COMPUTER SCIENCE & TECHNOLOGY  
H.No. 737, Ram Niwas Colony, Amrapur, Agra.  
Amrapur, Agra.

3) Primary key :-

Primary key uniquely identifying a record in a table, in all data

- based their can be only one primary key in a table and also primary key doesn't accept duplicate values, null values.

- whenever we are creating primary key also oracle server internally automatically creates B-tree indexes on those columns.

column-level :-

### Table level:-

```
SQL> create table z5 (sno number(10), name varchar2(10),
 Primary Key (sno, name));
```

This is also called as composite primary key i.e. it is a combination of columns as a single primary key.

#### 4. Foreign key :-

- In all databases if you want to establish relationship b/w tables then we are using referential integrity constraints foreign key.
  - One table foreign key must belong to another table primary key and also these two columns must belongs to some datatype.
  - Always Foreign key values always based on primary key values only.
  - Generally primary key doesn't accept duplicate, Null values whereas foreign key accepts duplicate, null values.

Column level (Reference8) —

## Syntax -

SQL) create table tablename (col1 datatype(size) references  
mastertablename (primarykey coln)

SQL> Create table M1 (sno number(10) references Z4);

Table Level:— (foreign key, references)

Syntax:

```
create table tablename (col1 datatype(size), col2 datatype(size)...
 foreign key (col1, col2...) references
 Mastertablename (Primary key colnames));
```

eg- SQL> create table Z5 (sno number(10), name varchar2(10),
 Foreign key (sno, name) references Z5);

- whenever we are establishing ref' between table by using Foreign key then oracle server automatically violet following two rules; these are—

- a> deletion in Master table.
- b> Insertion in child table.

#### a> Deletion in master table:

When we are trying to delete a master table record in master table if the record exists in child table then oracle server returns an error ora-2292. To overcome this problem if we want to delete master table record in master table then first we must to delete child table record in child table then only we are allow to delete those record in master table otherwise use an on delete cascade clause.

#### On delete cascade -

This clause is used along with foreign key only. whenever we are specifying this clause in child table then we are deleting a master table record within master table then automatically the record is deleted in master table and those records are automatically deleted in child table.

Syntax:

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

ex:-

SQL> create table mas (sno number(10), Primary key);

SQL> insert into mas values (...);

SQL> select \* from mas;

O/P: 

| SNO |
|-----|
| 1   |
| 2   |
| 3   |

SQL> create table child ( sno number(10) references mas on delete cascade);

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

SQL> select \* from child;

O/P: 

| sno |
|-----|
| 1   |
| 1   |
| 2   |
| 2   |
| 3   |

### TESTING (DELETION IN MASTER TABLE) :

SQL> delete from mas where sno = 1;

One row deleted

SQL> select \* from mas;

O/P: 

| sno |
|-----|
| 2   |
| 3   |

SQL> select \* from child;

O/P: 

| sno |
|-----|
| 2   |
| 2   |
| 3   |

### on delete set null :-

oracle also supports another clause along with foreign key on delete set null. whenever we use this clause, whenever we are deleting primary key value in master table then automatically that record is deleted in master table and corresponding foreign key value are automatically set to null in child table.

Syntax :- create table tablename ( col1 datatype (size) references  
mastertablename (primarykey colname) on delete set null  
... );

**MANOJ ENTERPRISES**  
Plot No: 40, Gayathri Nagar,  
Ameerpet, Hyderabad,  
Cell: 8125378496

### b) insertion in child table :-

- When we are try to insert other than primary key values into foreign key then oracle server returning an error ORA - 2291 because in all databases always foreign key values based on primary key values only.

Ex:- SQL> insert into child values (5);

ERROR: ORA-2291: integrity constraint violated Parent key not found

Solution -

SQL> insert into mas values (5);

SQL> select \* from mas;

Values

3  
5

SQL> insert into child values (5);

O/P:-      sno  
3  
3  
5

Note - Generally when we are truncating Master table by using "truncate table tablename" then DB servers returns error to overcome this problem Oracle 12c introduced cascade clause along with truncate table tablename.

Syntax

truncate table mastertablename cascade;

Before we are using this command we must use on delete cascade class along with foreign key

Ex:- SQL> create table mas (sno number(10) primary key);

SQL> insert into mas values (...);

SQL> select \* from mas;

sno  
1  
2  
3

SQL> create table child (sno number(10) references mas on delete

cascade);  
SQL> insert into child values (...);

SQL> select \* from child

sno  
2

```
SQL> truncate table mas;
```

ERROR: Unique/Primary keys in table referenced by enabled Foreign key.

```
SQL> truncate table mas cascade;
```

```
SQL> select * from mas;
```

```
SQL> select * from child;
```

## 5. Check :-

- Check constraints are used to define logical conditions according to client business rules.

In oracle check constraints does not work with sysdate function.

### Column level :-

#### Syntax -

```
SQL> create table tablename (col1 datatype(size) check(logical cond),
 col2 datatype(size),--);
```

ex-1:

```
SQL> create table test (sal number(10) check(sal > 2000));
```

```
SQL> insert into test values(1000);
```

Error:

```
SQL> insert into test values(5000);
```

1 row created

```
SQL> select * from test;
```

| SAL  |
|------|
| 5000 |

ex-2

```
SQL> create table test1 (name varchar2(10)
```

check(name=upper(name)));

```
SQL> insert into test1 values('abc');
```

Error

```
SQL> insert into test1 values('ABC');
```

1 row created

```
SQL> select * from test1;
```

O/P :- 

| name |
|------|
| ABC  |

table level :-Syntax -

```
create table tablename (column1 datatype(size), column2 datatype(size)
 check (cond1, cond2 . . .));
```

ex -

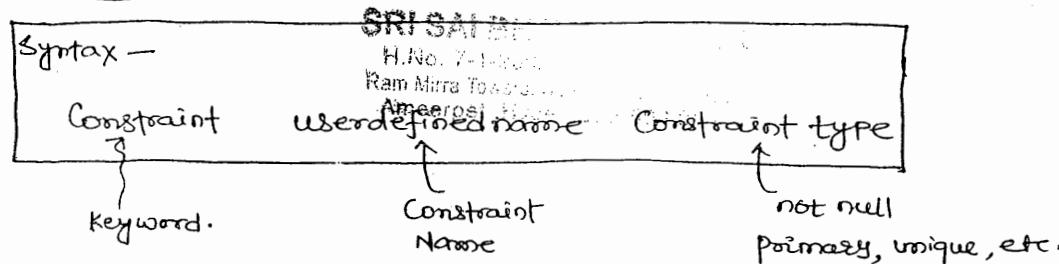
```
SQL> create table test1 (name varchar2(10), sal number(10),
 check (sal > 2000 and name=upper(name)));
```

- Assign user defined names to constraints :-

(OR)

Constraints Names :-

- In all DB whenever we are creating constraints then DB Server automatically converts an unique identification no for identifying a constraint uniquely. Oracle Server also automatically generates an unique identification number in the format of sys\_constraint. This is called predefined constraint name. In place of any number of this one we can also create our own name by using constraints keyword. This is called userdefined constraint name.



ex - Predefined Constraint name -

```
SQL> create table test(sno number(10) primary key);
```

```
SQL> insert into test values(1); ✓
```

```
SQL> insert into test values(1); X
```

Error: unique constraints

(SCOTT\_SYS (005571)) violated.

ex - User defined constraint name -

SQL> create table test1 (sno number(10) constraint P\_sno Primary key);

SQL> insert into test1 values(1); ✓

SQL> insert into test1 values(1); x

Error: Unique Constraints

(SCOTT\_P.SNO) violated

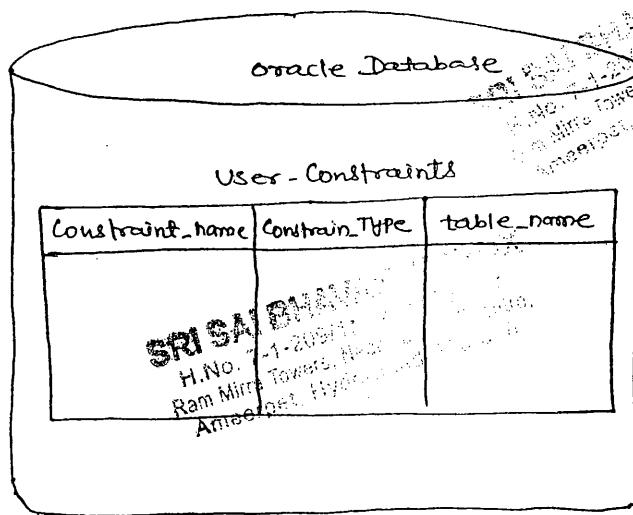
↑ user-defined constraint name.

SRI SAI INSTITUTE OF MANAGEMENT

Hall No. 7-1, Sector 7, Noida  
Phone: 0120-4567890, 0120-4567891  
Address: Sector 7, Noida, U.P., India-201307

## DATA DICTIONARY

- whenever we are installing oracle server then automatically so many read only tables are created these tables store specific objects information and also store specific objects related information.



Note 1: In oracle all constraints information stored under User\_Constraints data dictionary.

ex- SQL> desc user\_constraints;  
SQL> select constraint\_name, constraint\_type from user\_constraints  
where table\_name = 'EMP';

| CONSTRAINT NAME | CONSTRAINT_TYPE |
|-----------------|-----------------|
| PK_EMP          | P               |
| FK_DEPTNO       | R               |

ex- create table emp (empno number(4) constraint PK\_emp primary key  
ename varchar2(10), ... , deptno number(2) constraint FK\_deptno  
references dept);

#### Note-2.

In oracle if you want to view column names along with  
constraints name then we are using USER\_CONS\_COLUMNS  
data dictionary.

SQL> desc user\_cons\_columns;  
SQL> select column\_name, constraint\_name from user\_cons\_columns  
where table\_name = 'EMP';

| COLUMN NAME | CONSTRAINT NAME |
|-------------|-----------------|
| EMPNO       | PK_EMP          |
| DEPTNO      | FK_DEPTNO       |

#### Note-3 -

In oracle if you want to view logical condition of the check  
constraint then we are using search\_condition property from  
user\_CONSTRAINTS data dictionary.

ex- SQL> create table test (name varchar2(10) check(name=upper(name))  
SQL> desc test;  
SQL> desc user\_constraints;  
SQL> select search\_condition from user\_constraint where  
table\_name = 'TEST';

#### SEARCH\_CONDITION

name = upper(name)

## default clause -

In oracle we can also provide default values for a column by using default clause.

### Syntax -

```
columnname datatype(size) default defaultvalue;
```

ex:-

```
SQL> create table test (name varchar2(10), sal number(10), default(4000));
```

```
SQL> insert into test (name) values ('abc');
```

```
SQL> select * from test;
```

|      |      |      |
|------|------|------|
| O/P: | Name | sal  |
|      | abc  | 4000 |

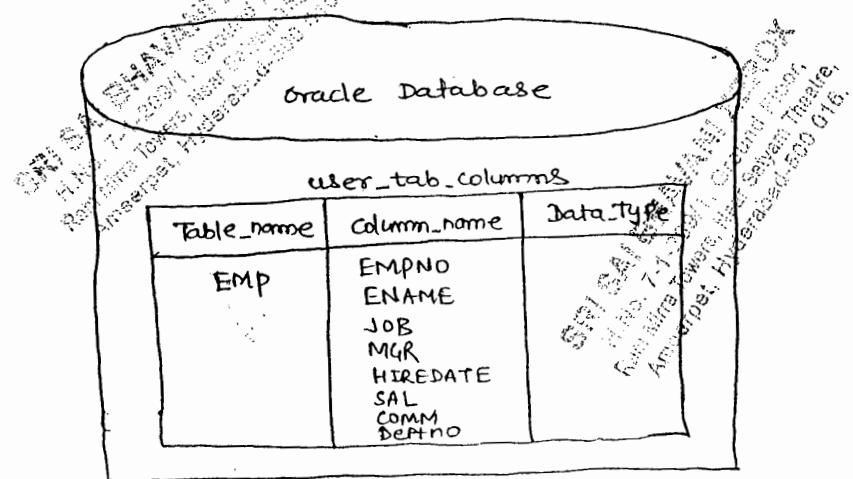
- In oracle if you want to view default values for a column then we are using `data_default` property from `user_tab_columns` data dictionary.

ex -

```
SQL> desc user_tab_columns;
SQL> select column_name, data_default from user_tab_columns
 where table_name = 'TEST';
```

|      |             |              |
|------|-------------|--------------|
| O/P: | COLUMN_NAME | DATA_DEFAULT |
|      | SAL         | 4000         |

- In oracle all columns info. stored under `user_tab_columns` data dictionary



ex -

```
SQL> desc user_tab_columns;
SQL> select column_name from user_tab_columns
 where table_name='EMP';
```

O/P :- COLUMN\_NAME

- EMPNO
- ENAME
- JOB
- MGR
- HIREDATE
- SAL
- COMM
- DEPTNO

Q. WAP which counts no. of columns in Emp table by using data dictionary?

```
SQL> select count(*) from user_tab_columns where table_name =
 'EMP';
```

O/P :- Count (\*)

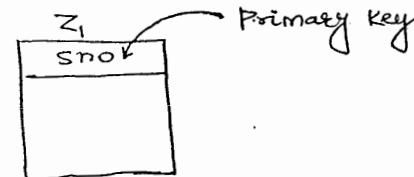
8

- ADDING (OR) DROPPING CONSTRAINTS ON EXISTING TABLE :-

using alter command we can also add constraints on existing table.

Note - In oracle if we want to add constraints on existing table & existing columns then we must use table level syntax methods.

```
SQL> Alter table Z1 add primary key (Sno);
```



Note - we want to add new column along with constraints then we are using column level syntax method.

```
SQL> alter table Z1 add name varchar2(20) unique;
```

Ex:- SQL> create table Z2 (Sno number(10));

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

Note - In oracle if we want to add not null constraint on existing table, existing column then we are using alter with modify

Syntax -

|                                                         |
|---------------------------------------------------------|
| alter table table tablename modify columnname not null; |
|---------------------------------------------------------|

Ex:- create table z3 (sno number(10));

SQL> alter table z3 modify sno not null;

Ex- SQL> alter table z3 add name varchar(20) not null;

SQL> desc z3;

Note- In all databases whenever we copying a data from another table except not null all other constraints are never copied.

Ex:- SQL> create table dept1 as select \* from dept;

SQL> alter table dept1 add primary key (deptno);

SQL> create table emp1 as select \* from emp;

SQL> alter table emp1 add primary key (empno);

SQL> alter table emp1 add Foreign key (deptno) references dept1 (deptno) on delete cascade.

Ex- SQL> create table z4 as select \* from z3;

SQL> desc z4;

### Dropping Constraints :-

method 1:

#### Syntax

Alter table tablename drop constraint Constraintname;

Method 2:

#### Syntax —

Alter table tablename drop primary key;

#### Syntax —

Alter table tablename drop unique (col1, col2 ...);

Ex- SQL> create table e1 (sno number(10) primary key);

Table created

SQL> alter table e1 drop Primary key;

Note —

If you want to drop primary key along with referenced foreign key then we are using cascade along with alter-drop.

ex—

`alter table tablename drop primary key cascade`

e.g. 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;  
table altered

SQL> desc z3;

SQL> desc user\_cons\_columns;

SQL> select column\_name, constraint\_name from user\_cons\_columns  
where table\_name = 'z3';

| COLUMN_NAME | CONSTRAINT_NAME |
|-------------|-----------------|
| SNO         | SYS_C005579     |
| NAME        | SYS_C005580     |

SQL> alter table z3 drop constraints SYS\_C005580;

SQL> desc z3;

- Query within another query is also called as subQuery (or) Nested Query.
- Subqueries are used to retrieve data from single or multiple table based on more than one step process.
- All DB systems having two types of Subqueries. These are -
  - I> Non-correlated Subqueries
  - 2> Correlated - Subqueries
- In non-correlated subqueries child queries are executed first then only parent queries executed.  
Whereas in correlated subqueries parent queries executed first then only child queries executed.

#### I> Non-correlated Subqueries -

- Non-correlated subqueries mainly consist of two part - child queries (or) inner query  
parent query (or) outer query

#### a> Child Queries -

A query which provides values to the another queries is called child query.

#### b> Parent Query

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

#### Non-correlated subquery

- 1. → Single row subqueries
- 2. → Multiple row subqueries
- 3. → Multiple column subqueries

↳ Inline views (or) subquery are used from clause.

Q. WAP To display the employee's who are getting more salary than the avg SAL from emp table?

SQL> Select \* from emp where SAL > (select avg(SAL) from emp);

In the above Queries ; this is single row subquery because child Query returning single values ; in single row subquery we are using =, <, <=, >, >=, between operators.

Execution -

Step 1: SQL> select avg(sal) from emp;  
O/P: 3491.07143

Step 2: SQL> select \* from emp where sal > 3491.07143;

Q. WAP to display the employees who are working in SALES Dept. From emp, dept table ?

SQL> select \* from emp where deptno = (select deptno from dept);  
where dname = 'SALES');

- 1> 90% same column
- 2> 5% group fxn
- 3> 3% expression
- 4> 2% different columns

Q. WAP to display senior most employee detail from emp table ?

SQL> select \* from emp where Hiredate = (select Min(hiredate) from emp);

Note :-

→ generally in all DB we're not allowed to use child Query table column with Parent Query because SubQuery always returns Parent Query table col. in output, To overcome this problem we must use Joins within Parent Query.

e.g:- SQL> select ename, dname from emp e, dept d  
where e.deptno = d.deptno  
and d.deptno = (select deptno from dept where dname='SALES')

| O/P: | ENAME  | DNAME |
|------|--------|-------|
|      | ALLEN  | SALES |
|      | WARD   | SALES |
|      | SMITH  | SALES |
|      | JORDAN | SALES |

Q. W4Q to display the employees who are working same as 'SMITH' deptno from dept table?

SQL> select \* from emp where deptno = (select deptno from emp where ename = 'SMITH');

Q. Write a Query ~~to~~ to display the employee who are getting more salary than the highest Paid employee in 20<sup>th</sup> Dept in emp table?

SQL> select \* from emp where sal > (select max(sal) from emp where deptno = 20);

Q. W4Q to display deptname of the highest paid employee from emp, dept table?

SQL> select dname from dept where deptno = (select deptno from emp where sal = (select max(sal) from emp));

O/P:- DNAME  
ACCOUNTING

Q. W4Q to display 2<sup>nd</sup> max salary from emp table?

SQL> select max(sal) from emp where sal < (select max(sal) from emp);

Q. W4Q to display 2<sup>nd</sup> highest salary detail from emp table?

SQL> select \* from emp where sal = (select max(sal) from emp where sal < (select max(sal) from emp));

Q. W4Q to display lowest avg salary Job from emp table by using group by?

- SQL> select job, avg(sal) from emp  
group by job having  
avg(sal) = (select min(avg(sal)) from emp);

error: nested group function without GROUP BY

- SQL> select job, avg(sal) from emp  
group by job having  
avg(sal) = (select min(avg(sal)) from emp group by job);

O/P:- JOB AVG(SAL)  
SALESMAN 1962.5

Note -

In all DB whenever child query having nested group function then we must use group by clause in child query.

Q. WAP to display job, avg salary which job, avg salary more than clerk job avg salary?

SQL> Select job, avg(sal) from emp  
group by job having  
avg(sal) > (select avg(sal) from emp where job = 'CLERK');

O/P: 

| JOB       | Avg(SAL) |
|-----------|----------|
| PRESIDENT | 8000     |
| MANAGER   | 4291.667 |
| ANALYST   | 4600     |

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

O/P: 

| DEPT NO | MIN(SAL) |
|---------|----------|
| 20      | 2900     |
| 10      | 5100     |

Q. WAP to display the employee's who are working under 'BLAKE' as Manager by using empno, mgr columns from emp table?

Select \* from emp where

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

Multiple row SubQuery :-

when ever child query returns these type of subquery is called multiple row subquery.

- In all databases we are using in all. any operators in multiple row subqueries.

Note -

"we can also use 'in' operator in single row subquery."

Q. WAP to display the employee detail, who are getting Max(sal) in each Dept from emp table?

SQL> Select \* from emp

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

Error: single row subquery return more than one row.

- This is multiple row subquery because here child query returning multiple values in multiple row subquery we using 'in' operator in the place of "=" operator.

SQL> Select \* from emp

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

| DEPTNO | SAL  | NAME  |
|--------|------|-------|
| 30     | 3550 | BLAKE |
| 20     | 5200 | SCOTT |
| 10     | 8500 | KING  |

SQL> Select deptno, sal, ename from emp

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

Q. WAP to displays the employee's who are working in 'SALES' or 'RESEARCH' Department from dept, emp table by using subquery

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

#### • Multiple Columns SubQuery

In all DB we can also compare multiple columns values of child values with the multiple column values of Parent Query these types of subquery are also called as multiple column subquery. In this case we must specified Parent Query where condition column within Parenthesis () -

Syntax :-

Select \* from tablename  
where (col1, col2 ...) in (select col1, col2, ...) from  
tablename where condition'

Q. WAP to display the employee's who's job, mgr match with the job, mgr of the employee 'SCOTT' from emp ?

SQL> select \* from emp  
 where (job, mgr) in (select job, mgr from emp where  
 ename='SCOTT');

Q. write a multiple row subquery to display the emp. who are getting max. sal from each dept. from emp table ?

SQL> update emp set sal = 3550 where ename = 'FORD';

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

| DEPT NO | SAL  | ENAME |
|---------|------|-------|
| 10      | 8550 | KING  |
| 20      | 3550 | FORD  |
| 30      | 3550 | BLAKE |
| 20      | 5200 | SCOTT |

Q:- write a query to display the employees who are getting max. sal from each dept from emp table using multiple columns Sub Query.

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

| DEPTNO | SAL  | ENAME |
|--------|------|-------|
| 10     | 8500 | KING  |
| 20     | 5200 | SCOTT |
| 30     | 3550 | BLAKE |

Q. WAP to display ename, dname, sal of the employee's whose sal, comm match with the sal, comm of the employee's working in the location 'DALLAS'.

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

Q. Write a multiple column subquery to display senior most employee in each job from emp table?

SQL> select job, hiredate, ename from emp  
where (job, hiredate) in (select job, min(hiredate) from emp,  
group by job)

| O/P: | JOB       | HIREDATE  | ENAME |
|------|-----------|-----------|-------|
|      | CLERK     | 17-Dec-80 | Smith |
|      | SALESMAN  | 20-Feb-81 | Allen |
|      | PRESIDENT | 17-Nov-81 | King  |
|      | MANAGER   | 02-Apr-81 | Jones |
|      | ANALYST   | 03-Dec-81 | Ford  |

## TOP-N Analysis

Date - 08/Dec/15

1. inline views

2. rownum

### 1. inline Views :-

- oracle 7.2 introduced inline views.
- Inline views a special type of query having a sub-query in from clause of the parent query.
- Generally we are not allowed to use order by clause in child query to overcome this problem oracle 7.2 introduced subqueries in from clause in parent query these type of queries are also called as inline views.

Syntax -

select \* from (subquery);

- Generally we aren't allowed to use column alias name in where clause if we want to use column alias name in where clause then we must us inline views, because in this case alias names behaves like a adject column name.

SQL> select ename, sal, sal\*12 annsal from emp where annsal > 3000  
error: "ANNSAL": invalid identifier

sol:- SQL> select \* from (select ename, sal, sal\*12 annsal from emp)  
where annsal > 3000;

| O/P:- | Enname | Sal  | AnnSal |
|-------|--------|------|--------|
|       | SMITH  | 3000 | 36000  |
|       | JONES  | 4075 | 48900  |
|       | ...    | ...  | ...    |
|       |        |      |        |

select \* from ( 

|       |     |        |
|-------|-----|--------|
| ename | sal | annsal |
|       |     |        |

 ) where AnnSal > 3000;

## 2) rownum :-

- rownum is pseudo column which behaves like a table column.
- rownum is used to restrict no. of rows in a table.
- whenever we are installing oracle server then automatically some pseudo columns are created in a database. These pseudo columns belongs to all databases. These are rownum, rowid, ...
- rownum is pseudo column which automatically assign no. to each row in a table at the time of selection.

SQL> select rownum, ename from emp;

| Rownum | ENAME |
|--------|-------|
| 1      | SMITH |
| 2      | ALLEN |
| 3      | SMITH |
| :      | :     |

Generally rownum having temporary values.

ex:- select rownum, ename from emp where deptno = 10;

| O/P:- | Rownum | Enname |
|-------|--------|--------|
|       | 1      | CLARK  |
|       | 2      | KING   |
|       | 3      | MILLER |

Q. What to display first row from emp table by using rownum ?

SQL> select \* from emp where rownum = 1 ;

Q. What to display second row from emp table by using rownum ?

SQL> select \* from emp where rownum = 2 ;

NO row selected X

- Generally rownum doesn't work with more than 1 positive integer i.e. it works with <, <= operators.

Q. Write a Query to display 1st five row from emp table by using rownum ?

SQL> select \* from emp where rownum <= 5 ;

- Generally rownum pseudo column works based on following pseudo code in oracle.

```
rownum = 1
for i in (select statement)
 loop
 if (test condition) then output a row
 rownum = rownum + 1;
 end if;
 end loop;
```

Ex -

SQL> select \* from emp where rownum <= 2 ;
2 rows are selected ;

Execution -

```
Step.1: rownum = 1
 for
 loop
 if (i <= 2) then output a row
 rownum = rownum + 1;
 end if;
 end loop;
```

```
Step.2:
 rownum = 1
 for
 loop
```

```
if (2 <= 2) then output a row;
 rownum = rownum + 1;
end if;
end loop;
```

99

Q. WAP to display first five highest salary employee from emp table by using rownum?

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

Q. WAP to display 5<sup>th</sup> highest salary emp from emp table by using rownum?

SQL) 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. WAP to display 2<sup>nd</sup> row from emp table by using row num?

SQL) select \* from emp where rownum <= 2

minus  
select \* from emp where rownum <= 1;

Q. WAP to display rows between 1 to 5 from emp table by using rownum?

SQL) select \* from emp where rownum between 1 and 5;

Q. WAP to display rows between 3 to 7 from emp table by using rownum?

SQL) select \* from emp where rownum <= 7  
minus  
select \* from emp where rownum <= 3;

SQL) select \* from emp where rownum >= 1  
rows are selected.

SQL) select \* from emp where rownum > 1;  
no rows selected.

Q. WAP to display last two rows from emp table by using rownum ?

SQL> select \* from emp  
minus

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

- whenever we are creating a rownum alias name in inline views then that alias name works with all SQL operators.

Q. WAP to display 2<sup>nd</sup> row from emp table by using rownum alias name ?

SQL> select \* from (select rownum r, ename, sal from emp)  
where r=2;

O/P:

| R | ENAME | SAL |
|---|-------|-----|
| 2 | ALLEN | 900 |

To display all column —

SQL> select \* from (select rownum r, emp.\* from emp)  
where r=2;

Q. WAP to display rows between 3 to 7 from emp table by using rownum alias name .

SQL> select \* from (select rownum r, ename, sal from emp)  
where r between 3 and 7;

O/P:

| R | ENAME  | SAL  |
|---|--------|------|
| 3 | WARD   | 2500 |
| 4 | JONES  | 4700 |
| 5 | MARTIN | 5300 |
| 6 | BLAKE  | 1560 |
| 7 | CLARK  | 7600 |

Q. WAP to display 2<sup>nd</sup>, 3<sup>rd</sup>, 7<sup>th</sup>, 10<sup>th</sup> records from emp table by using rownum alias name .

SQL> select \* from (select rownum r, ename, sal from emp)  
where r in(2,3,7,10);

Q. WAF to display 1<sup>st</sup> row, last row from emp table by using 101  
rownum alias name?

SQL> select \* from (select rownum r, ename, sal from emp)  
where r=1 or r=(select count(\*) from emp);

O/P:-

| R  | ENAME | SAL  |
|----|-------|------|
| 1  | ALLEN | 3000 |
| 14 | SMITH | 5000 |

Q. WAF to display even no. of record from emp table by using  
rownum alias name?

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

Q. WAF to display 5<sup>th</sup> highest salary employee from emp table  
by using rownum alias name?

SQL> select \* from (select rownum r, ename, sal from (select \*  
from emp order by sal desc)) where r = 5;

O/P:-

| R | ENAME | SAL  |
|---|-------|------|
| 5 | JONES | 4075 |

### Analytical Function:-

\* Analytical function are used in inline views:-

- Oracle 8i introduced analytical function.
- Analytical fx<sup>n</sup> also similar to group function, generally group function  
reduces no. of rows in each group whereas Analytical functions  
doesn't reduces no. of rows in each group, i.e. analytical fx<sup>n</sup>  
executes each & every row within a group.  
example - group fx<sup>n</sup>

SQL> select deptno, avg(sal) from emp group by deptno;

O/P:-

| DEPTNO | AVG(SAL)  |
|--------|-----------|
| 10     | 6783.3333 |
| 20     | 3785      |
| 30     | 1991.6667 |

example - analytical function

| <u>DEPTNO</u> | <u>AVG(SAL)</u> |
|---------------|-----------------|
| 10            | 6783.3333       |
| 10            | 6783.3333       |
| 10            | 6783.3333       |
| 20            | 6783.3333       |
| 20            | 3785            |
| 20            | 3785            |
| 20            | 3785            |
| :             | 3785            |

- oracle having following analytical function -

These are -

- ① row\_number()
- ② rank()
- ③ dense\_rank()

These three analytical function automatically assign ranks, either group by or rows wise in a table.

Syntax -

Analytical function name () over [ Partition of columnname  
order of columnname(s)asc/dsc];

row\_number Analytical function automatically assigns different rank number when values are same where as rank, dense\_rank analytical function automatically assigns same rank numbers when values are same and also rank skips next consecutive number (1,1,3,4,5) where as dense\_rank does not skip next consecutive rank numbers.

row\_number()

O/P:-

| <u>deptno</u> | <u>ename</u> | <u>sal</u> | <u>r</u> |
|---------------|--------------|------------|----------|
| 20            | JONES        | 5200       | 1        |
| 20            | SCOTT        | 5200       | 2        |
| 20            | FORD         | 3550       | 3        |

rank()

|    |       |      |   |
|----|-------|------|---|
| 20 | JONES | 5200 | 1 |
| 20 | SCOTT | 5200 | 1 |
| 20 | FORD  | 3550 | 3 |

| <u>deptno</u> | <u>ename</u> | <u>sal</u> | <u>r</u> |
|---------------|--------------|------------|----------|
| 20            | JONES        | 5200       | 1        |
| 20            | SCOTT        | 5500       | 1        |
| 20            | FORD         | 3550       | 2        |

Q. w4Q to display the employee details highest salary to lowest salary and also automatically assigns rank in each dept from table by using analytical function?

SQL> select \* from (select deptno, ename, sal, row\_number() over(partition by deptno order by sal desc) r from emp) where r <= 10;

\*\*\* Q. w4Q to display 2<sup>nd</sup> highest salary employee in each dept from emp table by using analytical function?

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

\*\*\* Q. w4Q to display 5<sup>th</sup> highest salary employee from emp table by using analytical function?

SQL> select \* from (select deptno, ename, sal, dense\_rank() over(order by sal desc) r from emp) where r = 5

#### NOTE:-

In analytical function Partition by clause is an optional clause.

\*\*\* Q. w4Q to display n<sup>th</sup> highest salary employee from emp table by using analytical function?

select \* from (select deptno, ename, sal, dense\_rank() over(order by sal desc) r from emp) where r = &n;

Enter value for n :- 1.

| Deptno | Ename  | Sal   | r |
|--------|--------|-------|---|
| 20     | DARLKS | 20000 | 1 |

### Row-Id

- row\_id is a pseudo column, it behaves like a table column.
- Generally, rownum having temporary values whereas rowid having fixed values.
- whenever we are inserting data into table then oracle server automatically generates a unique identification number for identifying the record uniquely, this is called row\_id.

Ex:- select rownum, rowid, ename from emp;

SQL> select rownum, rowid, ename from emp where deptno=10;

- In oracle if you want to retrieve data very fastly then using row\_id.
- we can also use min, max functions into its rowid.

eg:- SQL> select min(rowid) from emp;

SQL> select max(rowid) from emp;

- In oracle by default rowid having ascending order.

Q. WAP to display first row from emp table by using rowid?

SQL> select \* from emp where rowid = (select min(rowid) from emp);

Q. WAP to display last row from emp table by using rowid?

SQL> select \* from emp where rowid = (select max(rowid) from emp);

### Note:-

we can also use rowid in order by clause within analytical fx

Q. WAP to display 2<sup>nd</sup> row from emp table by using analytical fx, rowid?

SQL> select \* from (select ename, sal, deptno, row\_number() over (order by rowid) r from emp) where r=2;

Q. WAP to display last two record from employee table by using analytical function and rowid?

SQL> select \* from (select ename, sal, deptno, row\_number() over (order by rowid desc) r from emp) where r <= 2;

Q. w4Q to display 2<sup>nd</sup> row from each dept by using analytical Fx,  
rowid from emp table ? 105

SQL> select \* from (select ename, sal, deptno, row\_number() over  
(partition by deptno order by rowid) r from emp)  
where r=2;

O/P:

| Ename | sal  | Deptno | r |
|-------|------|--------|---|
| King  | 8900 | 10     | 2 |
| Jones | 5100 | 20     | 2 |
| Ward  | 2600 | 30     | 2 |

Note:-

If oracle if you want to delete duplicate row from a table, then we must use rowid.

Q. w4Q to display duplicate data from emp ?

SQL> select \* from first;

| sno |
|-----|
| 10  |
| 10  |
| 10  |
| 20  |
| 20  |
| 30  |
| 30  |
| 40  |
| 50  |

SQL> select sno, count(\*) from first group by sno having count(\*) > 1;

| sno | count(*) |
|-----|----------|
| 30  | 2        |
| 20  | 2        |
| 10  | 3        |

\*\*\*\*

Q. write a query to delete duplicate rows from the above table by using rowid ?

(or)

delete duplicate row except 1 row from the above table ?

SQL> delete from first where rowid not in (select min(rowid) from first group by sno);

SQL> select \* from first;

| sno |
|-----|
| 10  |
| 20  |
| 30  |
| 40  |
| 50  |

## 2. > Co-related SubQuery :-

- generally in non-correlated subquery, child query is executed first then parent query is executed.
  - whereas in correlated subquery parent query executed first then after child query is executed.
  - generally in non-correlated subquery child query is executed only once per parent query whereas correlated subquery child query is executed for each row per parent query table.
  - In correlated subqueries we must create an alias name for parent query table and then pass the alias name in child's query in "where" condition.

## Syntax -

select \* from tablename aliasname  
where columnname = (select \* from tablename where  
columnname = aliasname . columnname)

- generally correlated subquery used in denormalization process  
In this process we are using correlated update.  
If we want to modify 1-table column value based on another table column then only we're using correlated updates.

## Syntax

update tablename<sub>1</sub> aliasname<sub>1</sub> set  
 columnname = (select columnname from tablename<sub>2</sub> aliasname<sub>2</sub>  
 where aliasname<sub>1</sub> • Commoncolumn = aliasname<sub>2</sub> • common  
 name  
 column  
 name<sub>1</sub>)

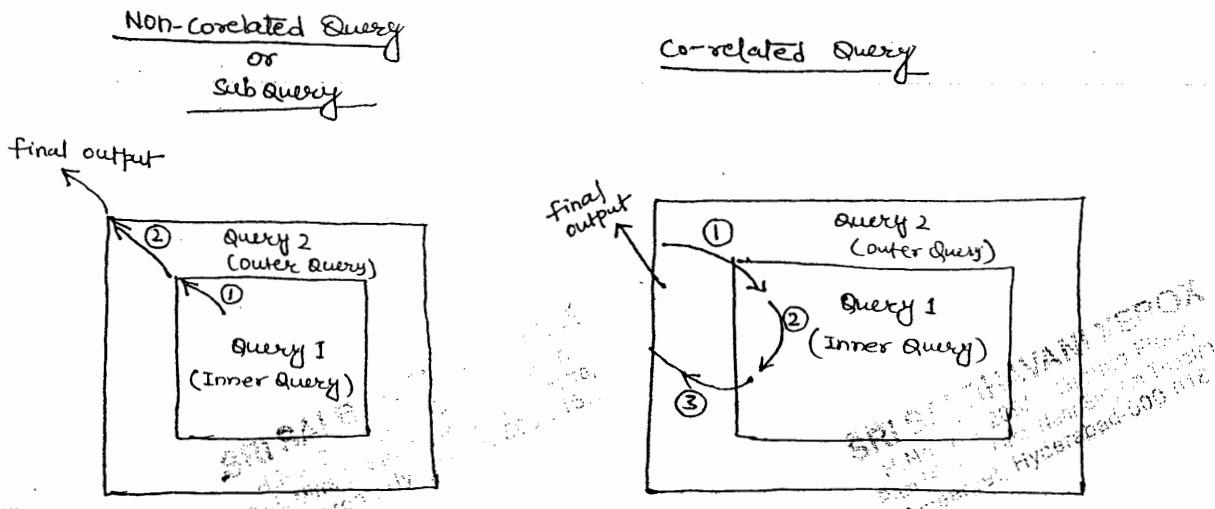
eg - SQL> alter table emp add dname varchar2(10);

SQL) select \* from emp;

SQL> update emp e set domane = (select domane from dept d  
where e.deptno = d.deptno):

SQL) select \* from emp;

- Whenever we are submitting co-related subquery into the database Server then database server get a candidate row from Parent query table and then control passed into child query where condition and then based on evaluation values, it compares value with parent query.



\*\*\* Q. WAP to display 1st highest salary employee from emp table by using co-related SubQuery ?

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

O/P: KING 7200

Q. WAP to display 2<sup>nd</sup> highest salary employee from the following table by using correlated SubQuery ?

SQL> create table test (ename varchar2(10), sal number(10));

SQL> insert into test values ('...');

SQL> commit;

SQL> select \* from test;

| Enname | sal |
|--------|-----|
| murali | 100 |
| XYZ    | 150 |
| PQR    | 200 |
| abc    | 300 |

SQL> Select \* from test e<sub>1</sub> where 2 = (select count(\*) from test e<sub>2</sub>

where e<sub>2</sub>.sal >= e<sub>1</sub>.sal);

| Enname | sal |
|--------|-----|
| PQR    | 200 |

## Execution

### Phase-I

Step-1 :- get the candidate row  
[first row  $\rightarrow$  murali 100]

Step 2 :-

Select count (\*) from test e<sub>2</sub> where  $e_2 \cdot \text{sal} >= e_1 \cdot \text{sal}$  ;

| Test   |     |
|--------|-----|
| ename  | sal |
| murali | 100 |
| abc    | 150 |
| xyz    | 200 |
| pqr    | 300 |

[1]

Step 3 :-

Select \* from test e<sub>1</sub> where 2 = 1 ;  
[false]

### Phase-II [if Phase-I is false]

Step-1 :- get the candidate row  
[abc, 150]

Step-2 :-

Select count (\*) from test e<sub>2</sub> where  $e_2 \cdot \text{sal} >= e_1 \cdot \text{sal}$  ;

[3]

| Test   |     |
|--------|-----|
| ename  | sal |
| murali | 100 |
| abc    | 150 |
| xyz    | 200 |
| pqr    | 300 |

150

Step-3 :-

Select \* from test e<sub>1</sub> where 2 = 2 ;  
[false]

### Phase-III [if Phase-II is false]

Step-1 :- get the candidate row  
[xyz 200]

Step-2 :-

Select count (\*) from test e<sub>2</sub> where  $e_2 \cdot \text{sal} >= e_1 \cdot \text{sal}$  ;

[2]

200

| Test   |     |
|--------|-----|
| ename  | sal |
| murali | 100 |
| abc    | 150 |
| xyz    | 200 |
| pqr    | 300 |

Step-3 :-

Select \* from test e<sub>1</sub> where 2 = 2 ;  
[True]

Output : xyz 200

- whenever resource table columns having duplicate data then the above query doesn't return any row.
- To overcome this problem then we must use distinct clause within count() function.

ex- SQL> insert into test values ('aaa', 200);

SQL> commit;

SQL> select \* from test;

| Ename  | sal |
|--------|-----|
| murali | 100 |
| abc    | 150 |
| xyz    | 200 |
| pqr    | 300 |
| aaa    | 200 |

Ans: 2 rows selected  
Ph: 9908195156

SQL> select \* from test e<sub>1</sub> where 2 =

= (select count(\*) from test e<sub>2</sub> where e<sub>2</sub>.sal >= e<sub>1</sub>.sal);

O/P: no row selected

### Solution -

SQL> select \* from test e<sub>1</sub> where

2 = (select count(distinct(sal)) from test e<sub>2</sub> where e<sub>2</sub>.sal >= e<sub>1</sub>.sal);

O/P:

| Ename | sal |
|-------|-----|
| xyz   | 200 |
| aaa   | 200 |

Q. WAP to display 2<sup>nd</sup> highest salary employee from the above table by using correlated subquery (n-1)<sup>th</sup> method ?

SQL> select \* from test e<sub>1</sub> where (2-1) =

(select count(distinct(sal)) from test e<sub>2</sub> where e<sub>2</sub>.sal > e<sub>1</sub>.sal);

O/P:

| Ename | sal |
|-------|-----|
| xyz   | 200 |
| aaa   | 200 |

Q. WAP to display n<sup>th</sup> highest salary employee from the emp table by using correlated subquery ?

SQL> select \* from emp e<sub>1</sub> where &n =

(select count(distinct(sal)) from emp e<sub>2</sub> where e<sub>2</sub>.sal > e<sub>1</sub>.sal);

O/P: enter the value for n : 1

KING 7000

Q. wAQ to display the employee's who are getting more salary than the Avg(sal) of their jobs from emp table by using co-related subQuery ?

SQL) select \* from emp e where  
 $\text{sal} > (\text{select avg(sal)} \text{ from emp where job} = \text{e.job})$ ;

- Exists Operator -

- Exists operator always return boolean value either true or false.
- Exists operator performance is very high compare to in operator.
- Exists operator used in co-related subQuery.
- Exist operator is used in where condition only, when we are using exists operator then we're not allowed to use column-name along with exists operator.

Syntax -

select \* from tablename aliasname

where exists (select \* from tablename where tablename=aliasname  
Columnname)

- generally if you want to test one table column value available or not available in another table column when both the tables are related then only we are using exists operator.
- exists operator is used to test whether given set is empty or not empty, exists operator on non-empty set returns true, whereas exists operator on empty set returns false.

ex - ① exists {1, 2, 3, 4} = True

② exists {} = False

Q. wAQ to display department from dept table those dept having employees in emp table by using correlated subquery exists operator.

SQL) select \* from dept d where exists

(select \* from emp where deptno = d.deptno);

O/P :-

| Deptno | Dname      | Loc      |
|--------|------------|----------|
| 10     | Accounting | New York |
| 20     | Research   | Dallas   |
| 30     | Sales      | Chicago  |

Q. What is to display those department from dept table having employee's in emp table by using non-correlated subquery in operator? 111

SQL> select \* from dept where deptno  
in (select deptno from emp);

| Deptno | Dname      | Loc      |
|--------|------------|----------|
| 10     | Accounting | New York |
| 20     | Research   | Dallas   |
| 30     | Sales      | Chicago  |

Note -

exists operator performance is very high compare to in operator.

Q. Difference between in, exists operator?

Ex -  
Dept → 1000 rows  
emp → 1000 rows

in  
• SQL> select \* from dept  
where deptno in (select deptno  
from emp);  
  
Execution - All rows in emp table will  
read for every row in dept table  
i.e. oracle server reads 1000000  
rows from emp table.

exists  
• SQL> select \* from dept d  
where exists (select \* from emp  
where deptno = d.deptno);

Execution - A maximum of one row  
will be read for emp table for each  
row in dept table.  
i.e. it reduces processing stat,  
then improve performance of  
the query.

Q. What to display the employee's who are getting same salary as  
'scott' salary from emp table by using correlated Query exists operator?

SQL> select \* from emp e<sub>1</sub> where exists  
(select \* from emp e<sub>2</sub> where e<sub>1</sub>.sal = e<sub>2</sub>.sal and e<sub>2</sub>.ename='scott');

Q. What to display those department from dept table does not have a  
employee in emp table by using co-related subQuery?

Select \* From dept d  
where not exists (select \* from emp where deptno=d.deptno);

O/P:

| Deptno | Dname      | Loc    |
|--------|------------|--------|
| 40     | Operations | Boston |

Q. write to display those dept from dept table does not have employee in emp table by using non-correlated subquery ?

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

O/P :-

| Dept no | Dname      | loc    |
|---------|------------|--------|
| 40      | operations | Boston |

- generally not in operator doesn't work with null values in place of this one we can also use not exists operator by using correlated sub-Query.

ex:- SQL> insert into emp (empno, deptno) values(10, null);  
SQL> select \* from emp;

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

O/P :- no row selected

SQL> select \* from dept d where not exists  
(select \* from emp where deptno = d.deptno);

O/P :-

| Deptno | Dname     | Loc    |
|--------|-----------|--------|
| 40     | operation | Boston |

\* all, any subquery special operators used in non-correlated sub-query :-

Q. write a Query to display the employees who are getting more salary than the highest paid employee's in 20<sup>th</sup> Dept from emp table ?

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

Q. write a Query to display the employee who are getting more sal than the lowest paid employee in 10<sup>th</sup> Dept from emp ?

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

- whenever resource table having large amount of data and also child query having max or min() function and also we are comparing values by using relational operator then those types of query degrades performance of the application.

To overcome this problem for improvement of performance of query ANSI/ISO SQL provided subquery special operators these are

a) all

b) any

These operators are used along with relational operator in parent query where condition.

Ex -

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

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

→ in, all, any, operation used in multiple row SubQuery :-

in → It returns same values in the <sup>list</sup><sub>child query</sub>.

≤ All → It satisfies all values in the list.

≥ Any → It satisfies any values in the list

in → = any

Q. What to display the emp who are getting more salary than the all salary of the "CLERK" from emp by using subquery special operator?

select \* from emp where sal > all (select sal from emp where

Note :-

whenever we're using subquery special operator "all" internally server uses logical operator 'and' and when we're using "any" internally server uses logical operator 'or'.

ex:- SQL> select \* from emp where deptno > all (10,20);

O/P: 30

SQL> select \* from emp where deptno > any (10,20);

O/P: 20  
30

### Note :-

Generally "in" operator is also same as special operator "any" whereas "not in" operator isn't same as "any" operator. ( $<>$  any)

ex:-

SQL) select \* from emp where deptno not in (10, 20);

O/P: 30

SQL) select \* from emp where deptno  $<>$  any (10, 20);

O/P: 10

20

30

### Note :-

"not in" operator is also same as " $<>$  all".

SQL) select \* from emp where deptno not in (10, 20);

O/P: 30

SQL) select \* from emp where deptno  $<>$  all (10, 20)

O/P: 30

Date - 18/ Dec/ 15  
The end.  
Subquery ??  
MERGE Statement

- oracle 9i introduced merge statement.
- merge is a dml statement which is used to transfer data from source table into target table when those two tables structure are same.
- merge statement is used in dataware housing application.
- merge statement are also called as upsert statement because in which statement we're are using insert, update statement.

[UPsert  $\rightarrow$  Update + Insert]

### Syntax

merge into target tablename using sourcetablename

on (Joining condition)

when matched then

update 'Set Targettablecol<sub>1</sub> = Sourcetablecol<sub>1</sub>, ...'

when not matched then

insert (targettablecolnames) values (sourcetablecolname)

```

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.dname = s.dname, d.loc = s.loc
 when not matched then
 insert (d.deptno, d.dname, d.loc) values (s.deptno, s.dname, s.loc);

SQL> select * from dept;

```

Note:-

Through the merge statement we can't update "on" clause columns.

- on (d.deptno = s.deptno)

update set d.deptno = s.deptno

error: can't be update on clause columns

SEVEN STAR EDUCATIONAL ACADEMY  
Mall Road, 112001, Ground Floor,  
Rani Mira Chowk, Near Satyaniketan  
Ameerpet, Hyderabad - 500 016.

2017/18 SAI BHAVAN

No. 7-1-20543, Gomti  
Nagar Towers, Near Pashan Park,  
Ameerpet, Hyderabad - 500 011

## VIEWS

19-DEC-15

- View is a database object which is used to provides authority level of security.
- generally data security point of view database Administrator creating views from the table and then those views given to the no. of user
- generally views doesnot store data that's why view is also called as virtual table (or) window of a table.
- generally views are created from base table. Based on the base table views are categorized into two types —
  - (a) Simple view [only one base table]
  - (b) Complex view or Join view [no. of base table]

### (a) Simple 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 table.

#### Syntax -

```
create or replace view viewname
as
select statement;
```

Ex- create or replace view V;  
as  
select \* from emp where deptno = 10;

SQL> select \* from V;

#### DML operation on simple view:-

In oracle we can also perform DML operation through simple view to base table based on following restrictions:-

- 1) if a simple view having group functions, group by clause, rownum, distinct, set operators, Joins then we can't perform DML operation through simple views to base table.

(2) We must include base table not null column into the view then only we are allow to perform insertion operation through simple view to base table.

ex:-①

```
create or replace view v1
as
select * from emp where deptno = 10;
SQL> select * from v1;
SQL> insert into v1 (empno, ename, deptno) values (1, 'Murali', 30);
O/P: 1 row created.
SQL> select * from emp;
```

ex-②

create or replace view v2

as

select \* from sal,deptno from emp where deptno = 10;

SQL> select \* from v2;

SQL> insert into v2 (ename, sal, deptno) values ('abc', 2000, 30);

errors cannot insert NULL into empno;

- In all databases whenever we are creating a view then automatically view definition (select statement) are permanently stored in database.
- In oracle if we want to view these definitions then we are using user\_views data dictionary.

Example -

SQL> desc .user\_views

SQL> select text from user\_views where view\_name = 'v1';

O/P: Text

select "SNO", "NAME", from base

NOTE:-

Views also used for simplifying query purpose i.e. regularly used query we are putting in a view and whenever necessary select that view when a view contain functions or expression then we must create alias name for those function and expression otherwise oracle server returns an error.

SQL> create or replace view v<sub>1</sub>

as

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

error:- must name this expression with a column alias.

solution -

SQL> create or replace view v<sub>1</sub>

as

select deptno, max(sal) a from emp  
group by deptno;

SQL> select \* from v<sub>1</sub>;

| Deptno | A    |
|--------|------|
| 30     | 3350 |
| 20     | 3800 |
| 10     | 7700 |

ex:- for viewing aliasname A=?

SQL> desc user\_views;

SQL> select text from user\_views

where

view\_name = 'v<sub>1</sub>' ;

Text

Select deptno, max(sal) a from emp group by deptno

Note:-

In oracle when a view having rownum then also we must create aliasname for rownum.

```
SQL> create or replace view V1
 as
 Select rownum, ename From emp;
```

Error: must name this expression with a column alias.

```
SQL> create or replace view V1
 as
 Select rownum so, ename from emp;
```

```
SQL> select * From V1;
```

| so | ename  |
|----|--------|
| 1  | Smith  |
| 2  | Millar |
| 3  | Jone   |

```
SQL> desc user_views;
```

```
SQL> select text from user_views
 where view_name = 'V1';
```

TEXT

```
Select rownum so, ename From emp;
```

Complex views -

Complex view is a view which is created from multiple base table.

```
SQL> create or replace view V5
 as
 select ename, Sal, dname, loc
 from emp,dept
 where emp.deptno=dept.deptno;
```

```
SQL> select * from V5;
```

Generally we can't perform DML operation through complex view to base table.

```
SQL> update V5 set ename='abc' where ename='SMITH';
1 row updated.
```

SQL> update VS set dname = 'xyz' where dname = 'SALES';

Error: Cannot modify a column.

In oracle when we are trying to perform DML operations through complex view to base table then some other table columns are effected and some others are not effected. If we want to view effected, unaffected columns then we are using user\_updatable\_columns data dictionary.

ex- SQL> desc user\_updatable\_columns;

SQL> select COLUMN\_NAME, UPDATABLE from user\_updatable\_columns  
where table\_name = 'VS';

O/P:-

| COLUMN_NAME | UPDATABLE |
|-------------|-----------|
| ENAME       | YES       |
| SAL         | YES       |
| DNAME       | NO        |
| LOC         | NO        |

- Generally in oracle also we can't perform DML operations through complex view to base table. To overcome this problem oracle 8.0 introduced instead of triggers in PL/SQL.
- whenever we are creating instead of triggers in this complex view then only we are allowed to perform DML operations through complex view to base table. By default instead of trigger are row level triggers and instead of trigger are created on views.

### \* \* \* \* \* triggers - (PL/SQL) -

trigger is also same as stored procedure and also it will automatically invoked whenever DML operations performed on a table.

- All DB system having two types of triggers -
  - Statement level triggers.
  - Row level triggers.

- In statement level triggers trigger body is executed only once per DML statement.
- whereas in row-level triggers trigger body is executed for each row for DML statement.

### Trigger Syntax

```

 { create or replace Trigger TriggerName
Trigger specification } before/alter insert/delete/update on tablename
 [for each row] ↗ for row level trigger

 { begin
trigger body _____
 =
 =
 -
 end;
 }
```

### Difference between statement level, row level triggers :-

SQL> select create table test (col1 date);

#### Statement level trigger

```

SQL> create or replace trigger th1
 after update on emp
 begin
 insert into test
 values (sysdate);
 end;
 /
```

#### Testing :-

SQL> update emp set sal = sal + 100  
where deptno = 10;

O/P: 3 rows updated

SQL> select \* from test;

O/P: col1  
21-Dec-15

SQL> delete from test;

SQL> select \* from test;

    No row selected

SQL> drop trigger th1;

### row level trigger :-

SQL> create or replace trigger th2

after update on emp

for each row

begin

insert into test

values (sysdate);

end;

/

K. Rajasekhar  
8374344542

### Testing :-

SQL> update emp set sal = sal + 100  
where deptno = 10;

O/P :- 3 row updated

SQL> select \* from test;

O/P :- col1

21-DEC-15 ✓

21-DEC-15 ✓

21-DEC-15 ✓

### \* row-level triggers :-

→ In row-level trigger, trigger body is executed for each row for DML statement that's why we are using for each row clause in trigger specifications and also DML transactional values internally storing two rollback segment Qualifiers.

These are -

(1) :old

(2) :New

These buffers are used in either in triggers specification or the trigger body. When we are using the buffer in buffer body then we must use column(:) in front of the buffer name.

Syntax -

: Old • Columnname

Syntax

: New • Columnname

|      | Insert | Update | Delete |
|------|--------|--------|--------|
| :new | ✓      | ✓      | X      |
| :old | X      | ✓      | ✓      |

(Q) write a PL/SQL low level trigger on emp table whenever user deleting data on emp table those deleted records are automatically stored in another table?

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

SQL> select \* from backup;

No rows selected

SQL> desc backup;

SQL> create or replace trigger tk1  
after delete on emp

for each row

begin

insert into backup

values (:old.empno, :old.ename, :old.job, :old.mgr, :old.hiredate,  
:old.sal, :old.comm, :old.deptno);

end;

/

data is not updated if only  
structure is update

Testing :-

SQL> delete from emp where sal > 2000;

10 rows deleted.

SQL> select \* from backup; SQL> drop trigger tk1;

\* Instead of trigger:-

→ generally we can't perform DML operations through complex view to base table.

→ To overcome this problem oracle 8.0 introduced instead of triggers in pl/sql. By default instead of triggers are row-level triggers and also instead of triggers are created a views

Syntax -

Trigger specification: { create or Replace Trigger Triggername  
instead of insert/update/delete on View name.

For each row

Trigger body { begin

-----

-----

-----

end;

**MANOJ ENTERPRISES**  
Plot No: 40, Gayathri Nagar,  
Ameerpet, Hyderabad.  
Cell: 8125378496

SQL> select \* from V5;  
 SQL> desc user\_updatable\_columns;  
 SQL> select column\_name, updatable from user\_updatable\_columns where table\_name = 'V5';

| O/P :- | COLUMN NAME | UPDATABLE |
|--------|-------------|-----------|
|        | ENAME       | Yes       |
|        | SAL         | Yes       |
|        | DNAME       | No        |
|        | LOC         | No        |

Solution (By using instead of triggers)

SQL> create or replace trigger TV1

instead of update on V5  
for each row.



begin

update dept set

dname = :new.dname where dname = :old.dname;  
update dept set loc = :new.loc where loc = :old.loc;

end;

/

Testing -

ex-1:

SQL> update V5 set dname = 'xyz'  
where dname = 'SALES';

O/P : 1 row updated

ex-2

SQL> update

SQL> desc user\_updatable\_columns;

SQL> select column\_name, updatable from user\_updatable\_columns where table\_name = 'V5';

| O/P :- | COLUMN NAME | UPDATABLE |
|--------|-------------|-----------|
|        | ENAME       | Yes       |
|        | SAL         | Yes       |
|        | DNAME       | Yes       |
|        | LOC         | Yes       |

## Materialized views -

125

- Oracle 8i introduced materialized view.
- M.view used in Data warehousing app!.
- M.views are handled by database Administration.
- Generally Views doesn't store data whereas Materialized views stores data.
- Generally materialized views are used to improved performance of the joined or aggregate Queries.
- Materialized views stores result of the query. I.e. M.views stores replication of remote database into local node.
- M.view also stores data same like table but when we are refreshing M.views it synchronize the data based on base table.

### Q) difference between views, Materialized views.

#### Views

1. View does not store data.
2. Security purpose.
3. When we are dropping base table then view can not be accessible.
4. Through the view we can perform DML operation.

#### Materialized views

1. Materialized view stores data.
2. Improve performance purpose.
3. When we are dropping base table also materialized view can be accessible.
4. we can't perform DML operation.

#### Syntax -

```
create Materialized view
Viewname
as
Select statement;
```

- In oracle before we are creating materialized view then database Administrator must give create any materialized view privilege to user.

Syntax -

grant create any materialized view to username;

ex:-  
 SQL> conn system/sud;  
 SQL> create materialized view mz1  
 as  
 select \* from emp;

Error: Insufficient privileges.

SQL> conn sys as sysdba;  
 Enter password : sys  
 SQL> grant create any materialized view to ~~sud~~ system;  
 SQL> conn system/sud;  
 SQL> create materialized view mz1  
 as  
 select \* from emp;

o/p: materialized view created

Note :-

In oracle some times views also returning insufficient privileges error. To overcome this problem database administrator must give create any view privilege to user.

Syntax -

grant create any view to username;

ex:-  
 SQL> conn system/sud;  
 SQL> create or replace view v1  
 as  
 select \* from emp;  
 Error: insufficient privileges  
 SQL> conn sys as sysdba;  
 Enter Password: sys  
 SQL> grant create any view to scott;

```
SQL> conn scott/tiger;
SQL> create or replace view v1
 as
 select * from emp;
```

Op: View created.

- one of the materialized based table must have a primary key then only create Materialized view among those table.

ex:-

```
SQL> create table test(sno number(10));
SQL> create materialized view mv1
 as
 select * from test;
```

error: table 'Test' does not contain a Primary key constraints

Note :-

In oracle 11g, 12c we can also create materialized view when base table doesn't have primary key also.

ex:-

```
SQL> create table base (sno number(10) primary key, name varchar2(10));
SQL> insert into base values(...);
SQL> select * from base;
```

| sno | Name |
|-----|------|
| 1   | a    |
| 2   | b    |
| 3   | c    |
| 4   | d    |

```
SQL> create or replace view v1
 as
 Select * from base;
```

```
SQL> create materialized view mv1
 as
 select * from base;
```

- In this case materialized view is also same as view because whenever we are creating view then automatically view definitions are permanently stored in DB.

- Whenever we are creating materialized view definitions are automatically permanently stored in database same like a view definition.
- In oracle if we want to view materialized view definitions then we are using user\_mviews data dictionary

ex:-

```
SQL> desc user_mviews;
SQL> select query from
 user_mviews where
 mview_name = 'mji';
```

#### Execution -

Whenever we are creating a view then automatically view definitions are permanently stored in Database. whenever we are requesting view each and every time Query definition are executed. That's why each and every time base table is affected that's why views doesnot improve performance of the query.

- Whenever we are creating a materialized view then oracle server automatically stores materialized view definitions in data dictionary and also oracle server automatically stores query result within materialized view.
- Whenever we are requesting materialized view by using select statement then oracle server each and every time retrieve data from materialized view. In this case oracle server doesn't execute query definition that's why in this case base tables are not affected. That's why a materialized view improves performance of the query.
- Whenever we are refreshing materialized view then only oracle server executes materialized view definition. In this case only base tables are affected.

```
SQL> select randid, sno, name from base;
SQL> select randid, sno, name from v1;
```

Here view randid are same as base randid that's why view doesnot store data therefore it is called also virtual table. i.e. through the views we can access base table data.

SQL> select rowid, sno, name from m1;

In this case materialized view rowid is different from base table rowid that's why materialized view stored data.

SQL> select \* from base;

| SNO | NAME |
|-----|------|
| 1   | a    |
| 2   | b    |
| 3   | c    |
| 4   | d    |

SQL> update base set name = upper(name);

SQL> select \* from base;

| SNO | NAME |
|-----|------|
| 1   | A    |
| 2   | B    |
| 3   | C    |
| 4   | D    |

SQL> select \* from v1;

| SNO | NAME |
|-----|------|
| 1   | A    |
| 2   | B    |
| 3   | C    |
| 4   | D    |

SQL> select \* from m1;

| SNO | NAME |
|-----|------|
| 1   | a    |
| 2   | b    |
| 3   | c    |
| 4   | d    |

- materialized view also stores data same like a table but when we are refreshing materialized view its synchronized data based on base table.

If we want to refresh materialized view then we are using refresh procedure from dbms\_mview package.

Syntax.

SQL> exec dbms\_mview.refresh('materializedviewName');

Package Name      Procedure Name.

```
SQL> exec
dbms_mview.refresh ('mj1');
```

```
SQL> select * from mj1;
```

| SNO | NAME |
|-----|------|
| 1   | A    |
| 2   | B    |
| 3   | C    |
| 4   | D    |

- oracle having two types of materialized view -

1. complete refresh materialized views

2. Fast Refresh materialized view

#### 1. Complete Refresh Materialized View :-

In oracle by default materialized view are complete refresh materialized views in these materialized views internally rowids are recreated when we are refreshing materialized view if we are not modifying data in base table also.

That's why these materialized view does not give more performance when we are ~~use~~ refreshing m.view's more no. of time.

#### Syntax :-

```
create materialized view
view name
refresh complete
as
select statement;
```

ex:-

```
SQL> select rowid, sno, name from mj1;
```

```
SQL> exec
```

```
dbms_mview.refresh ('mj1');
```

```
SQL> select rowid, sno, name from mj1;
```

[ Here Rowid are changed ]

2) Fast refresh materialized view :-

Fast refresh m.view are also called as incremental refresh M.view

This m.view performance is very high compare to complete refresh M.view because in this M.view rowid's are not changed when we are refreshing m.view no. of types also.

Syntax :-

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

Before we are creating fast refresh m.view it needs a mechanism to capture any changes made to its base table. This requirement is also called as M.view log.

That's why before creating fast refresh materialized view we must create m.view on base table by using following syntaxes

Syntax -

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

Ex:-

SQL> ~~update~~ select \* from base;

| SNO | NAME |
|-----|------|
| 1   | A    |
| 2   | B    |
| 3   | C    |
| 4   | D    |

SQL> create materialized view log on base;

SQL> create materialized view **mj2**

refresh fast

as

Select \* from base;

SQL> Select rowid, sno, name from **mj2**;

SQL> update base set name = 'pqr' where sno=2;

SQL> exec

dbms\_mview.refresh('mj2');

SQL> select rowid, sno, name from **mj2**;

[Here rowid are never changed]

on demand / on commit

- generally we are refreshing M-view in two ways -

1> Manually

2> automatically

i. Manually :-

In manual method we are refreshing M-view by using dbms\_mview package.

This method are also called as on demand method.

In oracle by default refresh method is on demand.

ii) automatically :-

• In oracle we can also refresh M-view without using dbms\_mview package ; this method is called on commit method.

Syntax -

```
create materialized view Viewname
refresh complete / refresh fast ondemand / on commit
as
select statement;
```

Ex:- select \* from base;

| SNO | NAME |
|-----|------|
| 1   | A    |
| 2   | PQR  |
| 3   | E    |
| 4   | D    |

SQL> create materialized view mj3

refresh fast on commit

as

select \* from base;

SQL> update base set name = 'zzz' where sno = 3;

SQL> select \* from base;

| SNO | NAME |
|-----|------|
| 1   | A    |
| 2   | PQR  |
| 3   | zzz  |
| 4   | D    |

SQL> select \* from mj3;

| SNO | NAME |
|-----|------|
| 1   | A    |
| 2   | PQR  |
| 3   | C    |
| 4   | D    |

SQL> commit;

SQL> select \* from m13;

| SNO | NAME |
|-----|------|
| 1   | A    |
| 2   | PQR  |
| 3   | ZZZ  |
| 4   | D    |

- generally M.view are used to improves performance of the join & aggregate query whereas views are used to provide security i.e. in all databases if you want to restrict table columns from one user to another user then only we're creating view with the required column then only those views given to the no. of users by using grant command.

### Data Control Language (DCL)

Date  
24-Dec-15

- grant (viewing permission)
- revoke (cancel permission)

Creating a user:-

SQL> conn sys as sysdba | or SQL> conn system/manager  
Enter password: sys |

Syntax: create user username identified by password;

Syntax: Grant connect, Resource to username;

(or)

DBA

SQL> conn username/password;

```
SQL> create user murali;
 identified by murali;
SQL> grant connect, resource to murali;
SQL> conn murali/murali;
SQL> Select * from emp;
 error: table or view does not exist
SQL> conn scott/tiger;
SQL> grant all on emp to murali;
SQL> conn murali/murali;
SQL> select select * from scott.emp;
SQL> create synonym holiday for scott.emp;
SQL> select * from holiday;
```

Ex(1)

## \* Privilege :-

Privilege is a right given to the another user.

privilege is also called as "permission".

Data security point of view all DB systems having two types of privileges :-

These are -

- (1) system privileges
- (2) object privileges.

### 1) System privileges :-

In all DB system privileges are given by DBA only.

- system privileges are allowed to users to create database objects and also drop database objects.

- oracle having more than 80 system privileges these are create session, create table, create any view, create procedure, create trigger, create any materialized view, ...;

Syntax -

Grant systemprivileges to username1,username2,...;

NOTE:-

oracle having create system privilege which is used to connect client (users) connect to the oracle database

Ex:- SQL> conn sys as sysdba;

Enter Password: sys

SQL> create user newyear;

identified by newyear;

SQL> conn newyear / newyear;

Error: user newyear lacks create session privilege;  
logon denied.

solution.

SQL> conn sys as sysdba;

Enter password : sys

SQL> grant create session to newyear;

SQL> conn newyear / newyear;

connected to Database.....

SQL> show user;

SQL> user is "newyear"

Ex:-

SQL> conn sys as sysdba;

Enter password: sys

SQL> grant create procedure, create trigger, create any materialized view to scott;

~~26/7/16~~

ROLE

Role is nothing but collection of either system privileges or collection of object privileges.

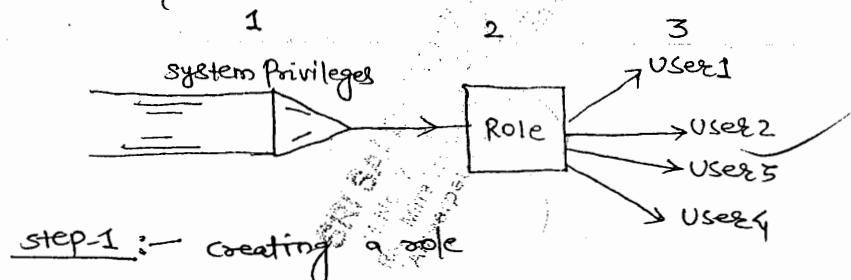
All database systems having two types of role.

1) User defined roles

2) Predefined roles.

1) User defined roles:- These roles are created by DBA only

- In multiuser environment (DB) no. of user's works on same project in this case some users requires common set of privileges in this case DBA create a role and then assigning common set of privileges to role and then only the role given to the no. of users.



Syntax - create role rolename;

step-2 :- assign system privileges to rolename;

Syntax - grant system Privileges to rolename;

Step-3 - assign role to number of users -

Syntax - grant rolename to username1, username2, ...;

ex:- SQL> conn sys as sysdba;

- Enter password : sys;

SQL> create role r1;

SQL> grant create any materialized view, create any view, create trigger to r1;

SQL> grant r1 to scott, murray, newyear;

- In oracle all system privileges related to role stored under role-sys-privs data dictionary.

ex:- desc role-sys-privs [Connect sysdba then desc]

SQL> select role, privilege from role\_sys\_Privs where role='r1';

Right side  
role-sys-privs

| ROLE | PRIVILEGE                    |
|------|------------------------------|
| R1   | CREATE TRIGGER               |
| R1   | CREATE ANY MATERIALIZED VIEW |
| R1   | CREATE ANY VIEW.             |

## 2) Predefined roles :-

Whenever we are installing oracle server then automatically three predefined roles are created -

- (a) connect -----> end user
- (b) resource -----> developer
- (c) dba -----> database administrator purpose

connect role internally having create session privilege which is used to connect to the server.

Ex:- SQL> connect sys as sysdba;  
Enter password : sys

```
SQL> desc role_sys_privs;
SQL> select role, privilege from
role_sys_privs where role
in ('CONNECT', 'RESOURCE');
SQL> select role, privileges from
role_sys_privs where role
in ('DBA');
```

SQL> conn sys as sysdba;  
Enter password : sys

SQL> create user a1 identified by a1;

SQL> conn a1/a1;

Error: user A1, lacks CREATE SESSION PRIVILEGES; logon denied.

SQL> conn sys as sysdba;  
Enter password : sys

SQL> grant connect to a1;

SQL> conn a1/a1;

SQL> create table n1 (Sno number(10));

Error: insufficient privileges.

IMP

## 2) Object Privileges :-

- object privileges are given by either database developers or database administrator.
- whenever we are using object privileges then only users allow to perform some operation of the objects.
- Oracle having insert, update, delete, select execute, read, write object privileges.

NOTE:- In oracle we can also use all keyword in place of insert, update, delete, select object privileges.

Syntax :- Grant objectPrivileges on objectname  
to username/solename/public;

SQL> conn scott/tiger

SQL> grant all on emp to murali, newyear;

SQL> grant update (ename) on emp to murali;

NOTE:- In oracle if we want to view object privileges related to user then we are using user\_tab\_priv data dictionary.

SQL> desc user\_tab\_priv;

27-7-16

## Revoke

- It is used to cancel system or object privileges from user.

Syntax -

① Revoke systemPrivileges from username, username2, ...

② Revoke objectPrivileges on objectname  
from usernames, username2, ...

generally in all databases if you want to restrict table columns from one user to another user then only we are using views i.e. we are creating a view with the required column then only that view given to the no. of users by using object privileges.

Syntax - Grant objectPrivileges on viewname  
to username1, username2, ...

```

SQL> create or replace view v1
 as
 select empno, ename, sal, deptno from emp;
SQL> grant all on v1 to muzali;
SQL> conn muzali/muzali;
SQL> select * from scott.v1;

```

#### NOTE:-

In all DB through the views we can achieve to logical data independence i.e. whenever we are adding new column into existing table it is not affected within view in the external level.

```

SQL> conn scott/tiger;
SQL> alter table emp add address varchar2(10);
SQL> select * from emp;
SQL> conn muzali/muzali;
SQL> select * from scott.v1;

```

#### Uses of views in all databases :-

1. Views provides security.
2. Simplifying complex queries.
3. We can achieve logical data independence.

#### With with check option :-

If we want to provide constraint type mechanism on views then only we are using with check option clause.

When the views having check option clause then we're not allowed to insert other than where cond' values through views to base table.

#### Syntax -

```

create or replace view viewname
as
Select * from tablename where
Condition with check option;

```

e.g. - create or replace view v2  
as  
select \* from emp where  
deptno = 10 with check option;

SQL> select \* from v2;  
 SQL> insert into  
 V2 (empno, ename, deptno) values (1, 'Gokula', 30);  
 error: view with CHECK OPTION  
 where-clause violation  
 SQL> insert into  
 V2 (empno, ename, deptno) values (4, 'Bhau', 10);  
 o/p: 1 row created  
 SQL> select \* from v2;

- with read only

- When a views having optioned with read only clause then those type of views is called read only views.
- we can't perform DML operations on read only views.

ex:- SQL> create or replace view v3  
 as  
 select \* from emp where  
 deptno = 10 with read only;  
 SQL> select \* from v3;  
 SQL> delete from v3 where deptno = 10;  
 error: cannot delete from view.

- Force View (or) Forced View :-

In oracle we can also create a view without having base table. these type of views are also called as forced view.

syntax:- create or replace  
 force view viewname  
 as  
 select \* from anyname;

ex:- SQL> create or replace view v4  
 as  
 select \* from dec31st;

Error: table or view doesnot exist.

SQL> create or replace force view V4

as

select \* from dec31st;

/

Warning: View created with compilation error.

SQL> create table dec31st (sno number(10));

SQL> alter view V4 compile;

SQL> desc V4;

- we can also drop a view by using drop view viewname

SAI SAI 30-Dec-15  
M.No. 1  
Date Mins  
Answe... 100

## SEQUENCE

30-Dec-15.

- Sequence is a database obj. which is used to generate sequence number automatically.
- generally sequences are used to generate primary key values automatically.
- generally sequences are created by Database Administrators. Once sequence are created, then no. of users simultaneously access that sequence.
- Sequence is an independent Database object

### Syntax:-

```
create sequence sequence name
start with n
increment by n
minvalue n
maxvalue n
cycle/nocycle
cache/nocache;
```

- If we want to generate sequence value or access sequence value then we are using following 2 pseudo columns. These are:—

1) currval

2) nextval

Syntax :-

Sequencename.curval

Syntax :-

Sequencename.nextval

These pseudo column are used in select, insert, update, delete statement.

- In oracle if we want to generates sequence values by using select statements then we must use dual table.

Syntax :-

Select sequencename.curval From dual;

Syntax :-

Select sequencename.nextval From dual;

Ex:- create sequence s1

start with 5

increment by 2

Maxvalue 100;

SQL> select s1.curval from dual;

error: Sequence S1.CURRVAL is not yet defined in this session.

In oracle we want to generate first sequence number then we must use nextval pseudo column; because curval pseudo column returns current value of the sequence. If sequences session already having a value.

SQL> select s1.nextval from dual;

nextval

5

SQL> select s1.nextval from dual;

nextval

7

SQL> select s1.nextval from dual;

nextval

9

:

:

- Current always returns current value of the sequence session whereas nextval returning next value of the sequence.

Ex:-

Session 1

SQL> create sequence s<sub>1</sub> ;

SQL> select s<sub>1</sub>.nextval from dual;

nextval  
1

SQL> select s<sub>1</sub>.nextval from dual;

2

SQL> select s<sub>1</sub>.nextval from dual;

3

SQL> select s<sub>1</sub>.currval from dual;

3

Session 2

SQL> select s<sub>1</sub>.nextval from dual;

4

SQL> select s<sub>2</sub>.nextval from dual;

5

SQL> select s<sub>1</sub>.currval from dual;

5

- If we want to generate primary key value automatically then we are using auto increment concept in all DB.
- If we want to implement this concept in oracle then we are using sequence, row level triggers. i.e. we are defining sequence DB object in SQL and then use that sequence in PL/SQL row level triggers.
- In oracle 12C we can also generate Primary key value without using PL/SQL row level triggers. In this case we are creating sequence in a SQL and use that sequence name.nextvalue along with default clause within a table.

Syntax :-

|                                                          |             |
|----------------------------------------------------------|-------------|
| columnname datatype (size) default sequence_name.nextval | Primary Key |
|----------------------------------------------------------|-------------|

Ex:- (oracle 12C)

SQL> create sequence s<sub>1</sub> ;

SQL> create table test (sno number(10) default  
s<sub>1</sub>.nextval primary key, name varchar2(10));

SQL> insert into test (name) values ('&name');

~~SQL>~~ Enter value for name: Murali .

SQL> /

O/P: Enter value for name: xyz

SQL> select \* from test;

| O/P:- | SNO | NAME   |
|-------|-----|--------|
|       | 1   | Murali |
|       | 2   | xyz    |
|       | 3   | :      |
|       | 4   | :      |

Prior to oracle 12C (testing in sql):-

SQL> create sequence s1;

Sequence created

SQL> create table test(sno number(10), ~~name~~ varchar2(10) primary key, name varchar2(10));

SQL> insert into test (sno, name)  
values (s1.nextval, '&name');

Enter value for name: abc

SQL> /

Enter value for name: xyz

SQL> select \* from test;

| O/P:- | SNO | NAME |
|-------|-----|------|
|       | 1   | abc  |
|       | 2   | xyz  |
|       | 3   | :    |

update

SQL> alter table test add orderno number(10);

SQL> select \* from test;

| SNO | NAME | ORDERNO |
|-----|------|---------|
| 1   | abc  | 0       |
| 2   | xyz  |         |

SQL> create sequence s2 start with 1001;

SQL> update test set orderno = s2.nextval;

SQL> select \* from test;

| SNO | NAME | ORDERNO |
|-----|------|---------|
| 1   | abc  | 1001    |
| 2   | xyz  | 1002    |

NOTE:-

In oracle we can also change sequence parameter value by using alter command but we are not allowed to change starting sequence number.

Syntax:-

```
alter sequence sequence_name
 Parameter_name new_value;
```

ex:-

```
SQL> create sequence s1;
```

start with 5

increment by 1

minvalue 3;

```
SQL> select s1.nextval from dual;
```

5

```
SQL> select s1.nextval from dual;
```

6

```
SQL> select s1.nextval from dual;
```

7

```
SQL> alter sequence s1;
```

increment by -1;

Sequence altered

```
SQL> select s1.nextval from dual;
```

6

```
SQL> select s1.nextval from dual;
```

5

```
SQL> alter sequence s1;
```

start with 4;

error: Cannot altered starting sequence number

NOTE:-

In sequences start with cannot be less than min value.

```
SQL> create sequence s1;
```

start with 3

increment by 1

minvalue 5;

Error: start with cannot be less than minvalue.

28-7-16

## SQL SEQUENCE CACHE

P.O. No. 7-1026/1, Gomti Nagar  
Amaravati, Hyderabad, Telangana  
Ameerpet, Hyderabad-500 011.

31-Dec-15

### Cache :-

- Cache is a Memory area which is used to access sequence value very fastly.
- Generally cache memory area stores set of sequence numbers which is used to improves performance of the sequence.
- Generally whenever we are creating a sequence then the sequences are generated in Hard-disk. Whenever we are requesting sequence value by using current and nextval through a tool then server process checks if requested sequence is available in cache memory area. If it is not available in cache memory area then oracle server searches requested sequences are available in Harddisk. If it is available then only oracle server fetches requested sequence value to cache memory area. Then only that value is return to client tool.
- whenever we are requesting sequence number ~~at time~~ more no. of times then this process degrades performance of the sequence to overcome this problem for improve the performance of the sequence we are putting set of sequence no. in cache memory area by using cache optional clause.  
Then only whenever we are using sequence number then server process directly fetches sequence number from cache memory area. This process improve performance of the sequence because it reduces disk i/o.

### NOTE:-

- whenever system crashes then automatically cache values are lost.
- In oracle by default cache value is 10; and also cache min value is 2.

Ex:-  
SQL> create sequence s1 ;  
SQL> select \* from  
user\_sequences ;  
SQL> drop sequence s1 ;

**MANOJ ENTERPRISES**  
Plot No: 40, Gayathri Nagar,  
Ameerpet, Hyderabad,  
Cell: 8125378496

ex2 :-

SQL> create sequence s,  
 start with 1  
 increment by 1  
 cache 1

Error:- the number of values to CACHE must be greater than 1.

- In oracle sequences information stored under user\_sequences data dictionary,

SQL> desc user\_sequences;

31-Dec-15

## LOCKS

- Locking is a mechanism which prevents unauthorized access for our resource.

- All database systems having two types of locks -

- ✓ a) Row level locks
- ✓ b) Table level locks

### a) Row level locks :-

- oracle 6.0 introduced row level locks.
- In this method we are locking set of rows by using For update clause. This clause is used in select statement only.

Syntax :-

```
Select * from tablename

where condition for update [nowait];
```

NOTE :-

whenever we are performing locks then another user query the data but they can't perform DML operations and also in all databases whenever we using commit or rollback then automatically locks are released.

ex:-

SQL> conn scott/tiger

SQL> select \* from emp  
 where deptno = 10 for update;

SQL> Commit;

[for Releasing locks]

SQL> conn murali/murali

SQL> update scott.emp  
 Set sal = sal + 100  
 where deptno = 10;

[we cannot perform DML]

01-JAN-16

### nowait:-

- nowait, is an optional clause using with along optional clause.
- whenever we're using this clause then oracle server automatically, returns controlled the current session.
- If another user not releasing locks also in this case oracle server returning an error ORA-00054: resource busy.

scott/tiger

```
SQL> select * from emp
 where deptno=10 for update
 nowait;
```

error: ORA-00054: resource busy

SQL> -

murali/murali

```
SQL> select * from scott.emp
 where deptno=10 for update;
```

### default locks :-

In all DB whenever we're using DML statement then DB server automatically establishes locks.

Ex:-

scott/tiger

```
SQL> update emp set
 sal = sal + 100 where deptno=10;
```

SQL> commit;

[For releasing locks].

scott/tiger

```
SQL> update emp set sal=sal+100
 where deptno=10;
```

[we can't perform DML because of internal locks].

### b.) table level locks :-

table level locks only handle by DBA only.

In this method we are locking a table. oracle having following

Level of locks -

1) Share lock

2) Exclusive lock.

#### 1.) Share lock -

whenever we're using this lock then another user query the data but they can't perform DML operation and also at a time no. of user's lock the resource.

Syntax:-

```
LOCK table tablename
in share mode;
```

Scott/tiger

SQL> lock table emp  
in share mode;

SQL&gt; commit;

[for Releasing lock]

murali/murali

SQL> select \* from scott.emp; ✓

SQL> lock table scott.emp  
in share mode; ✓

SQL> update scott.emp set sal = 100+sal; ✗  
[we can't perform DML]

2) Exclusive lock:-

Whenever we're using this lock then another user query the data but they can't perform DML operation and also at a time only one user lock the resource.

Syntax -

```
LOCK table tablename
in exclusive mode;
```

eg:-

scott/tiger

SQL> lock table emp  
in exclusive mode;

SQL&gt; commit;

[for Releasing lock]

murali/murali

SQL> select \* from scott.emp; ✓

SQL> lock table scott.emp  
in share mode; ✗

NOTE:-

In all Database system whenever we are using cursor locking Mechanism then DataBase servers internally uses exclusive locks.

29-7-16

## SRI SAI B.TECH CLASS

No. 7-6-39  
Rani Mira Marg  
Ameerpet, Hyderabad - 500012

# INDEX

Date - 02 Jan 16.

- Index is a Database obj which is used to retrieved data very fastly from the Database. That's why indexes are used to improvement of Query.
- Generally indexes are created on table columns and also indexes are created by Database Administrator.
- In oracle we are creating indexes in two ways -
  - a) Automatically.
  - b) Manually.

### a) Automatically :-

In oracle whenever we are creating a primary key or unique key in a table column then oracle server internally automatically creates btree indexes on those columns.

### b) Manually :-

We can also created index explicitly by using following syntax.

Syntax -

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

### NOTE :-

- whenever we are requesting data by using where clause or order by clause then only DB Server searching for indexes.
- In oracle whenever where clause having not equal to or, is null or is not null operators then oracle server doesn't search for indexes. If those columns having already indexes also.
- Oracle having two types of indexes -
  - a) Btree indexes
  - b) Bitmap indexes

### a) Btree indexes :-

In oracle by default indexes are Btree indexes.

Syntax

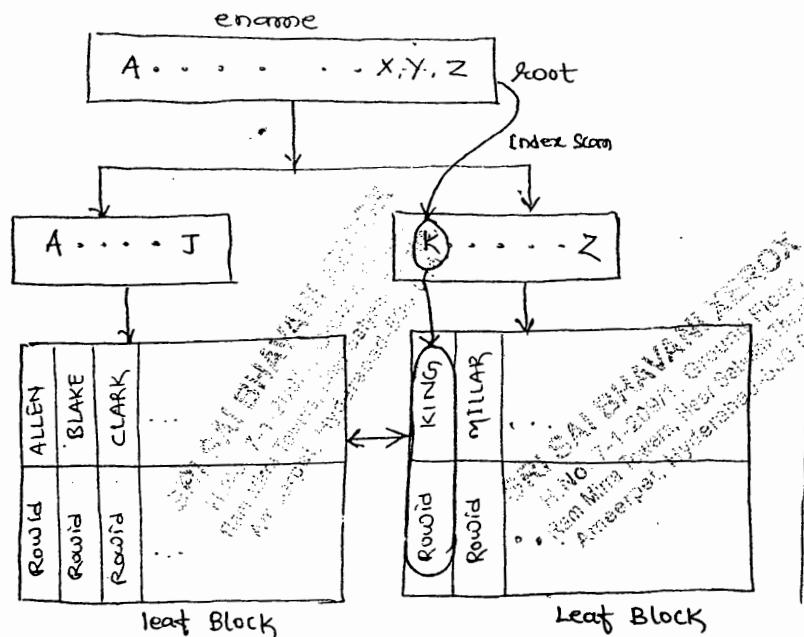
```
create index indexname on tablename(columnname);
```

- whenever we are creating Btree index then oracle server automatically creates Btree structure based on the column values in this Btree structure always leaf block stores actual data along with row-id
- whenever user requesting data by using indexed columns in where clause then oracle server automatically uses indexes scan of the B-tree str. for retrieving data very fastly from the leaf blocks.
- whenever we requesting data by using where clause. if where clause columns doesnot have any indexes then oracle server internally uses full-table scan for retrieving data from DB.

ex:-

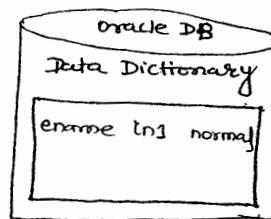
### Developer

```
SQL> Select * from emp
where ename = 'KING';
```



### DBA

```
SQL> create index in1
on emp (ename);
```



### NOTE:-

In oracle all index information stored under `user_indexes` Data Dictionary

Q:- SQL> create index

```
in1 on emp (ename);
```

```
SQL> desc user_indexes;
```

```
SQL> select index_name, index_type from user_indexes where
table_name = 'EMP'
```

| INDEX_NAME | INDEX_TYPE |
|------------|------------|
| IN1        | NORMAL     |
| PK_EMPNO   | NORMAL     |

### NOTE:-

In oracle if we want to view column names along with indexname then we are using **USER\_IND\_COLUMNS** [Database Dictionary].

Ex:-

SQL> desc user\_ind\_columns;

SQL> select column\_name, index\_name from user\_ind\_columns  
where table\_name = 'EMP';

| COLUMN_NAME | INDEX_NAME |
|-------------|------------|
| ENAME       | IN1        |
| EMPNO       | PK_EMPNO   |

### 30-7-16 • calculate Performance of the query

(or)

Testing query is searching For indexes or not:-

- Step-1:- use an explain plan for clause in front of the query.

Syntax-

SQL> explain plan for select statement;

when ever we use specifying this clause oracle server internally automatically creates plan table, which display query performance.

- Step-2:- (display plan table)

if we want to view plan table then we are using DISPLAY function dbms\_xplan package by using following syntax.

Syntax:-

SQL> select \* from table(dbms\_xplan.display());

Ex:-

without using indexes -

SQL> select \* from emp where ename = 'SCOTT';

Testing Performance -

SQL> explain plan for select \* from emp where ename = 'SCOTT';

SQL> select \* from

table(dbms\_xplan.display());

with using Indexes :-

SQL> create index idx on emp(ename);  
 SQL> select \* from emp where ename = 'SCOTT';

Testing performance -

SQL> explain plan for select \* from emp where ename = 'SCOTT';  
 SQL> select \* from table (dbms\_xplan.display());

Function based Indexes :-

Oracle 8.0 introduced function based indexes in oracle, whenever `where` clause having functions or expressions then oracle server doesn't search for indexes if those columns having already indexes also. To overcome this problem oracle introduced extension of the Btree indexes called function based indexes which used to create indexes on columns along with functions.

By default function based indexes also Btree indexes also.

Syntax -

create index indexname<sup>on</sup> tablename(functionname(columnname));  
 (or)  
 stored expression

ex:- without using function based indexes :-

SQL> select \* from emp where upper(ename) = 'SCOTT';

Testing performance -

SQL> explain plan for select \* from emp where  
 upper(ename) = 'SCOTT';

SQL> select \* from

table (dbms\_xplan.display());

with using function based indexes

~~SQL> select~~ \*

SQL> create index idx2 on emp(upper(ename));

SQL> select \* from emp where upper(ename) = 'SCOTT';

Testing performance -

SQL> explain plan for select \* from emp where  
 upper(ename) = 'SCOTT';

eg. SQL> desc user\_indexes;

SQL> select index\_name, index\_type  
from user\_indexes where  
table\_name = 'EMP';

| <u>INDEX_NAME</u> | <u>INDEX_TYPE</u>     |
|-------------------|-----------------------|
| IN2               | FUNCTION BASED NORMAL |

#### • Virtual Column :-

oracle 11g introduced virtual columns in a table which is used to store stored expression directly in a table column.

- Prior to oracle 11g we are store in stored expression indirectly by using views, function based indexes. These two methods are used by database Administrator only - whereas the oracle 11g we are storing stored expression directly in table column by using generated always ~~as~~ clause.

#### Syntax -

columnname datatype(size) generated always as (stored expression)  
[Virtual]

eg. SQL> create table test  
(a number(10), b number(10), c number(10)  
generated always as (a+b) virtual);  
  
SQL> insert into  
test(a,b) values (20,30);  
  
SQL> select \* from test;

O/P :-      a    b    c  
              20    30    50

Note -- In oracle if you want to view virtual column expressions then we are using data\_default property from user\_tab\_columns data dictionary.

ex:- SQL> desc user\_tab\_columns;  
SQL> select  
column\_name, data\_default from  
user\_tab\_columns where  
table\_name = 'TEST';

| <u>column_name</u> | <u>data-default</u> |
|--------------------|---------------------|
| C                  | "A+B"               |

- oracle have a two types of Btree indexes —

1) Non-unique indexes

1) Non-unique btree indexes

2) unique btree indexes.

- In oracle by default automatically created indexes are unique btree indexes. and manually created indexes are Non-unique btree indexes.
- generally unique btree indexes performances are very high compare to non-unique btree indexes.
- In oracle we can also create unique btree indexes explicitly by following syntax —

Syntax -

create unique index indexname on tablename(columnname);

- generally we are not allowed to create unique indexes on duplicate value columns.

ex:-

SQL> create unique index in3 on emp(ename);  
index created.

SQL> create unique index in4 on emp(job);  
error: cannot CREATE UNIQUE INDEX;  
duplicate key found

SRI SAI  
H.No. 7-102  
Ram Mira Bagh  
Ameerpet, Hyderabad  
500012

1-08-16

### Bitmap indexes:-

oracle 7.3 introduced bitmap indexes.

- generally bitmap indexes are used in data warehousing application.
- bitmap indexes are created on low cardinality columns.

cardinality of a column =  $\frac{\text{no. of distinct values}}{\text{total no. of rows}}$

① cardinality of empno =  $\frac{14}{14} = 1$  (High cardinality)

↳ btree indexes.

② cardinality of job =  $\frac{5}{14} = 0.357\ldots$  (Low cardinality)

↳ bitmap indexes

#### Syntax -

```
create bitmap index indexname on tablename(columnname);
```

- whenever we are creating bitmap index then oracle server internally automatically creates a bitmap table based on the column value in a table.

Whenever user request a data by using logical operator or equality operator in where clause then directly bits are operated within bitmap table. Then oracle server automatically converts result and bitmap into RowID by using internal bitmap function.

#### DBA

SQL> create bitmap index i5 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  |

#### Developer

SQL select \* from emp  
where job = 'CLERK';

Row ID ↗ Row# ↗ RowID ↗

High 7-12-13-14  
Row ID ↗ Row# ↗ RowID ↗  
Amarpal Ayedai

- We can also drop index by using -

```
drop index indexname;
```

NOTE:-

In all databases through the indexes we can achieve physical data independence i.e whenever we are adding an index at internal level it is not affected in structure of the table within conceptual level but performance will be affected.

## SYNONYMS

Date :- 04/Jan/16.

- Synonyms is a database object which provide security.
- Synonyms is an alias name or reference name for original object.
- generally synonyms hides another schema Username, Object Name.

Syntax -

```
create synonym synonymname for username.objectname;
```

- All database systems having two types of synonyms -

1> Public Synonym

2> Private Synonym

Note :- In all databases by default synonyms are private synonyms. in oracle if we want to create private synonyms then we are follow these syntax -

```
create synonym synonymname for username.objectname@data
baseLink;
```

- In oracle if we want to create public synonyms then we are using following syntax -

```
create public synonym synonymname for
username.objectname @ databasename;
```

- Before we are creating Public synonyms DBA first give ~~to~~ create Public Synonyms System privilege to user by using following syntax otherwise oracle server return an error insufficient privilege

Syntax -

```
grant create public synonym to username;
```

SQL> conn scott/tiger

SQL> grant all on emp to murali, a1;

SQL> murali/murali

SQL> select \* from scott.emp;

SQL> create synonym abc  
for scott.emp;

SQL> select \* from abc; ✓

SQL> create public synonym  
abc xyz for scott.emp;  
error: insufficient privileges.

SQL> conn sys.as sysdba  
Enter Password: sys

SQL> grant create public  
synonym to murali;

SQL> conn murali/murali;

SQL> create public synonym  
xyz for scott.emp; ✓

SQL> select \* from xyz; ✓

SQL> a1/a1:

SQL> select \* from scott.emp; ↵

SQL> select \* from abc; ↵

error: Table or view does  
not exists. ↵

SQL> select \* from xyz; ✓

- We can also drop synonyms by using

drop synonym synonymname;

- In oracle all synonyms information stored under user\_synonyms  
data dictionary.

SQL> desc user\_synonyms;

NOTE:-

oracle 10g provided Recyclebin synonym name for user\_recyclebin  
data dictionary.

9-8-16

## Normalization

Date  
05/Jan/16.

159

- Normalization is a scientific process which is used to decomposing a table into no. of tables: This process automatically reduces duplicate data and also automatically avoids insertion, updation, deletion problems.
- In design phase of SDLC database designers designs LOGICAL MODEL of the database in this logical model only database designers uses normalization process by using normal forms.
- In 1970 E. F. Codd written a paper "Relational Model of data for large shared data banks". In this paper only E. F. Codd introduced first-three Normal forms.

Normal Forms :-

1) 1-NF

2) 2-NF

3) 3-NF

4) BCNF

5) 4-NF

6) 5-NF

1) 1-NF (First Normal Form) :-

If a table is in 1-NF in that table data in each column should be atomic and also identifying a key record uniquely by using key.

| Item Table (Not in 1NF) |             |       |     |
|-------------------------|-------------|-------|-----|
| Item name               | color       | price | Tax |
| Marker                  | Red, green  | 20    | 0.2 |
| Pen                     | blue, black | 30    | 0.3 |

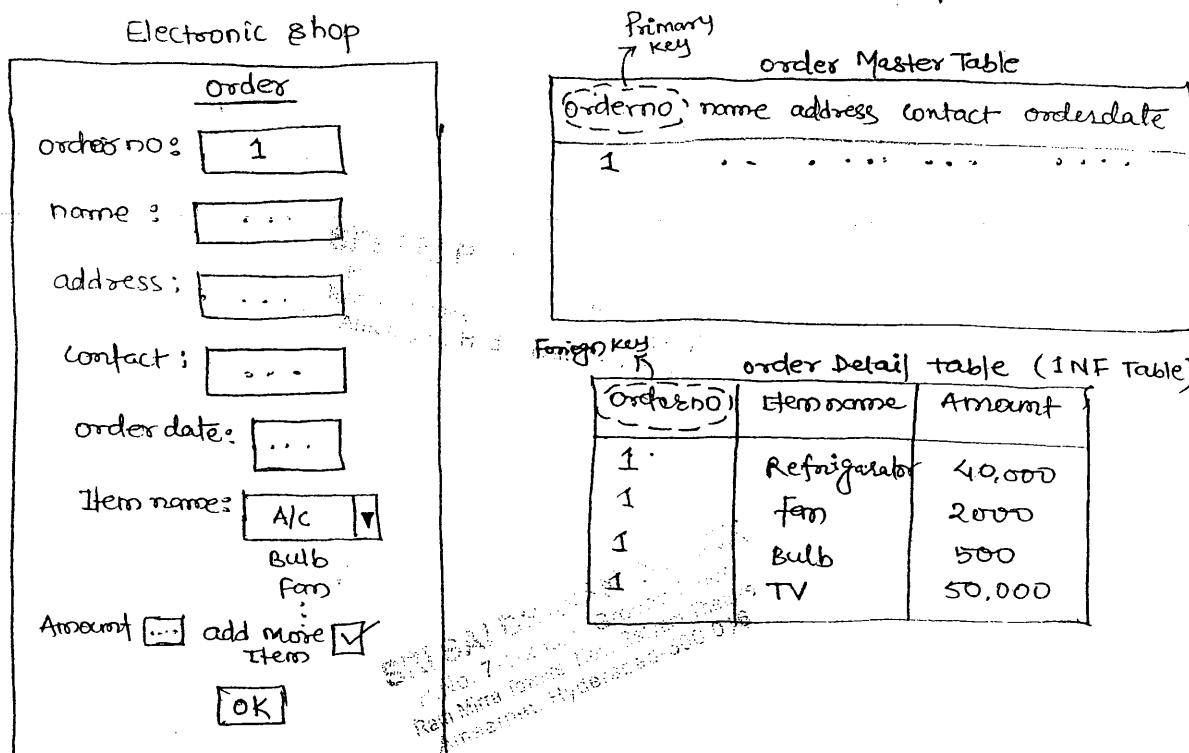
SHRISHAI BHAVAN  
H.No. 7-1-2001, 2nd floor, 10th main,  
Ram Mira Tower, 6, Mira Road (East),  
Amarpet, Hyderabad - 500062.

| Item Table (1NF) |       |       |     |
|------------------|-------|-------|-----|
| Item name        | color | price | Tax |
| Marker           | Red   | 20    | 0.2 |
| Marker           | green | 20    | 0.2 |
| Pen              | black | 30    | 0.3 |
| pen              | blue  | 30    | 0.3 |

Process :-

Identifying repeating column groups and put into separate table in more atomic form.

By default this table is called 1NF table, by default this is a child table because in this table one ~~one~~ column having duplicate data.

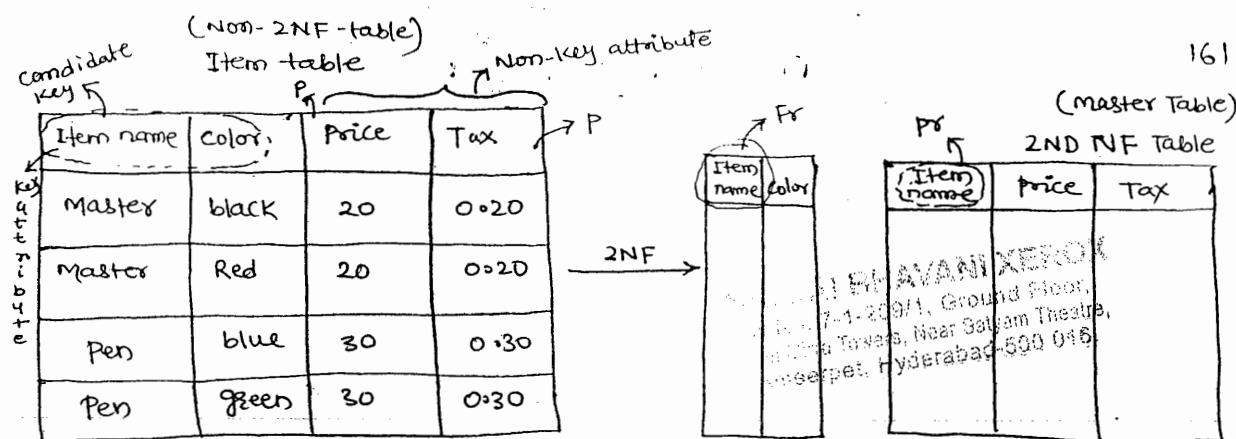


Q. 8/16 (2) **2-NF (Second Normal Form):—**

- If a table is in 1-NF and also all non-key attributes are fully functionally dependent on total candidate key.
- Always 1-NF deals with atomicity whereas 2-NF deals with relationship between key and non-key attributes.
- If a table is 1-NF and also the table having any partially non-key attribute then the table Not in 2NF.

Process:-

Identify Partial Non-key attributes which depends on partial - key attributes. Those all attributes put into specific table. This table is called 2NF table. By default this table is a Master table. In this table only all non-key attributes are fully functionally dependent on total candidate key.



Sports Activity Table

| stno | sname | activity <sub>1</sub> | cost <sub>1</sub> | activity <sub>2</sub> | cost <sub>2</sub> |
|------|-------|-----------------------|-------------------|-----------------------|-------------------|
| 101  | abc   | cricket               | \$10              | Football              | \$20              |
| 102  | xyz   | cricket               | \$10              | swimming              | \$30              |
| 103  | pqr   | swimming              | \$30              | football              | \$20              |
| 104  | zzz   | football              | \$20              |                       |                   |

| stno | sname |
|------|-------|
| 101  | abc   |
| 102  | xyz   |
| 103  | pqr   |
| 104  | zzz   |

| stno | activity <sub>1</sub> | cost <sub>1</sub> | activity <sub>2</sub> | cost <sub>2</sub> |
|------|-----------------------|-------------------|-----------------------|-------------------|
| 101  | cricket               | \$10              | football              | \$20              |
| 102  | cricket               | \$10              | swim                  | \$30              |
| 103  | swim                  | \$30              | football              | \$20              |
| 104  | Football              | \$20              |                       |                   |

| stno | sname | activity | cost |
|------|-------|----------|------|
| 101  | abc   | cricket  | \$10 |
| 102  | xyz   | Football | \$20 |
| 103  | pqr   | cricket  | \$10 |
| 104  | zzz   | swim     | \$30 |
|      |       | swim     | \$30 |
|      |       | Football | \$20 |
|      |       | Football | \$20 |

| stno | sname |
|------|-------|
| 101  | abc   |
| 102  | xyz   |
| 103  | pqr   |
| 104  | zzz   |

| stno | activity |
|------|----------|
| 101  | cricket  |
| 101  | football |
| 102  | cricket  |
| 102  | swim     |
| 103  | swim     |
| 104  | Football |

| activity | cost |
|----------|------|
| cricket  | \$10 |
| football | \$20 |
| swim     | \$30 |

2NF Table (master Table)

10/8/16

Candidate key

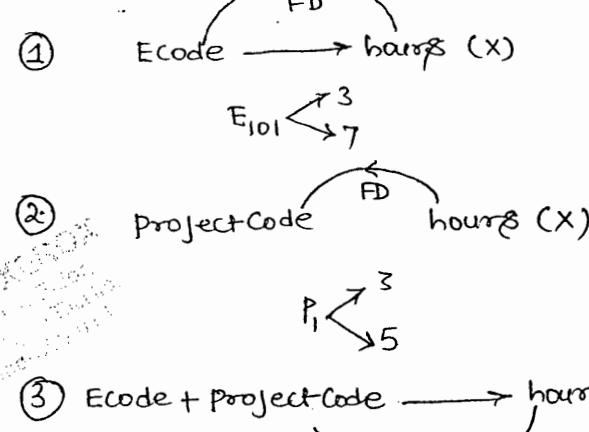
key attributes

| (Ecode) | ProjectCode    | Dept     | Depthead       | hours |
|---------|----------------|----------|----------------|-------|
| E101    | P <sub>1</sub> | System   | D <sub>1</sub> | 3     |
| E102    | P <sub>1</sub> | SALES    | D <sub>2</sub> | 5     |
| E101    | P <sub>2</sub> | System   | D <sub>1</sub> | 7     |
| E103    | P <sub>2</sub> | Research | D <sub>3</sub> | 8     |
| E104    | P <sub>3</sub> | IT       | D <sub>4</sub> | 9     |
| E102    | P <sub>3</sub> | Sales    | D <sub>2</sub> | 10    |

non-key attributes

Not in 2NF

Phase-I

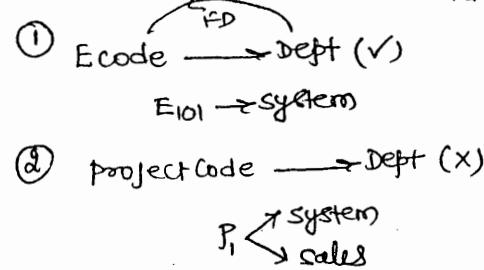


Master Table

2NF Table

| (Ecode) | Dept     | Depthead       |
|---------|----------|----------------|
| E101    | System   | D <sub>1</sub> |
| E102    | Sales    | D <sub>2</sub> |
| E103    | Research | D <sub>3</sub> |
| E104    | IT       | D <sub>4</sub> |

Phase-II



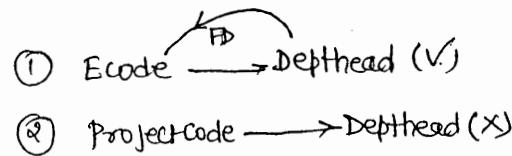
Child Table.

Fr

Child Table

| (Ecode) | ProjectCode    | hours |
|---------|----------------|-------|
| E101    | P <sub>1</sub> | 3     |
| E101    | P <sub>2</sub> | 7     |
| E102    | P <sub>1</sub> | 5     |
| E102    | P <sub>3</sub> | 10    |
| E103    | P <sub>2</sub> | 8     |
| E104    | P <sub>3</sub> | 9     |

Phase-III



- Before Normalization process above resource table having insertion, updation, deletion problems:

## ~~A~~ Insetion Problem:-

In the above resource table when we are trying to load particular department employee then we must assign project code details.

If project code is not available then we need to supply null values for the project code field, this is called insertion problem.

## \* Updation Problem - 1

- In the above resource table Code, dept, depthead attribute values are repeated. whenever an employee transferred from one department to another department then we need to modify all these three attributes values correctly otherwise inconsistency problems occurred. This is called updation problem.

## \* Deletion Problem:-

In the above resource table when we are trying to delete an employee detail then automatically department details also deleted. This is called deletion Problem.

→ All these problems are automatically avoided when we are using normalization process.

## Why Normalization Process -

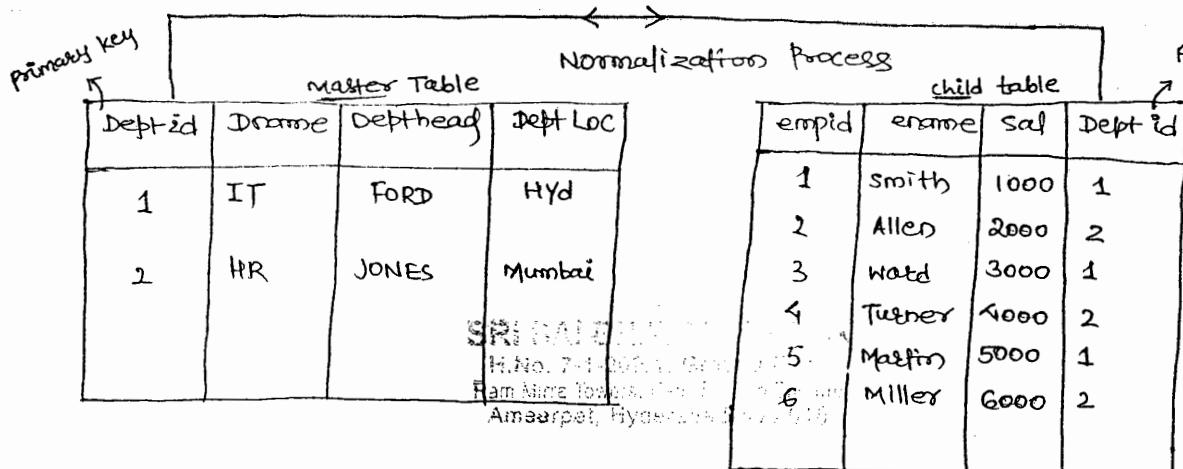
Normalization is a Scientific process which is used to reduce & duplicate data and also automatically avoids insertion, updation, deletion problems.

| emp   |        |      |       |          |         |
|-------|--------|------|-------|----------|---------|
| empid | ename  | sal  | Dname | Depthead | DeptLoc |
| 1     | Smith  | 1000 | IT    | FORD     | Hyd     |
| 2     | Allen  | 2000 | HR    | JONES    | Mumbai  |
| 3     | Ward   | 3000 | IT    | FORD     | Hyd     |
| 4     | Turner | 4000 | HR    | JONES    | Mumbai  |
| 5     | Martin | 5000 | IT    | FORD     | Hyd     |
| 6     | Miller | 6000 | HR    | JONES    | Mumbai  |

~~Full  
Wk~~

### Problems for Redundancy -

- ✓ 1> Disk space wasted
- ✓ 2> Inconsistency
- ✓ 3> DML query becomes very slow.



~~Full  
Wk~~

### If table is in 1NF -

- ✓ 1. Data in each column should be atomic, no multiple values specified separated with comma.
- ✓ 2. That table does not have Repeating column groups.
- ✓ 3. Identifying a record uniquely by using a key.

ex:-

①

#### Non-atomic ~~at~~ Column

| Dname | ename                  |
|-------|------------------------|
| IT    | smith, Allen, Ward, DK |
| HR    | Mutali                 |

#### ~~Full Wk~~ Problems for non-atomic column

It is not possible to select, insert, update, delete one employee.

②

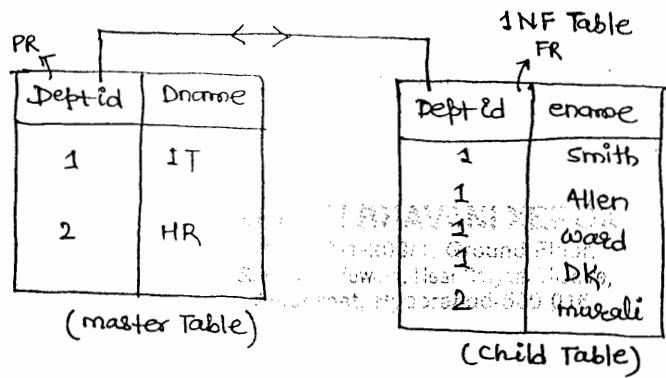
#### Repeating column groups crop

| Dname | ename <sub>1</sub> | ename <sub>2</sub> | ename <sub>3</sub> | ename <sub>4</sub> |
|-------|--------------------|--------------------|--------------------|--------------------|
| IT    | Smith              | Allen              | Ward               | DK                 |
| HR    | Mutali             |                    |                    |                    |

#### Problems for repeating column group

When we are trying to add more than 4 employees than we need to change structure of the table.

When we are trying to load less than 4 employees then disk space wasted.



### Functional Dependency (FD):-

If any given two tuples in a relation & when  $X$  (one attribute set) agrees then  $Y$  (another attribute set) cannot disagree then  $X \rightarrow Y$  is called functional dependency.

$$X \rightarrow Y$$

Here  $Y$  is functionally dependent on  $X$   
(or)

$X$  functionally determines  $Y$ .

### 3> 3-NF (Third Normal Form) :-

If a table is in 2NF and all non-key attributes are only dependent on total candidate key.

If a table is in 2NF and also any non-key attributes which depends on another non-key attributes then that table not in 3NF.

#### Process

Identify Non-key attributes which depends on another non-key attributes put into separate table. By default this is a 3NF table and also it is a Master table.

In this table only all non-key attributes are only dependent on Candidate key.

Not in 2NF

| 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 |

1NF table

2NF

2NF

| Item Name                              | Color |
|----------------------------------------|-------|
| BRIGHAM HANNAH                         |       |
| NO. 7-6-2007/1, Ground Floor,          |       |
| Ram Murti Flores, Near Satyam Theatre, |       |
| Ameerpet, Hyderabad-500 016            |       |

2NF (master table)

Not in 3NF

3NF

| Item name | Color | Item name | Price  | Price | Tax |
|-----------|-------|-----------|--------|-------|-----|
| SP1       | Red   | SP2       | Yellow | PR    |     |
| SP3       | Blue  | SP4       | Green  | PR    |     |
| SP5       | Black | SP6       | White  | PR    |     |
| SP7       | Grey  | SP8       | Pink   | PR    |     |

(master Table)  
3NF

Student table

eg -

Candidate Key

| Strname | Course id | Grade | Grade Value |
|---------|-----------|-------|-------------|
| abc     | CS111     | A     | 4.000       |
| xyz     | CS111     | B     | 3.000       |
| pqr     | CS222     | C     | 2.000       |
| zzz     | CS222     | A     | 4.000       |

Not in 3NF

3NF →

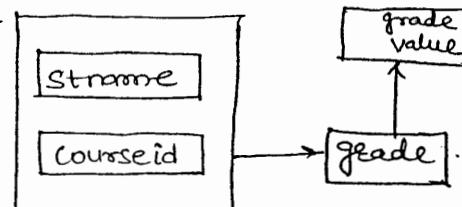
| Strname | Course id | Grade |
|---------|-----------|-------|
| abc     | CS111     | A     |
| xyz     | CS111     | B     |
| pqr     | CS222     | C     |
| zzz     | CS222     | A     |

(master Table)

| Grade | Grade Value |
|-------|-------------|
| A     | 4.000       |
| B     | 3.000       |
| C     | 2.000       |

3NF Table

Logical Diagram -



eg:-

Order Form

order NO

order date

cust no

Address

qty

discount

Phone no

Item name

item no

Amount

if add another item

OK

Cust. Enquiry Form

cust no

cust name

address

Phone no

OK

| order Master Table |            |           |
|--------------------|------------|-----------|
| order no.          | order date | cust. no. |
|                    |            |           |

order Master Table

| cust no | cust name | add | phone no |
|---------|-----------|-----|----------|
|         |           |     |          |

(3 NF Table)

1NF → remove repeating groups

2NF → remove partial attributes.

3NF → remove non-key attributes which are not dependent on key attribute  
 (or)

remove non-key attributes which are dependent on another non-key attribute.

12/8/16

GRISSAI BHAVI

H.NO. 7-1-366/1  
Sam Mira Towers, Marathahalli  
Ambedkar Layout, Bangalore - 560037

## CLUSTER

Date  
09/Jan/16.

- Cluster is a database object which contains groups of tables together and also shares some data block.
- In all databases cluster database object is used to improves performance of the joins that's why cluster database obj is created by database Administrator only.
- Cluster tables must have a common column name, this common column is also called as cluster key.
- Generally clusters is created at time of table creation.
- In all databases whenever we're submitting inner join or outer join then database servers internally checks from clause tables are available in cluster or not? if those tables are available on cluster then database servers very fastly retrieve data from the cluster table.

### Creating clusters in oracle -

Step 1:- creating a cluster ✓

Step 2:- create an index on cluster. ✓

Step 3:- create cluster table. ✓

#### 1) creating a cluster :-

Create a cluster based on a common column name, this common column name is also called as key.

Syntax -

create cluster clustername (common columnname datatype (size));

~~Step 2:-~~

#### 2) create an index on cluster:-

Syntax -

create index indexname on cluster clustername;

#### 3) creating cluster table:-

Syntax - create table tablename (common columnname datatype (size),  
col1 datatype (size), ...)

cluster clustername (common columnname);

```

SQL> create cluster emp_dept (deptno number (10));
SQL> create index abc on cluster emp_dept;
SQL> create table emp1 (empno number (10), ename varchar2 (10), sal
 number(10), deptno number (10)) cluster emp_dept (deptno);
SQL> create table dept1 (deptno number (10), dname varchar2 (10),
 loc varchar2(10)) cluster emp_dept (deptno);
SQL> desc emp1;
SQL> desc dept1;

```

NOTE:-

- In oracle cluster tables having same rowid.
- we cannot drop cluster, if cluster having tables, to overcome this problem oracle 8.0 introduced including tables clause along with drop cluster clustername which is used to drop cluster with tables.

## Syntax-

drop cluster clustername including tables;

ex:- SQL> drop cluster emp\_dept;

error: cluster not empty

SQL> drop cluster emp\_dept including tables;

O/P:- cluster dropped

- In oracle all information of cluster stored under user\_clusters data dictionary

ex- SQL> desc user\_clusters;

Super Key, Candidate Key :-Super key :-

A column or a combination of columns which uniquely identifying a record in a table is called a Super key.

Candidate key

A minimal super key which uniquely identifying a record in a table is called candidate key, (or) A Super key which is a subset of another super key then those super keys are not a candidate key.

emp

| empno | ename  | skillid | skill      | voterid        |
|-------|--------|---------|------------|----------------|
| 1     | Murali | 1       | oracle     | v <sub>1</sub> |
| 1     | Murali | 2       | sybase     | v <sub>1</sub> |
| 1     | Murali | 3       | teradata   | 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     | pqr    |         |            | v <sub>4</sub> |
| 5     | zzz    | 6       | sql server | v <sub>5</sub> |
| 5     | zzz    | 4       | DB2        | v <sub>5</sub> |

not a super key

- empno + ename
- empno + voterid
- ename + voterid

super key

- ① empno + skill
- ② empno + ename + skill
- ③ empno + voterid + skill
- ④ ename + skill
- ⑤ ename + skill + voterid
- ⑥ voterid + skill
- ⑦ empno + ename + voterid + skill

Candidate Key

- ① empno + skill
- ② ename + skill
- ③ voterid + skill

### BCNF :-

If a table is in BCNF, in that table every determinant is a candidate key.

whenever table have a multiple composite candidate keys and also those candidate keys are overlapped, and also one candidate key non-key attributes which depends on another candidate key non-key attributes then only we're using BCNF process.

#### Process -

Identify one composite candidate key non-key attributes which depends on another composite candidate key non-key attributes put into separate table. This table is called BCNF table, in this table only every determinant is candidate key. This BCNF table doesn't have non-key attributes.

- generally 2NF and 3NF deals with relationship between key to non-key attributes, non-key to nonkey attributes. whereas BCNF deals with relationship between non-key attributes within multiple composite candidate keys itself.

NOTE:-

When a table have a multiple composite candidate key then deletion problem occurred, to overcome this problem Database designers uses BCNF process.

Table

| Prof. Code     | Dept      | HOD | hours |
|----------------|-----------|-----|-------|
| P <sub>1</sub> | Physics   | abc | 3     |
| P <sub>1</sub> | Computer  | xyz | 2     |
| P <sub>2</sub> | Chemistry | pqr | 5     |
| P <sub>2</sub> | Physics   | abc | 4     |
| P <sub>3</sub> | Botany    | zzz | 7     |

non-key attribute

① Prof. Code + Dept → hours  
② Prof. Code + HOD → hours

Candidate key

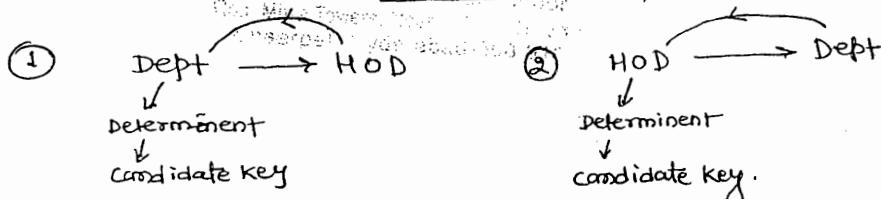
① Prof. Code + Dept  
② Prof. Code + HOD

BCNF process

bcnf table

| Dept   | HOD |
|--------|-----|
| Phy    | abc |
| Che    | pqr |
| Comp   | xyz |
| Botany | zzz |

| Prof. code     | Dept   | hours |
|----------------|--------|-------|
| P <sub>1</sub> | Phy    | 3     |
| P <sub>1</sub> | Comp   | 2     |
| P <sub>2</sub> | Che    | 5     |
| P <sub>2</sub> | Phy    | 4     |
| P <sub>3</sub> | Botany | 7     |

Multivalued Dependency: (MVD) ( $\rightarrow\rightarrow$ )

- one column data Match with multiple values in another column within a same table is called ~~multiple~~ multi valued dependency (MVD).

Ex:-

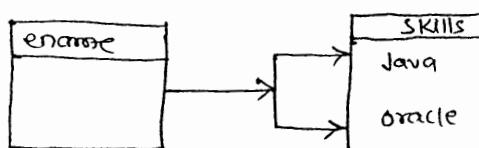
ename  $\rightarrow\rightarrow$  SKILLS

Fig: multivalued dependency.

## 4NF :-

If a table is in fourth Normal Form then that table doesn't contain more than one independent Multivalued attributes.

- When a table have a more than two attributes and also identifying a record uniquely by using the combination of all attributes and also one attribute set of values which depends on another attribute set of values and also some attribute set of values which depends on another attribute set of values and also some attributes are not logically related then only we are allowed to use 4NF process.

### Process -

Identify independent multivalued attributes put into separate table these tables are called 4NF tables.

These table doesn't contain more than one independent multi-valued attribute.

Whenever a table having more than one independent multivalued attribute then the table having more duplicate data for reducing this duplicate data, Database designer uses 4NF process.

Date  
18/Jan/16.

### Assumption :-

Each employee having multiple projects.

and also each employee having multiple skills.

| Candidate Key | empid | Project id   | Skills     |             |
|---------------|-------|--------------|------------|-------------|
|               |       |              | Skills     | Skills      |
|               | 1111  | Railway res. | oracle     |             |
|               | 1111  | Railways res | JSP        |             |
|               | 1111  | Railway res  | ASP.net    |             |
|               | 1111  | Lib. mgmt    | DB2        |             |
|               | 1111  | Lib. mgmt    | sql server | SAI BHAVANI |

H.NO. 7-1-203/1, Ground floor,  
Ananda Towers, Near Seetharama  
Aundh, Hyderabad - 500003

4NF

| empid | Project id  |
|-------|-------------|
| 1111  | Railway res |
| 1111  | Lib. mgmt   |

| empid | Skills     |
|-------|------------|
| 1111  | oracle     |
| 1111  | JSP        |
| 1111  | ASP.net    |
| 1111  | DB2        |
| 1111  | sql server |

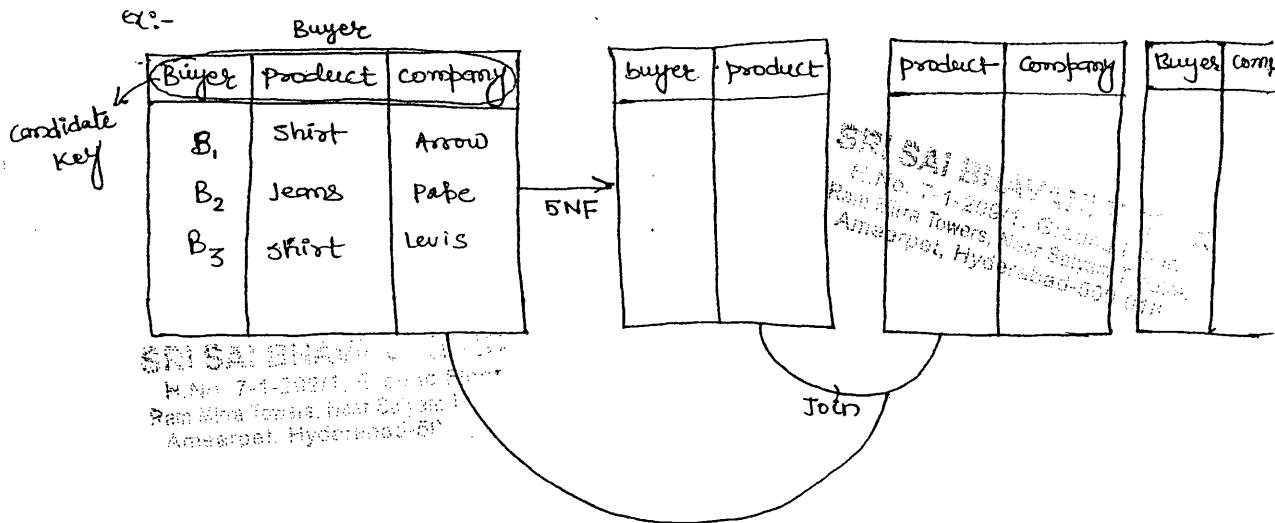
- Before 4NF process in the above resource table whenever empid got a new Project then we need to supply null values for the skills attribute. whenever employee got the new skills then we need to supply null values for the Project id attributes. These null value are automatically avoided when we are using 4NF process and also one of the 4NF table automatically reduces duplicate data.

because 4NF process a table doesn't store more than one 173 independent attributes value.

- If a table having multivalued attribute and also we can't decompose into multiple table then the table is called 5NF table.
- If possible we can also decompose no. of tables into and also we must test when we are joining process table those resultant records are available in resource table that's why till Normal form is also called as projection-joined normal form.

NOTE:-

In 4NF resource table some attributes are not logically related whereas 5NF resource table all attributes are logically related.



## Set Operators

- Set operators are used to retrieve data from single or multiple tables, these operators are also called as vertical joins.  
These operators are -

- 1) Union
- 2) Union all
- 3) Intersect
- 4) Minus

*SRI SAI INSTITUTE OF TECHNOLOGY  
H.No. 1-1-666, 1st Floor, 7-1-203/1, 10th Main Road, 5th Block, Amravati, Hyderabad-500 016.*

- 1) union - It returns unique values and also sorting those values.
- 2) union all - unique + duplicate values
- 3) Intersect - It returns common values.
- 4) Minus - It returns values from first query those values are not in second query.

Ex:-

① SQL> select job from emp where deptno = 10;

union

select job from emp where deptno = 20;

JOB

|           |
|-----------|
| ANALYST   |
| CLERK     |
| MANAGER   |
| PRESIDENT |

SQL> select deptno from emp union  
select dname from dept;

error: expression must have some data-type as corresponding expression.

② SQL> select <sup>dname</sup> job from <sup>dept</sup> emp where deptno = 10;

union

select <sup>dname</sup> job from emp where deptno = 20;

JOB

|              |
|--------------|
| <u>DNAME</u> |
| ACCOUNTING   |
| ALLEN        |
| :            |
| :            |
| :            |

NOTE:- whenever we using set operator always corresponding expression must belongs to same datatype and also set operator returns first query columns name or alias names as column headings.

In oracle we can also retrieve data from multiple queries by specifying set operators when corresponding expression not belongs to also. In this case we are using appropriate type conversion function.

Solution -

```
sql> select deptno to_char(null) "DEPTNAMES" from emp
 union
 select to_number(null), dname from dept;
```

| O/P :- | DEPTNO | DEPTNAMES  |
|--------|--------|------------|
|        | 10     |            |
|        | 20     |            |
|        | 30     |            |
|        |        | ACCOUNTING |
|        |        | OPERATIONS |
|        |        | RESEARCH   |
|        |        | SALES      |

3. <sup>Q16</sup> Conversions :-

- converting one data type into another data type is called conversions.
- oracle having two types of conversions -
  - (a) Implicit (or) automatic conversions.
  - (b) Explicit conversions.

(a) Implicit conversions -

In oracle when an expression having string, string representing pure number automatically converted character type into number type.

ex:-    sql> select sal + '100' from emp;

O/p :    sal +100              ↑ pure number.  
           4300  
           3200  
           1600  
           :

NOTE :- In oracle whenever we're passing number into character function then oracle server automatically converts number type into character type.

sql> select length(22222) from dual;

O/p : 5

- In oracle whenever we are passing date string into predefined date() functions then oracle server automatically converts date string into date type, but here passed parameter must be default date format.

Ex:- SQL> select last\_day ('15-aug-05') from dual;  
 O/P :- 31-AUG-05

Implicit Conversion Table

| From                     | To                    | Assignment | Evaluation of expression |
|--------------------------|-----------------------|------------|--------------------------|
| varchar2<br>(or)<br>char | number                | yes        | Yes                      |
| varchar2 (or)<br>char    | Date                  | yes        | Yes                      |
| number                   | varchar2 (or)<br>char | yes        | No                       |
| Date                     | varchar2 (or)<br>char | yes        | No                       |

(b) Explicit Conversion:-

In oracle also we can convert one datatype into another data type explicitly by using following explicit function.

These are -

- 1) decode()
- 2) case statement.
- 3) to\_number()
- 4) to\_char()
- 5) to\_date()

1) decode() :-

decode() is a conversion function which is used to decoding the values.

decode conversion function is same as if-then-elsif construct in PL/SQL.

Syntax -

decode (column name, value1, stmt1, value2, stmt2, stmts);

- decode conversion function internally uses equality operators.

ex:- SQL> select

deptno, decode(deptno, 10, 'ten', 20, 'twenty', 'others') from emp;

O/P:-

| deptno | decode(deptno) |
|--------|----------------|
| 10     | ten            |
| 20     | twenty         |
| 30     | others         |
| 10     | ten            |
| :      | :              |

ex:- SQL> select decode(1, 2, 3, 4, 5, 6, 7) from dual;

[not matched 1 with any pair]

O/P: decode(1, --- 7)

null

SQL> select decode(1, 2, 3, 4, 5, 1, 7) from dual;

[match 1 with 7 pair value]

O/P: decode(1, --- 1, 7)

7

SQL> select decode(1, 2, 3, 1, 5, 1, 7) from dual;

[Match 1 with 5 & 7 pair value but first pair value will be return].

SQL> select decode(1, 2, 3, 4, 5, 6, 7, 8) from dual;

O/P: decode(1, --- 8)

[in matching 1 with any pair not occurred then print unpaired value i.e. else part i.e. 8]

Q.) update employee comm in emp table based on following cond:-

(a) if job = 'CLERK' then update

comm  $\rightarrow$  10% of sal

(b) If job = 'SALESMAN' then update

comm  $\rightarrow$  20% of sal

else

comm  $\rightarrow$  30% of sal

- SQL> update emp set

comm = decode (job, 'CLERK', sal \* 0.1, 'SALESMAN', sal \* 0.2,  
sal \* 0.3);

SQL> select \* from emp;

NOTE:-

In oracle if we want to implement pivot reports then we must use decode conversion function.  
In this case we are using decode conversion function within group by clause i.e. this function is used in aggregate function within group by clause. In all databases rows converted into columns is called pivoting.

- In pivot report we are displaying aggregate function values within tabular form.

ex:- SQL> 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;

O/P:-

| JOB       | deptno10 | deptno20 | deptno30 |
|-----------|----------|----------|----------|
| CLERK     | 4800     | 8500     | 3850     |
| SALESMAN  |          | 11450    |          |
| PRESIDENT | 8800     |          |          |
| MANAGER   | 3800     | 3975     | 3550     |
| ANALYST   |          | 8000     |          |

- SQL> select dname, sum (decode (job, 'CLERK', 1,0)) "CLERKS",  
sum (decode (job, 'SALESMAN', 1,0)) "SALESMANS",  
sum (decode (job, 'ANALYST', 1,0)) "analysts".  
from emp e, dept d  
where e.deptno = d.deptno  
group by dname;

| output: | DNAME      | CLERKS | SALESMANS | ANALYSTS |
|---------|------------|--------|-----------|----------|
|         | Accounting | 1      | 0         | 0        |
|         | Research   | 2      | 0         | 2        |
|         | Sales      | 1      | 4         | 0        |

• \*\*\*  
**Pivot()**

oracle 11g introduced pivot function which is used to convert rows as columns and also display aggregate functional value in tabular form. Pivot function performance is very high compare to decode conversion function.

Syntax :-

```
Select * from
(Select col1, col2, col3, ... from tablename)
Pivot (aggregatefunctionname () for columnname in (values, values, ...))
```

ex:- SQL> Select \* from
(Select job, deptno, sal from emp)
Pivot (sum(sal) for deptno in (10 as deptno10, 20 as deptno20,
30 as deptno30));

SQL> select \* from
(Select job, deptno from emp)
pivot (count(\*) for deptno in (10 as deptno10, 20 as deptno20,
30 as deptno30));

O/P:

| JOB       | deptno10 | deptno20 | deptno30 |
|-----------|----------|----------|----------|
| CLERK     | 1        | 2        | 1        |
| SALESMAN  | 0        | 0        | 4        |
| PRESIDENT | 1        | 0        | 0        |
| MANAGER   | 1        | 1        | 1        |
| ANALYST   | 0        | 2        | 0        |

**SRI SAI BHAVANI HERCOS**  
H.No. 7-1-209/1, Ground Floor,  
Ram Mira Towers, Near Begumpet Bus Stand  
Ameerpet, Hyderabad - 500012

5-8-11

## 2) case statement :-

- case statement also used to decoding the values.
- oracle 8.0 introduced case statement whereas oracle 8 i introduced case-conditional statement.
- case-conditional statement are also called as searched case statement.
- Case statement performance is very high compare to decode conversion function.

### NOTE:-

Decode conversion function internally uses equality operators whereas in case-statement we can also use all sql operators explicitly.

### Method-I case-statement

Syntax -

```
case columnname
when value then stmt;
when value2 then stmt2
...
else stmts end;
```

e.g:- SQL> select ename, sal, deptno, case deptno  
when 10 then 'ten'  
when 20 then 'twenty'  
else 'others' end from emp;

O/p:

| deptno | case(deptno) |
|--------|--------------|
| 10     | ten          |
| 20     | twenty       |
| 30     | others       |

### method-2 case-conditional statement (8 i) -

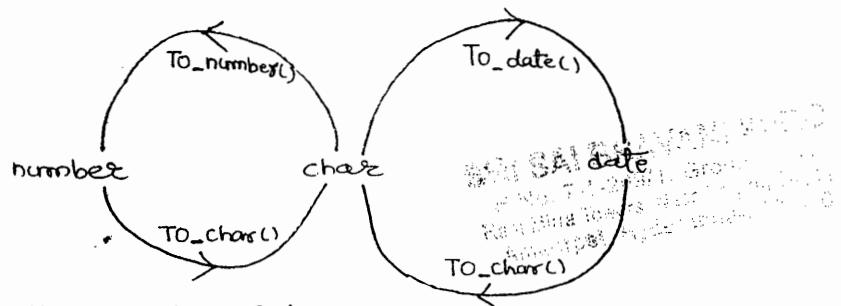
Syntax -

```
case
when column condition1 then stmt1
when column condition2 then stmt2
...
else stmts end
```

eg:- SQL> Select ename, sal, case  
 when sal < 1000 then 'low salary'  
 when sal between 1000 and 3000 then 'medium salary'  
 when sal in (3500, 3800, 4000) then 'special salary'  
 else 'others salary' end from emp;

O/P :-

| ename  | sal  | case           |
|--------|------|----------------|
| Shan   | 2000 | medium salary  |
| miller | 3500 | special salary |
| Allen  | 500  | low salary     |
| Blake  | 1000 | medium salary  |
| :      | :    | :              |



#### \* Converting number into words:-

- In oracle to\_char having Jsp format which takes Julian date and then Spell-out.

ex:- SQL> select to\_char (sysdate, 'JSP') from dual;

O/P :- two million four hundred fifty seven thousand four hundred eight

- If we want to view Julian-date then we are using format 'J' within to\_char function.

eg:- SQL> select to\_char (sysdate, 'J') from dual;

O/P: 2457408

#### Julian date

In oracle Julian date is a number of days since Jan 1, 4712 BC. Julian date always displays as number. These Julian dates are used in oracle to convert a no's into words by using to\_char, to\_date functions.

If we want to convert into any no to words then first we must convert given no's into Julian date. If we want to convert any no. into Julian date then we must use format 'J' with to\_Date function then only that Julian Date converted into word by using to\_Char JSP format.

eg:- SQL> select  
to\_char(TO\_DATE(1234,'j'), 'jsp') from dual;  
o/p:- one thousand two hundred thirty four.

- Q. WAP which is used to convert user entered number into word by using TO\_CHAR, TO\_DATE() function.

SQL> select  
to\_char(TO\_DATE(&no,'j'), 'jsp') from dual;  
o/p: Enter value for no: 786  
seven hundred eighty six.

### 3) to\_number()

Converting string representing a numeric value with format into numeric without format then only we're allowed to use TO\_NUMBER() function.

① eg:- SQL> select '\$ 56.8' +3 from dual;

ERROR

SQL> select to\_number('\$ 56.8') +3 from dual;  
ERROR

- whenever we are using to\_number also use a second parameter as some as first parameter format by using predefined format elements.

SQL> select to\_number('\$ 56.8', '\$99.9') +3 from dual;  
o/p: 59.8

② ex- SQL> select to\_number('a 56.8') +3 from dual  
error

SQL> select to\_number('a 56.8', 'a gg.g') +3 from dual  
error: invalid number format model.

#### 4) to\_char() -

`to_char` is a overloading function i.e. this function is used to convert number type into character type and also used to convert date type into date string.

#### Converting number type into character type -

| Syntax                                  |
|-----------------------------------------|
| <code>to_char (number, 'format')</code> |

#### Format elements -

- (1) G → group separator
- (2) d → decimal indicator
- (3) \$ → dollar sign
- (4) 0 → leading zero
- \*(5) L → local currency
- (6) , → group separator
- (7) . → decimal indicator
- \*(8) g → representing a number.

eg:- select `to_char (3456789, '99g 99g 999d.99')` from dual;  
o/p:- 34,56,789.00

SQL> select `to_char (345, '$999')` from dual;  
o/p: \$345

SQL> select `to_char (345, '0999')` from dual;  
o/p: 0345

#### Local currency (L) :-

SQL> select `to_char (345, 'L999')` from dual;  
o/p: \$345

NOTE:- In oracle by default local currency is \$, if we want to display other than \$ currency then we must use `nls_currency` clause in third Parameter of the `to_char ()` function. This Parameter we are must specify within single quotes.

syntax - `nls_currency = format`

↑ currency format which we want.

ex:- SQL> select `to_char (345, 'L999', 'nls_currency = RS')` from dual;  
o/p: RS345

eg:- SQL> select ename,  
 to\_char(sal,'9g999g99g999d99','nls\_currency=RS') from emp;

O/P: 

| <u>ename</u> | <u>to_char()</u> |
|--------------|------------------|
| SMITH        | RS 800.00        |
| ALLEN        | RS 2,500.00      |
| :            | :                |

SQL> select ename, nvl(to\_char(mge), 'no manager') from emp;

O/P: 

| <u>ename</u> | <u>mge</u> |
|--------------|------------|
| Smith        | 7612       |
| Allen        | 7832       |
| :            | :          |
| King         | NO manager |
| :            | :          |

SQL> select to\_char(sysdate, 'DD/MM/YY') from dual;

O/P: 21/01/16

### 5) to\_date()

It is used to convert date string into date type.

eg:- SQL> select to\_date('19-DEC-15')+7 from dual;

O/P: 26-DEC-15

SQL> select to\_date('19-~~DEC~~-15', 'DD-MM-YY')+7 from dual;

O/P: 26-12-15

## NULL VALUE FUNCTION

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

2) NVL(exp<sub>1</sub>, exp<sub>2</sub>, exp<sub>3</sub>) :-

oracle 9i introduced nvl2 function.

If exp<sub>1</sub> is null then it returns exp<sub>3</sub> otherwise it returns exp<sub>2</sub>.

3) nullif() :-

oracle 9i introduced nullif() function, this function accepts two parameters.

|                                              |
|----------------------------------------------|
| syntax -                                     |
| nullif(exp <sub>1</sub> , exp <sub>2</sub> ) |

Here, if  $\text{exp}_1 = \text{exp}_2$  then it returns null, otherwise it returns expression1. 185

e.g. SQL> select nullif (10,10) from dual;  
null

SQL> select nullif (10,20) from dual;  
10

Q. WAP to display the employee's who are getting < 2000 sal from emp table by using nullif() function returning null value in place of More than 2000 salary.

SQL> select ename, nullif (sal, greatest (2000,sal)) from emp;

| O/P:- | ename | sal  |
|-------|-------|------|
|       | SMITH | 800  |
|       | ALLEN |      |
|       | WARD  | 1000 |
|       | FORD  |      |
|       | KING  | 1000 |

### Coalesce () :-

oracle 9i introduced coalesce () Function.

coalesce is an ANSI function. This function accepts no. of expressions.

Syntax-

coalesce (exp<sub>1</sub>, exp<sub>2</sub>, exp<sub>3</sub>, ...)

This function returns first normal value from the given expression.

SQL> select coalesce (null, null, 20, null, 30) from dual;

O/P: 20

### \* Difference between NVL(), coalesce() :-

- NVL is an oracle function whereas coalesce is an ANSI function and also coalesce performance is very high as compare to NVL function.
- NVL function internally uses implicit conversions i.e. NVL function returning a value if the exp<sub>1</sub>, exp<sub>2</sub> is not belong to same datatype also, it exp<sub>2</sub> automatically converted into exp<sub>1</sub> whereas in coalesce function exp<sub>1</sub>, exp<sub>2</sub> must belongs to same datatype.

Ex:- SQL> select nvl ('a', sysdate) from dual;

O/P: 9

SQL> Select coalesce ('a', sysdate) from dual;

error: inconsistent datatypes: expected CHAR got DATE.

## Hierarchical Queries

22/3/16

- In all relational databases we can also store hierarchical data.  
If we want to store hierarchical data then relational table must have minimum three columns. In these three columns, two columns must belongs to same datatype and also having logical relation.
- In oracle if we want to retrieve hierarchical data then we are using following clauses these are -
  1. level
  2. start with
  3. connect by

1) level :- level is a pseudo column which automatically assigns numbers to each level in a tree structure.

Syntax - level

2) start with :-

start with clause specifies searching condition within tree structure.

Syntax : start with condition

3) connect by :-

using connect by clause we must specify relationship between parent, child columns by using prior operator.

Syntax : connect by prior parentcolumnname = child Column name

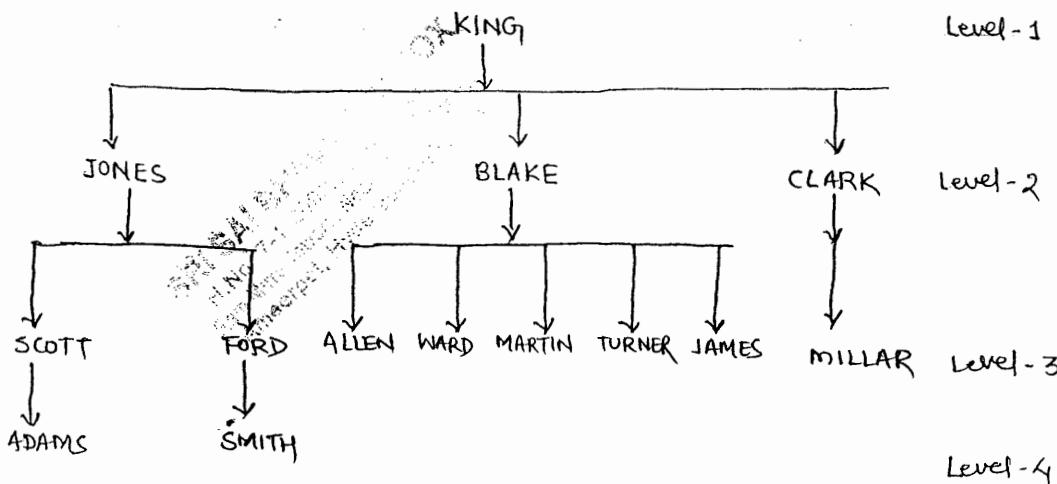
Syntax of hierarchical Queries :-

```
select level, columnname
from tablename
where condition
start with condition
connect by prior
Parentcolumnname = child columnname;
```

SHRI SAI BHAVANI MURUGAN  
H.no. 7-1-209/1, Ground floor  
Ram Mitra Towers, Near City Center  
Amarpet, Hyderabad - 500003

Q. Write a hierarchical query to display employee who are working in another employee based on root node onwards from emp table? 187

SQL> select level, ename from emp  
start with mgr is null  
connect by prior empno = mgr;



NOTE:-

oracle 9i introduced `sys_connect_by_Path()` which returning Path of the hierarchy within tree structure.  
This function accepts two parameters.

Syntax:-

`sys_connect_by_Path(columnname, 'delimitername')`

eg- SQL> select level, sys\_connect\_by\_Path(ename, '→') .  
from emp  
start with mgr is null  
connect by prior empno = mgr;

Q. write a H.Q. to display the employee's who are working under 'Blake' from emp table?

SQL> select level, sys\_connect\_by\_Path(ename, '\') from emp  
start with ename = 'BLAKE'  
connect by prior empno = mgr;

| O/P:- | Level | ename        |
|-------|-------|--------------|
|       | 1     | BLAKE        |
|       | 2     | BLAKE\ALLEN  |
|       | 2     | BLAKE\WARD   |
|       | 2     | BLAKE\MARTIN |
|       | 2     | BLAKE\TURNER |
|       | 2     | BLAKE\JAMES  |

SQL> select level, sys\_connect\_by\_path(ename, '→') from emp  
 Start with ename = 'MILLER'  
 Connect by / empno = prior mge ;

| O/P: | LEVEL | ENAME                   |
|------|-------|-------------------------|
|      | 1     | → MILLER                |
|      | 2     | → MILLER → CLARK        |
|      | 3     | → MILLER → CLARK → KING |

#### → Prior

Prior is an unary operator using along with connect by clause. whenever we are using prior operator in front of child column (empno) then oracle server uses Top-Bottom search within tree structure.

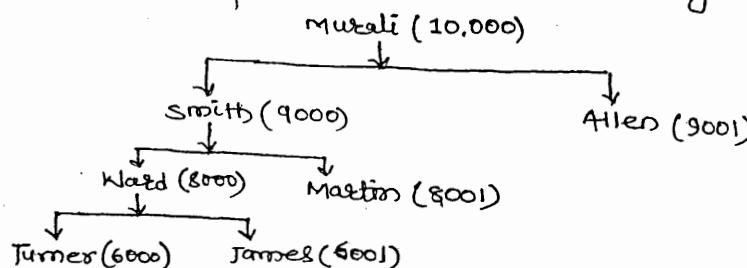
whereas whenever we are using prior operator in front of the parent column (mge) then oracle server uses Bottom-Top search within tree structure.

#### Execution:-

whenever we are submitting H.Q. then oracle server searching data based on startwith clause within relational table.

i.e. It will take a value from the prior operator column and then search that value is available within another hierarchical column in a relational table. If those values are available then corresponding data will be retrieved from third column.

Q. Create a relational table based on following tree structure



SQL> create table test (ename varchar2(10), empno number(10), mgr number(10));  
 SQL> insert into test values ('turner', 6000, 8000);  
 SQL> insert into test values ('james', 6001, 8000);  
 SQL> insert into test values ('ward', 8000, 9000);  
 SQL> insert into test values ('martin', 8001, 9000);  
 SQL> insert into test values ('smith', 9000, 10000);  
 SQL> insert into test values ('allen', 9001, 10000);  
 SQL> commit;

SQL> select \* from test;

189

| <u>ENAME</u> | <u>EMPNO</u> | <u>MGR</u> |
|--------------|--------------|------------|
| turner       | 6000         | 8000       |
| James        | 6001         | 8000       |
| ward         | 8000         | 9000       |
| Martin       | 8001         | 9000       |
| Smith        | 9000         | 10000      |
| Allen        | 9001         | 10000      |
| Murali       | 10000        |            |

NOTE:-

We can also use order by clause within Hierarchical queries but after order by clause change in structure of hierarchical within tree structure to overcome this problem oracle uses sibling by clause along with order by clause in this case oracle server does not change structure of the hierarchical and also data in each and every level within hierarchical.

Syntax: order siblings by columnname

e.g.- select level, ename from test  
start with mgr is null connect  
by prior empno = mgr order  
sibling by ename  
/

O/P: 1 murali  
2 allen  
2 Smith  
3 martin  
3 ward  
4 James  
4 turner

### Transaction Control Language (TCL) 23-Jan-16

#### Transactions:

A logical unit of work b/w two points is called transaction.

oracle having two transaction comments -

- (a) commit
- (b) Savepoint

1. Commit :-

It is a command which is used to save the transaction permanently into database.

2. Savepoint :-

It is a logical mark in between the transaction

Syntax :- Savepoint savepointname;

rollback - This command is used to undo the transaction from the memory

Syntax rollback

Rollback to particular savepoint -

Syntax - rollback to savepointname;

ex:-  
SQL> insert into emp (empno) values(1);  
SQL> update emp set sal = sal + 500 where ename = 'SMITH';  
SQL> savepoint s1;  
SQL> insert into emp (empno) values(2);  
SQL> update emp set sal = sal + 100 where ename = 'ALLEN';  
SQL> savepoint s2;  
SQL> insert into emp (empno) values(3);  
SQL> rollback to s1;  
SQL> commit;  
SQL> select \* from emp;

NOTE:- In all database before we are using rollback transaction data at least one time committed with the database otherwise rollback doesn't undo the transaction.

eg:- SQL> create table test (Sno number(10));  
SQL> insert into test value (50);  
SQL> select \* from test;

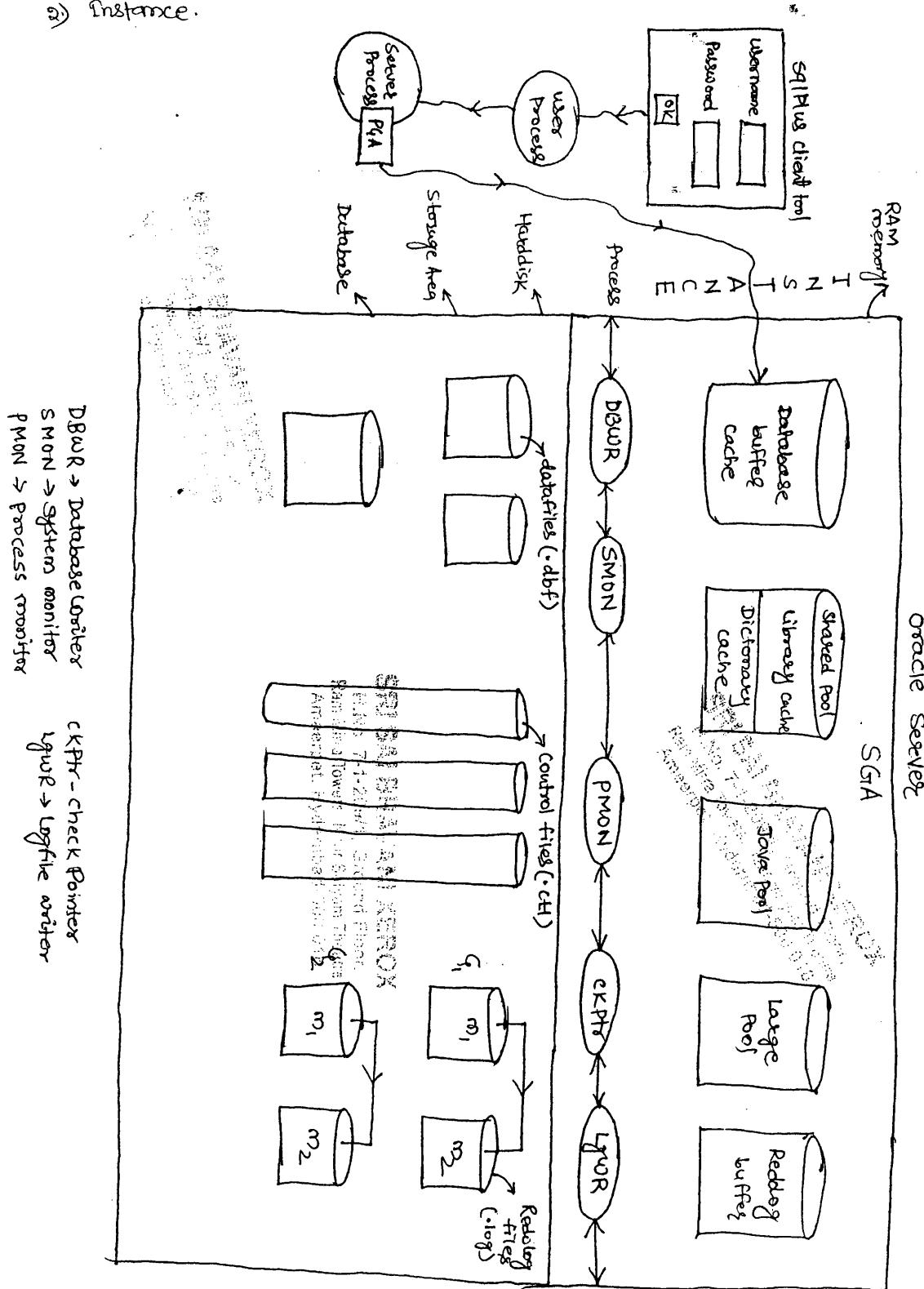
O/P: 50

SQL> delete from test;  
SQL> select \* from test;  
SQL> rollback;  
SQL> select \* from test;

O/P: no rows selected

- Oracle server architecture mainly consist of two parts -

- 1) Storage area
- 2) Instance.



## Storage Area -

when we are installing oracle server then automatically three files are created physically within hard-disk, these files are also called as Storage Area.

oracle database are mainly consist of two types of structure -

- 1) Logical Structure
- 2) Physical Structure.

### 1) Logical structure -

A structure which is not visible in o.s. is called logical structure.

Logical structure having Database like tables, views, synonyms, indexes etc.

### 2) Physical structure -

A structure which is visible in operating systems, is called physical structure. Physical structure is handled by DBA only. Oracle physical structure have a three types of files.

- a) data files (.dbf)
- b) control files (.ctl)
- c) redo log files (.log)

#### a) data files -

whenever we are creating database obj-logically then those objects are physically stored in data files.

In oracle all datafiles information stored under dba\_data\_files data dictionary.

If we want to path of the data files then we used -

eg:-  
SQL> conn sys as sysdba;  
Enter password: sys  
SQL> desc dba\_data\_files;

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

#### b) control files:-

Control files control all other files within storage area, these files extension is .ctl.

control files also stores databases informations physically. Control files used by database administrator in backup and recovery process.

All control files information stored under V\$Controlfile data dictionary.

(c) Redolog files -

Redolog files stores committed transactions from Redolog buffer. These files extension is .log.

These files are also used by DBA in Backup and recovery process.

In Oracle Redolog files information stored under V\$Logfile data-dictionary.

Instance -

In Oracle whenever user connect to the Oracle Server through the tool then Oracle Server automatically creates ~~an~~ memory area; this memory area called Instance.

Instance has two parts -

(a) SGA

(b) Background process

(a) SGA -

SGA is also called as System Global Area. These SGA memory are having set of buffers. These are -

- (1) Database buffer cache
- (2) Shared pool
- (3) Java Pool
- (4) Large Pool
- (5) Redolog buffer.

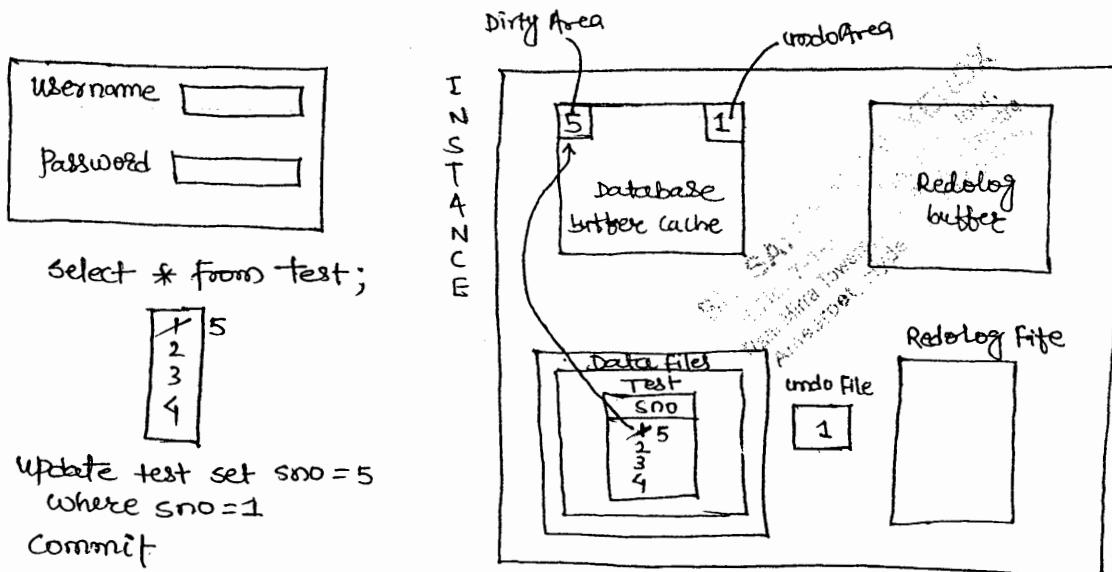
- Whenever we are submitting SQL/PLSQL code then the code is automatically stored in Library cache within Shared Pool.

Library cache always reduces parsing (syntax checkings). Library cache also stores cache value depends by sequence database objects.

- Shared pool also having dictionary cache which checks given user name, Passwords for available in appropriate data dictionary. Dictionary cache also executes TCJ related obj.

- Whenever we're requesting a table information by using select statement then server process checks requested table is available in database buffer cache. If it is not available then dbwr process checks requested table in data file within storage area. If it is available then copy of the table is transferred into database buffer cache. Then server process fetches requested table from database buffer cache into database client.

Java Pool executes Java related object whereas Large Pool perform large no. of operation and also redo log buffer stores new information for the transaction.



In oracle whenever performing DML transaction new values for the transaction stored in Dirty buffer and the old value for the transaction stored in undo area within DB buffer cache.

Whenever user using commit then automatically new value for the transaction stored in datafiles and also old value for the transaction stored in undo file within storage area. This undo file is used by oracle server when we are using flashback query.

### Flashback query.

- oracle 9i introduced flashback query.
- flashback query are handle by DB Administrator only flashback queries along allows content of the table to be retrieve with reference to specific point of time by using as of clause that is flashback queries retrieves clause that is flashback queries retrieves accidental data after committing the transaction also.
- Flashback Queries internally uses undo file that is flashback queries retrieve old data before committing the transaction.  
oracle provide two method for flashback queries-
  - method 1: using timestamp
  - method 2: using scn number.

Timestamp:- oracle 8i introduced timestamp data type.

195

timestamp datatype stores date and time upto 6 fraction of seconds.

Syntax:- columnname timestamp

If we want to view this function or see then we must insert data by using systime stamp function.

ex:- SQL> create table k1 (col1 timestamp);  
SQL> desc k1

| column name | Type         |
|-------------|--------------|
| col1        | Timestamp(6) |

SQL> insert into k1 values (sysdate);  
SQL> select \* from k1;

O/P: col1

26-Jan-16 02.52.28.00000 PM

SQL> insert into k1 values (systimestamp);  
SQL> select \* from k1;

col1

26-Jan-16 02.53.34.609000 PM

method 1: Using Timestamp

Syntax: select \* from tablename as of timestamp(Particular Point of time);

NOTE:- If you want to calculate Particular part of time then oracle 8i introduced interval function; through this function we can add or subtract days, hours, minutes, seconds.

whenever we are using interval function we are specifying number within single quotes.

ex:- SQL> create table test (sno number(10));  
SQL> insert into test values ('...');  
SQL> Commit;

SQL> select \* from test;

O/P: sno

1

2

3

4

SQL> delete from test;

O/P: 4 row deleted

SQL> Commit;

SQL> select \* from test;

O/P: No row selected

## flashback Query -

SQL select \* from test as of timestamp (sysdate - interval '1' minute);

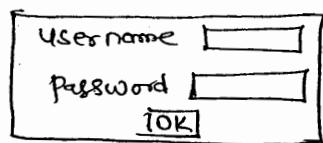
sno

1  
2  
3  
4

Redo - whenever we are performing transaction new value for the transaction is also stored in redo log buffer.

whenever user using commit or  $\gamma_3$  fill of redo log buffer then redo log buffer data automatically stored in redo files within storage area.

These redo log files used by DB Administrator if backup/recovery process.



① select \* from test;

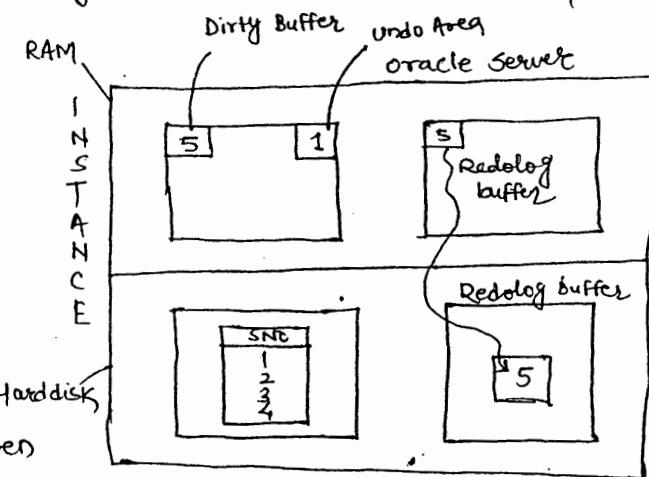
sno

|   |   |
|---|---|
| 1 | 5 |
| 2 |   |
| 3 |   |
| 4 |   |

② update test set sno=5 where sno=1;

③ commit

④  $\gamma_3$  fill of Redolog buffer



## Background Process -

oracle instance internally having 5 background process these are-

- ① DBWR (Database writer)
- ② LGWR (Log writer)
- ③ CKPT (Check pointer)
- ④ SMON (System monitor)
- ⑤ PMON (Process monitor)

① DBWR :- DB writer process reaches data from DB data buffer and store in DB file.

② LGWR :- LGWR process reaches data from redo log buffer into redo log file.

SRI SAI BHAVI, 10th Floor  
H.No. 7-4-278/1, Sec-17, Hyderabad  
Ram Niwas Towers, Mehdipatnam, Telangana  
Ameerpet, Hyderabad - 500081

⑥ CKptr :- whenever we are performing transaction or loading DB objects then check pointer process automatically generates an unique identification number for the transaction within DB files, control files, Redolog files this number also called as system changed number (SCN) using this number also we can write flashback query. 197

Method 2: using SCN number:-

Syntax: select \* from tablename as of scn 'Scnnumber';

In oracle if we want to view system change number then we are using current\_SCN properties from V\$database data dictionary.

```
SQL> conn sys as sysdba;
SQL> create table e1 (sno number(10));
SQL> desc v$database;
SQL> select current_SCN from v$database;
```

O/P: Current\_SCN

7642493

```
SQL> insert into e1 values(10);
```

```
SQL> select * from e1;
```

O/P: 10

```
SQL> commit;
```

```
SQL> select count(*) from e1 as of scn 7641493;
```

O/P: Count(\*)  
0

SP1641, SPOTLIGHT  
R.No. 7, 8th Main, Ground Floor,  
C.V.Raman Nagar, Malleswaram,  
Bangalore - 560034. Ph: 080 2222 0000.

#### 4) SMON (System monitor):-

smon process are also called as Instance recovery process because whenever problems occurs in background process then smon process identified the problem, and immediately recovery that problem.

#### 5) Pmon (Process monitor):-

pmon process automatically deallocate user processes if users not disconnected to the server properly.

PGA - PGA is also called as private global Area. This memory area is available to server process.

PGA memory Area uniquely identifies each client connect to the oracle server; because PGA memory area internally stores clients information. In oracle pl/sql collections also executed within PGA memory area.

### NESTED TABLE

- Table within another table is also called as Nested table.
- Nested table is the userdefined type which is used to store no. of data items into single unit.
- In oracle user-defined data-types are created by using "type" keyword.
- In oracle generally we are storing no. of data items in a single unit within index by table, but these tables are not allowed to store permanently into oracle database.

To overcome this problem oracle 8.0 introduced extension of the index by table called Nested table, which is used to store permanently into oracle database by using sql.

#### Creating Nested table:-

Step-1: Create an object type

Step 2: Create an nested table type

Step 3: Create a table.

#### ① Create an object type -

Before we are creating nested table type then we must create object type; Object type is a user-defined type, which is used to store different data types into single-unit. It is also same as structures in C-Language.

Syntax-

create or replace type typename as object (attribute, datatype(size),  
 ...; ...);

② Create a nested table type -

create a nested table type by using object type through following  
Syntax-

create or replace type typename as table of objectname;

③ Create a table -

create an relational table by using nested table type through  
 following syntax.

Syntax-

create table tablename (col<sub>1</sub> datatype(size), col<sub>2</sub> datatype(size),  
 ... col<sub>n</sub> nested tabletype)

nested table coln store as anyname;

e.g- SQL) create or replace type Y<sub>1</sub> as object

(bookid number(10), bookname varchar2(10), Price number(10));  
 /

SQL) create or replace type aaa as table of Y<sub>1</sub>;

/

SQL) create table student (stno.number(10), sname varchar2(10),  
 col<sub>3</sub> aaa)

nested table col<sub>3</sub> store as KKK;

SQL) desc student;

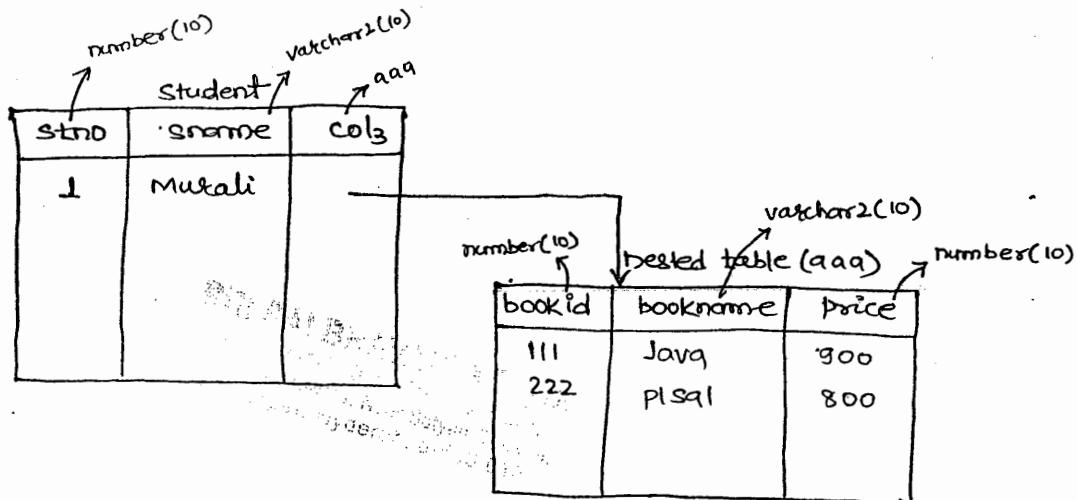
O/P:-

| Name  | Type         |
|-------|--------------|
| STNO  | Number(10)   |
| SNAME | varchar2(10) |
| COL3  | AAA          |

- If we want to store data into nested table type then we must use  
 constructor, here constructor name is also same as type name.

SQL) insert into student values (1, 'murali', AAA(Y<sub>1</sub>(111, 'java', 900),  
 Y<sub>1</sub>(222, 'PLSQL', 800)));

SQL) select \* from student;



- we can also view, update, delete data in nested table by using table operator. i.e. we can specify table operator along with subquery in place of tablename within select, update, delete statements.

Syntax -

table (select nestedtable.colname from relational.coltablename)

ex:-

`SQL> select * from table (select col3 from student);`

O/P:-

| Bookid | Bookname | Price |
|--------|----------|-------|
| 111    | Java     | 900   |
| 222    | plsql    | 800   |

`SQL> update table (select col3 from student)`

`set price = 1000 where Bookid = 111;`

`SQL> select * from table (select col3 from student);`

O/P:-

| Bookid | Bookname | Price |
|--------|----------|-------|
| 111    | Java     | 1000  |
| 222    | plsql    | 800   |

## Partitions Table

28-Jan-16. 201

Partition tables are used to improve performance of the appln in backup and recovery process.

partition tables are created by Database Administrator in very large databases.

partition tables are used in Data warehousing Applications; partitions tables are created based on a key-column. This column is also called as partition key.

In oracle If we want to view particular partitions then we use the following syntax -

```
select * from tablename
partition (partitionname, partitionname2, ...);
```

oracle having three types of partitions -

- 1) Range partitions
- 2) List partitions
- 3) Hash partitions

### 1. Range partitions :-

In this method partitions are created based on range of value

Syntax :-

```
create table tablename (col1 datatype(size),)
partition by range (keycolumnname)
(partition Partitionname1 values less than (value),
partition Partitionname values less than (maxvalue));
```

e.g:- SQL> create table test (sno number(10), name varchar2(10),  
partition by range(sal) sal number(10))  
(partition P<sub>1</sub> values less than (1000), partition P<sub>2</sub> values less than (2000),  
partition P<sub>3</sub> values less than (3000),  
partition P<sub>4</sub> values less than (maxvalues));

SQL> insert into test values (.....);

SQL> select \* from test partition (P<sub>4</sub>);

O/P:

| SNO | Name   | Sal  |
|-----|--------|------|
| 5   | Ramya  | 9000 |
| 6   | Ramesh | 8000 |

for (int i=1; i<10; i++)  
    System.out.println("value of " + i);

### a) List Partitions -

oracle 9i introduced List partition, using list Partition we can also create partition based on character Datatype columns.

In this method partitions are created based on List of values.

Syntax - create table tablename (colname1 datatype(size), ...)  
partition by list (keycolname)  
partition partitionname1 values (val1, val2, ...),  
...  
partition partitionname values (default));

eg:- SQL> create table test (sno number(10), name varchar2(10)).  
partition by list (name)

partition P1 values ('india', 'Pakistan'),

partition P2 values ('us', 'UK', 'Canada'),

partition others values (default));

SQL> insert into test values(...);

SQL> select \* from test partition (others);

| SNO | NAME  |
|-----|-------|
| 1   | aus   |
| 2   | japan |
| 3   | china |

### 3) Hash partition :-

Hash partitions are automatically created by oracle server based on Hash algorithm, in this case we are specifying no. of partitions explicitly.

Syntax-

create table tablename (col1 datatype(size), ...)  
partition by hash (keycolname)  
partitions anynumber;

eg:- SQL> create table test (sno number(10), sal number(10))

partition by hash (sal)

partitions 5;

In oracle all partitions information stored under user\_tab\_partitions data dictionary.

SQL> desc user\_tab\_partitions;

SQL> select PARTITION\_NAME from user\_tab\_partitions  
where table\_name = 'TEST';

MANOJ ENTERPRISES  
Plot No: 40, Gayathri Nagar,  
Ameerpet, Hyderabad.  
Cell: 8125378496

# **ORACLE 12C**

## **PL/SQL**

### **MURALI**

**MANOJ ENTERPRISES & XEROX**

**All soft ware institute materials, spiral-binding,**

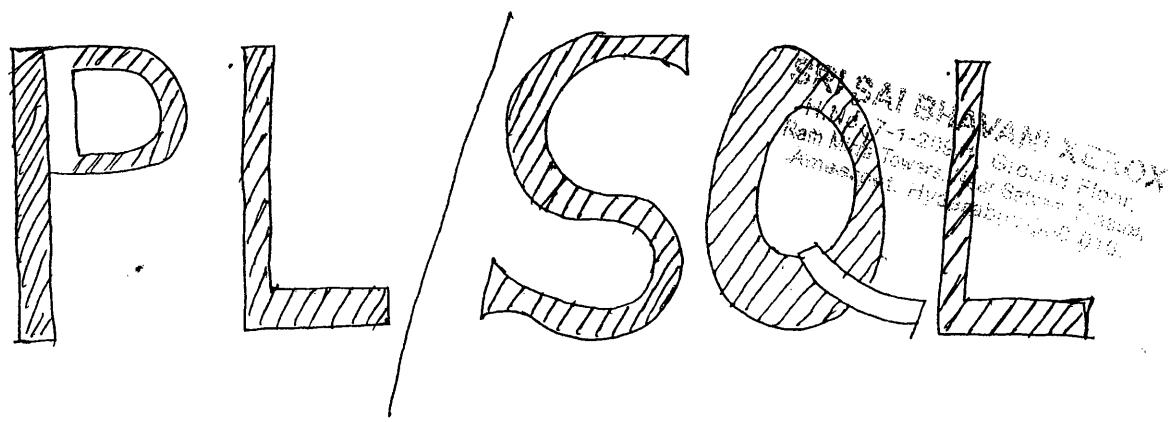
**Printouts & stationery also available ..,**

**Contact:8125378496**

Add: Plot No.40, Gayatri Nagar, Behind HUDA, mithrivannam, HYD.



**SRI SAI BHAVANI XEROX**  
H.No. 7-1-209/1, Ground Floor,  
Ram Mitra Towers, New Ameerpet Building,  
Ameerpet, Hyderabad - 500 016.



By - Mr. Murali  
(Natesh it)

**SRI SAI BHAVANI XEROX**  
H.No. 7-1-209/1, Ground Floor,  
Ram Mitra Towers, New Ameerpet Building,  
Ameerpet, Hyderabad - 500 016.

**MANOJ ENTERPRISES**  
Plot No: 40, Gayathri Nagar,  
Ameerpet, Hyderabad.  
Cell: 8125378496

C

C

C

C

C

C

Θ

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

31/16/116

## PL/SQL

Date - 03 - Dec - 15

### 1. Introduction

- select...into clause
- %type, %rowtype
- Conditional, Control statement
- bind variables.

SRI SAI BHAKTI VEDAM

H.No. 7-1-209/1, Ground Floor,  
Ram Mira Towers, Near Nizam Jnct.,  
Ameerpet, Hyderabad 500 013.

p/h: 9908195958

### 2. Cursor

- Implicit cursor
- explicit cursor
- explicit cursor life cycle
- explicit cursor attribute
- cursor.. for loop
- Parameterised cursor.
- update, delete statement are used in cursor (without using where current of, for update clauses).
- Implicit cursor attributes.

### 3. Exception -

- predefined exceptions
- userdefined exceptions
- Unnamed exceptions
- Exception Propagation
- Error trapping function  
(sqlcode, sqlerrm)
- raise\_application\_error()

SRI SAI BHAKTI VEDAM  
H.No. 7-1-209/1, Ground Floor,  
Ram Mira Towers, Near Nizam Jnct.,  
Ameerpet, Hyderabad 500 013.

p/h: 9908195958

### 4. Subprograms -

#### Stored Procedures:-

- procedure parameter  
(in, out, inout)
- no copy
- autonomous transaction
- authid current\_user.

#### Stored Functions:-

- dml statement used in function.
- select into clause used in function.
- out parameter used in function.
- cursor used in function
- wmi\_concat()

## 5. Triggers

- row level triggers
- application of row level triggers  
(auto increment, auditing a column)
- before / after (trigger timing)
- statement level triggers
- triggering events
- trigger execution order
- follows clause (oracle 11g)
- compound triggers (11g)
- mutating errors
- system triggers.

SRI SAI SHAWAR MEET  
H.NO. 7-1-200/1, Ground Floor,  
Pan Mira Towers, New Saboor Road,  
Ameerpet, Hyderabad-500 016.

## 6. Package :-

- global variable
- overloading procedures
- forward declaration

## 7. Types used in Packages:-

- pl/sql record
- index by table (or) pl/sql table (or) associative array
- nested table
- varray

visit  
date

## 8. Bulk Bind :-

- bulk collect clause
- forall statement
- indices of clause (10g)
- sql%bulk\_rows count()
- forall and dml errors  
(bulk exceptions)
- dbms\_utility package

## 9. REF CURSOR (or) CURSOR VARIABLE (or) dynamic cursor:-

- strong refcursor
- weak refcursor
- sys\_refcursor
- passing sys\_refcursor as in parameter to stored procedure.
- passing sys\_refcursor as out parameter to stored procedure.

10. Local Subprograms:-

- ursors used in local procedures
- types used as parameters to the local subprograms.

11. UTL-FILE Package

12. SQL \* LOADER

13. LOBS (large objects)

clob, blob, bfiles

14. where current of, for update clause are used in cursors.

15. avoiding mutating errors by using compound triggers.

16. member procedures, member functions

17. Dynamic SQL

18. Oracle 11g features.

19. Oracle 12c features.

SRI SAI BHAVANI XEROX  
Sri Sai Bhavani Xerox  
H.No. 7-1-269/1, Ground floor,  
Ramkrishna Puram, 1st Main Road,  
Ameerpet, Hyderabad-500 034.

Ph: 9908195958

## PL/SQL

SRI SAI BHAVANI XEROX

H.No. 7-4-209/1, Ground Floor,  
Ram Mitra Towers, Near Bagmane Theatre,  
Amarpet, Hyderabad 500 016.

- PL/SQL is a procedural language extension with SQL language.
- Oracle 6.0 introduced PL/SQL.
- It is a combination of SQL and procedural statements.

Oracle 6.0 - PL/SQL 1.0

Oracle 7.0 - PL/SQL 2.0

Oracle 8.0 - PL/SQL 8.0

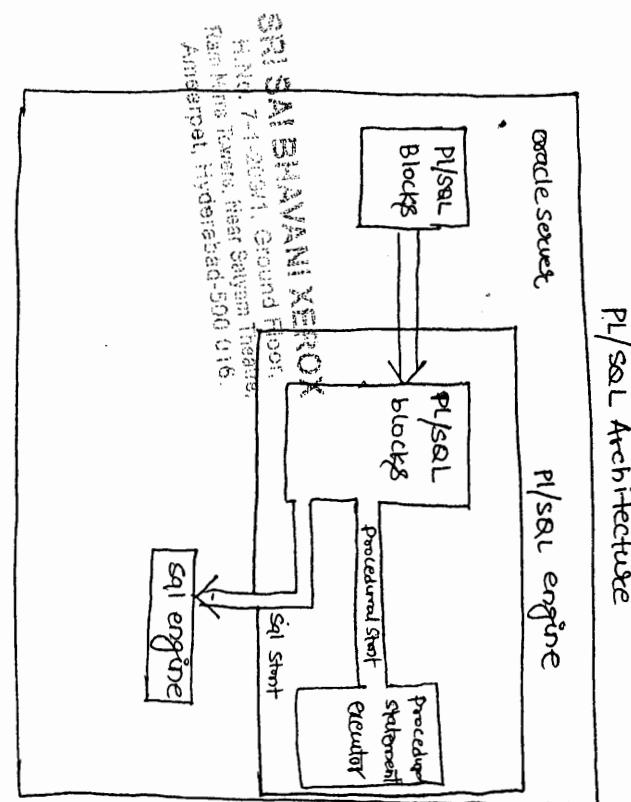
.....

.....

Oracle 12C - PL/SQL 12C

Basically PL/SQL is a block structure programming language. Whenever we are submitting PL/SQL blocks then all SQL statements are executing separately by using SQL engine and also all procedure statements are executed separately.

In procedural statement executor within PL/SQL engine.



## Block Statement

declare (optional)

- variable declaration, cursor, user defined exception

begin (mandatory)

DML, TCL;

Select into clause

Conditional, control statement

Exception (optional)

handling Runtime errors

end; (mandatory)

Declaring a variable :-

In declare section of the PL/SQL Block we can declare a variable.

Syntax -

variableName datatype (size)

e.g:- SQL> declare

a number(10);

b varchar2(20);

Storing a value into variable:-

Using assignment operator (:=)

Syntax -

variable name := value ;

SQL> declare

a number(10)

begin

a := 50 ;

end ;

/

SRI SAI BHAVANI XEROX

H.No. 7-1-208/1, Ground Floor,  
Ram Mirra Towers, Noor Bagh Colony,  
Ameerpet Hyderabad-500 016

Display a message (or) variable value:-

## Syntax -

```
dbms_output.put_line ('message');
```

(OR)

`dbms_output.put_line(variable_name);`

- This package is used in either in executable section or in exception section of the PL/SQL block.

```
SQL> set serveroutput on; ✓
```

sal> begin

```
dbms_output.put_line ('welcome');
```

end;

1

O/P : welcome .

- whenever we are passing data into `put_line` procedure from dbms\_output package then automatically passed the data into internal buffer . This data is not visible in SQL\* environment.

- If we want to display this data then we must set server output environment variable in SQL\* environment by using following :

## Syntax -

```
SQL> set serveroutput on; ✓
```

ex:- declare

a number (10)

begin

a:50;

```
dbms_output.put_line(a);
```

end ;

1

Output: 50

Select into clause -

- Select into clause is used to retrieve data from table into PL/SQL variable.
- select into clause always returns single record or single value at a time.

Syntax :-

```
select col1, col2, ... into var1, var2, ... from tablename
where condition;
```

- This clause is used in executable section of the PL/SQL block.

~~Ques.~~ write a PL/SQL program for user enter employee no that display name of the emp and his salary from emp table ?

```
declare
 v_ename varchar2(10);
 v_sal number(10);
begin
 select ename, sal into
 v_ename, v_sal from emp
 where empno = &n0;
 dbms_output.put_line(v_ename || ' ' || v_sal);
end;
/
```

Output: Enter value for n0: 7839  
KING 5000

~~Note :-~~ In PL/SQL whenever we are using not null or constant clauses in variable declaration then we must assign the value at the time of variable creation in declare section of the PL/SQL block.

Syntax :-

```
variable name datatype (size) not null := value;
```

Syntax -

```
variable name constant datatype (size) := value;
```

ex:-

```
declare
 a number(10) not null := 50;
 b constant number(10) := 7;
begin
```

```
dbms_output.Put_line (a);
dbms_output.Put_line (b);
```

Output: 50

7

Q. write a PL/SQL program to retrieve emp table and then display the salary?

```
declare
v_sal number(10);
begin
select max(sal) into v_sal;
from emp;
dbms_output.Put_line (v_sal);
end;
/
(09)
```

```
declare
a number(10);
b number(10);
c number(10);
begin
a := 70;
b := 30;
c := greatest(a,b); ✓
```

[ $c := \max(a, b)$ ] X wrong

```
dbms_output.Put_line (c);
```

```
end;
/
```

O/P: 70

## Note :-

- In PL/SQL expression we are not allowed to use group function, decode conversion function but we are allowed to use number function, character function and date function, date conversion function in PL/SQL expression.

declare

```
a varchar2(10);
begin
a:= upper('murali');
dbms_output.put_line(a);
end;
/
```

output: MURALI

## ✓ Variable Attribute :-

Variable attributes are used in place of data type in variable declaration whenever we are using attributes oracle server automatically allocates memory for the variable as corresponding column data types in a table.

PL/SQL having two types of variables attributes -

1. column level attributes ✓
2. Row level attributes ✓

### 1) column level attributes :-

In this method we are defining attributes for individual columns. These attributes are represented by using %.Type whenever we using these attributes oracle server internally automatically allocates some memory for the variable as corresponding column datatype in a type.

Syntax -

VariableName tablename.Columnname%Type

Variable attributes are also called as Anchor notation.

5/7/16

ex:-

```

declare
 v_ename emp.ename% type;
 v_sal emp.sal% type;
 v_hiredate emp.hiredate% type;
begin
 select ename, sal, hiredate into
 v_ename, v_sal, v_hiredate
 from emp
 where empno = &no;
 dbms_output.put_line (v_ename||' '||v_sal||' '||v_hiredate);
end;
/

```

Output? Enter the value of no : 7902

FORD 4100 03-Dec-81

Ans

### 2) Row-level-attributes :-

In this method a single variable can represent all different datatype into single unit. This variable is also called as record-type variable.

Row level attribute are represented by using %rowtype.

Syntax:-

|               |                    |
|---------------|--------------------|
| variable_name | tablename%rowtype; |
|---------------|--------------------|

EMP %rowtype

ex:-

```

declare
 i.emp%rowtype;
begin
 select ename, sal, hiredate into
 i.ename, i.sal, i.hiredate
 from emp where empno = &no;
 dbms_output.put_line
 (i.ename||' '||i.sal||' '||i.hiredate);
end;
/

```

→ It is same as  
structure in  
C-language

Output: enter value for no: 7902

FORD 3550 03-DEC-81

(02)

declare

i emp%rowtype

begin

Select \* into i from emp where

empno = &no;

dbms\_output.put\_line(i.ename || ' ' || i.sal || ' ' || i.hiredate);

end;

/

Output: enter value for no: 7902

FORD 3580 03-DEC-81

| empno | ename | iENAME | Job | mgr  | i.HIREDATE | sal  | i.SAL | i.deptno |
|-------|-------|--------|-----|------|------------|------|-------|----------|
| 7902  | FORD  |        |     | 7566 | 03-DEC-81  | 3550 | -     | 20       |

PL/SQL having 2 types of blocks.

(a) Anonymous block      (b) Named block.

#### Anonymous block

- These blocks do not have a name.
- These blocks are not stored in oracle database.

ex- declare  
....  
begin  
....  
end;

- These blocks are not allowed to call in client application.

#### Named block

- These blocks having a name.
- These blocks are automatically permanently stored in DB

ex- procedure and function

- These blocks are allowed to call in client application.

Syntax: variable\_name datatype (size) not null := value;

## ✓ Conditional Statement :-

1. If
2. if-else ✓
3. elseif ---> else if  
X

✓ 1. If :- syntax -  
if condition then  
    statement;  
end if;

✓ 2. If-else :-

Syntax :- if condition then  
    statement;  
else  
    statement;  
end if;

✓ 3. elseif :- else if

To check more no. of condition then we are using itself.

Syntax :-  
if condition1 then  
    stmt1;  
else condition2 then  
    stmt2;  
elseif condition3 then  
    stmt3;  
....  
....  
else  
    stmts;  
end if;

Prog :-

```
declare
 v_deptno number(10);
begin
 select deptno into v_deptno
 from dept where deptno = &deptno;
 if v_deptno = 10 then
 dbms_output.put_line('ten');
 elseif v_deptno = 20 then
```

```
dbms_output.put_line ('Twenty');
elseif v_deptno = 30 then
dbms_output.put_line ('Thirty');
else
dbms_output.put_line ('Others');
end if;
end;
/
```

Output: Enter the value for deptno = 40

Others

Enter the value for deptno = 90

error

Imp NOTE-I :- ✓

when PL/SQL block having "select into clause" and also  
if required data is not available in table then oracle server  
returns an error ORA-1403 no data found. Imp

✓NOTE-II :-

whenever PL/SQL block having "pure dml statement" and also  
if requested data is not available in table then oracle server  
does not return any error message.

for handling these type of block then we are using implicit cursor attributes.

Ex:-

```
(SQL) begin
 delete from emp where
 ename = 'welcome';
 end;
/
```

PL/SQL procedure successfully completed.

6/7/16

✓ NOTE-III:-

whenever select...into clause try to return multiple records or multiple values in a single column at a time then oracle Server returning an error ORA-1422; Exact fetch returns more than requested number of rows.

```
declare
 i emp%rowtype;
begin
 select * into i from emp
 where deptno = 20;
 dbms_output.put_line (i.empname || '||' || i.sal || '||' || i.deptno);
end;
/
```

Output: enter value for no: 10

ORA-1422: Exact fetch returns more than requested number of rows.

Imp ✓ Control Statement (Loops)

PL/SQL having 3 types of loops -

- ✓ 1. simple loop
- ✓ 2. while loop
- ✓ 3. for loop.

1. Simple loop :-

This loop is also called as infinite loop. Here body of the loop statement are executed repeatedly.

Syntax:-  
Loop  
 statements;  
end loop;

ex:- SQL> begin  
 Loop  
 dbms\_output.put\_line ('welcome');  
 end loop;  
 /

## To exist from infinite loop

To exist from infinite loop oracle provided two methods :-

### Method 1 :-

exist when True condition;  
declare

n number(10) := 1;

begin

loop

dbms\_output.put\_line(n);

exist when n >= 10;

n := n + 1;

end loop;

end;

/

Output :-

1  
2  
3  
4  
5  
6  
7  
8  
9  
10

### Method 2 :- (using if)

using if

Syntax :- { if true condition then }  
          { exist }  
          end if;

declare

n number(10);

begin

loop

dbms\_output.put\_line(n);

if (n >= 10) then

exist

end if;

end loop;

end;

/

Syntax :- if true condition then  
          exist;  
          endif;

## ✓ 2) while loop :-

Syntax - while condition  
loop  
  stmts;  
end loop;

Here condition always value boolean either true or false. whenever condition is true then only control goes to body of the loop then those loop stmts are executed repeatedly until condition become false.

declare

```
n number(10);
begin
 while n <= 10
 Loop
 dbms_output.put_line(n);
 n := n + 1;
 end loop;
 end;
 /
```

Output :-  
1  
2  
3  
4  
5  
6  
7  
8  
9  
10

## ✓ 3) for loop :-

Syntax :-

for indexvariablename in lowerbound .. x .. upperbound  
loop  
  stmts;  
end loop;

↑  
only 2 dots are written only  
Here

Ex - declare

```
n number(10);
begin
 for n in 1..10
 Loop
 dbms_output.put_line(n);
 end loop;
 end;
 /
```

O/P :-  
1  
2  
3  
4  
5  
6  
7  
8  
9  
10

⇒ declare

```
n number(10);
begin
 for n in reverse 1..10
 Loop
 dbms_output.put_line(n);
 end;
 /
```

O/P :-  
10  
9  
8  
7  
6  
5  
4  
3  
2  
1

8/7/16

To process multiple records → it is also called Static CURSOR

## CURSOR

Chapter-1

- cursor is a private SQL memory area which is used to process multiple records & also this is a record by record process.
- All database system having 2 type of static cursor these are:-
  1. Implicit cursor
  2. Explicit cursor.

### Explicit cursor :-

For SQL statement returning multiple records, it is called Explicit cursor & also this is a record by record process.

Explicit cursor memory area is also called as active set area.

Explicit cursor life cycle:- Every Explicit cursor having 4 steps then an

1. declare
2. open
3. fetch
4. close

**SRI SAI BHAVAN HOTEL**  
H.No. 7-1-203/1, Ground Floor,  
Ram Mohan Towers, Nizam Colony, Tarnaka,  
Amarapur, Hyderabad-500 016.

① declare:- In declare section of the PL/SQL block we are defining the cursor by using following syntax

#### Syntax:-

cursor cursorname is select \* from tablename where condition;

ex:- SQL> declare

cursor c1 is select \* from emp  
where job = 'CLERK';

② open:-

#### Syntax:-

open cursorname;

This open statement used in executable section of the PL/SQL blocks.

Whenever open in cursor then only crack starts fetching in to cursor memory or because all data bases whenever opening the cursor then only cursor Select statements are executed internally.

### NOTE:-

Whenever we are opening the cursor always cursor pointer point to first record in the cursor.

### (3) fetch:-

fetching data from cursor.

→ Using fetch statement we are fetching data from cursor into <sup>(6.1)</sup> cursor memory area pl/sql variable.

syntax - fetch cursorname into varname1, varname2 .....

### (4) close :-

whenever we are closing the cursor by using close statement then automatically all resources allocated from cursor memory area released.

syntax :- close cursorname;

declare

cursor c1 is select ename, sal from emp;

v\_empname varchar2(10);

v\_sal number(10);

begin

open c1;

fetch c1 into v\_empname, v\_sal;

dbms\_output.put\_line(v\_empname || ' ' || v\_sal);

fetch c1 into v\_empname, v\_sal;

dbms\_output.put\_line(v\_empname || ' ' || 'high salary');

close c1;

/

Output:

SMITH 5000

Cursor → To process multiple records  
→ Record by record process.

11/7/16

### Explicit cursor attributes:-

Every explicit cursor having four attributes these are—

1) %notfound

2) %found

3) %isopen

4) %rowcount

- whenever we are using these attributes in PI/SQL block then the must specify cursor name along with these attributes.

#### Syntax :-

cursorname % attribute name.

→ except %rowcount all other cursor attributes returns boolean value either true or false whereas %rowcount attributes always returns no. datatype.

#### 1) %notfound :-

This attribute returning boolean value either true or false.

- This attribute returns true when fetch statement does not return any row whereas attribute returns false when fetch statement returns at least one record.

#### Syntax :-

cursorname % notfound

write a PI/SQL cursor program to display all employee names and their salary from emp table by using %notfound attribute?

declare

cursor c is select ename, sal from emp;

v\_ename varchar2(10);

v\_sal number(10);

```
begin
open c1;
loop
fetch q into v_ename, v_sal;
exit when q%notfound;
dbms_output.put_line (v_ename || '||' || v_sal);
end loop;
close c1;
end;
/
```

Q.

SRI SAI BHAVANI XEROX  
H.No. 7-1-203/1, Ground floor,  
Ram Mira Towers, Near Suleman Toli Bazar,  
Ameerpet, Hyderabad-500 019.

SRI SAI BHAVANI XEROX  
H.No. 7-1-203/1, Ground floor,  
Ram Mira Towers, Near Suleman Toli Bazar,  
Ameerpet, Hyderabad-500 019.

IMP

Q. Write a PL/SQL cursor program which is used to calculate total sal. from emp table without using sum() function?

```
declare
cursor c1 is select sal from emp;
v_sal number(10);
n number(10) := 0;
begin
open c1;
loop
fetch c1 into v_sal;
exit when c1%notfound;
n := n + v_sal;
end loop;
dbms_output.put_line('total salary is :'||n);
close c1;
end;
/
```

Output: total salary is : 36975

Note:-

whenever resource column having null values and also when we're calculating summation then the oracle server returns total sum=null  
to overcome this problem we must use nvl() function.

example -

$n := n + \text{nvl}(v\_sal, 0);$

Q. write a PL/SQL program which is used to display 5<sup>st</sup> five highest salary employee's from emp table by using %rowcount attribute?

```
declare
cursor c1 is select ename, sal from emp order by sal desc;
v_ename varchar2(10);
v_sal number(200);
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) write PL/SQL program to display even no. of records from emp table by using rowcount attribute?

SQL> declare

```

cursor c1 is select ename, sal from emp;
v_ename varchar2(10);
v_sal number(10);
begin
open c1;
loop
fetch c1 into v_ename, v_sal;
exit when c1%notfound;
if mod(c1%.rowcount, 2) = 0 then
dbms_output.put_line(v_ename || ' ' || v_sal);
end if;
end loop;
close c1;
end;
/

```

SQL> declare

```

a number(10) a 50
b boolean; b true

```

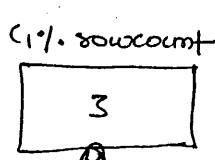
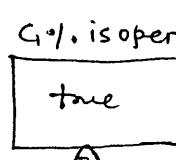
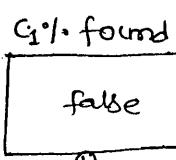
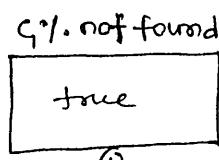
begin

a := 50

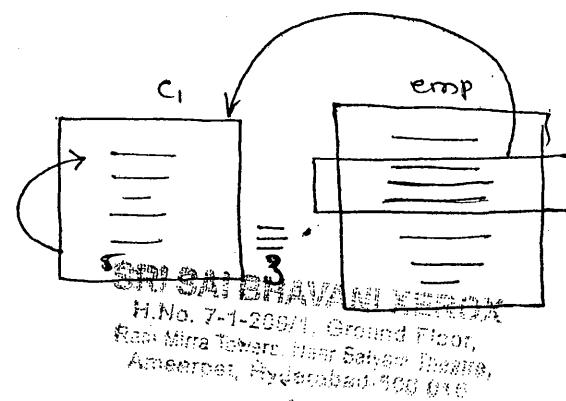
b := true

end;
/

4 memory areas



In oracle whenever we create a cursor, then oracle server automatically



18/7/16

## ✓ 2) %rowCount

This attribute always returns number datatype i.e. it counts no. of records number fetched from the cursor.

### Syntax

```
cursorname%rowcount
```

ex- 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;
```

```
fetch c1 into v_ename, v_sal;
```

```
fetch c1 into v_ename, v_sal;
```

```
dbms_output.put_line('number of records number fetched
from the cursor is:' || '|| c1%rowcount);
```

```
close c1;
```

```
end;
```

```
/
```

Output:-

Number of records number fetched from cursor is : 3

Note:-

Using cursor we can also transfer data from one oracle table into another oracle table.

Ex:-

```
SQL> create table target (sno number(10), name varchar2(10), salary number);
```

SQL> declare

```
cursor c1 is select * from
emp where sal > 2000;
```

```
i emp%rowtype;
```

```
n number(10);
```

```
begin
```

```
open c1;
```

```
loop
```

```
fetch c1 into i;
```

```

exit when c1%notfound;
n := c1%rowcount;
insert into target
values (n, i.ename, i.sal);
end loop;
close c1;
end;
/

```

SQL> select \* from target;

| SNO | Name   | sal  |
|-----|--------|------|
| 1   | smith  | 2990 |
| 2   | Jones  | 3250 |
| 3   | millar | 3370 |
| 4   | Allen  | 3478 |
| ... | ...    | .... |

Q. Write a PI/SQL cursor program to display 5<sup>th</sup> record from emp table by using %rowcount attribute?

SQL> declare

```

cursor c1 is select * from emp;
i emp%rowtype;
begin
open c1;
loop
fetch c1 into i;
exit when c1%notfound;
if c1%rowcount = 5 then
dbms_output.put_line(i.ename || ' ' || i.sal);
end if;
end loop;
close c1;
end;
/

```

Output: MARTIN 950

13/7/16

### 3. > % FOUND -

- This attribute returns boolean value either true or false
- This attribute returns true when Fetch Statement returning atleast one records. This records attribute returns false when fetch statement doesnot return any records.

#### Syntax

Cursorname % found

SQL> declare

```
cursor c1 is select * from emp
where ename = '&ename';
i emp%rowtype;
begin
open c1;
fetch c1 into i;
if c1%found then
dbms_output.put_line ('u r employee exists' || i.ename || '||'
i.sal);
else if c1%notfound then
dbms_output.put_line ('ur employee doesnot exists');
end if;
close c1;
end;
/
```

output: enter value for ename: murali  
employee doesnot exists

SQL/

output: enter value for ename: KING

employee exists KING 7400

Q) write a PL/SQL cursor Program to display all the employee's, and their salary from emp table by using %.found attribute?

SQL> declare

```
cursor c1 is select * from emp;
i emp%rowtype;
begin
open c1;
fetch c1 into i;
while (c1%found)
```

→ In this program 'fetch' is used  
two times.

```
loop
dbms_output.put_line(i.ename || '||' || i.sal);
fetch c1 into i;
end loop;
close c1;
end;
/
```

| <u>Output:</u> | ename | sal |
|----------------|-------|-----|
| miller         | 2500  |     |
| King           | 2890  |     |
| mark           | 3210  |     |
| Jones          | 3570  |     |
| :              | :     |     |
| :              | :     |     |

#### 4.) %open -

This attribute also returns boolean value either true or false.  
this attribute returns true when cursor already open. It  
returns false when cursor is not open.

##### Syntax

```
cursorname%open
```

```
SQL> declare
```

```
cursor c1 is select * from emp;
```

```
i emp%rowtype;
```

```
begin
```

```
if not c1%open then
```

```
open c1;
```

```
end if;
```

```
loop
```

```
fetch c1 into i;
```

```
exit when c1%notfound;
```

```
dbms_output.put_line(i.ename || '||' || i.sal);
```

```
end loop;
```

```
close c1;
```

```
end;
```

```
/
```

| attribute Name | Return value | Condition                                                |
|----------------|--------------|----------------------------------------------------------|
| %found         | True         | If fetch statement return at least one row.              |
|                | false        | If fetch statement doesn't returns any row.              |
| %notfound      | True         | If fetch statement doesn't return any row.               |
|                | false        | If fetch statement returns at least one row.             |
| %Isopen        | True         | If cursor is already opened.                             |
|                | false        | If cursor is not opened.                                 |
| %rowcount      | number       | It counts no. of records number fetches from the cursor. |

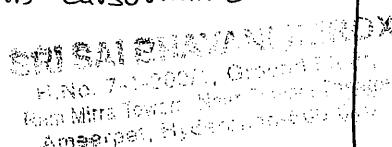
IMP

### Eliminating Explicit Cursor life cycle (or) cursor for loops :-

- Using cursor "for" loop we are eliminating explicit cursor life-cycle i.e. whenever we are using cursor forloop no need to use open, fetch, close statement explicitly i.e. when we are using cursor for loop oracle server only internally automatically open the cursor, and then fetch data from the cursor and close the cursor.

Syntax -

```
for indexVarName in cursorName
Loop
 statements;
end loop;
```



- cursor forloop is used in executable statement of the pl/sql block.

Note:-

In cursor for loop index variable internally behaves like a record type variable. ( %rowtype )

- Q. write a pl/sql cursor programs to display all employee names & their salary from emp by using cursor For loop ?

SQL> declare

```
cursor C, is select * from emp;
begin
 for i in C
 Loop
```

```

dbms_output.put_line(i.ename || i.sal);
end loop;
end;
/

```

Note:-

We can also eliminate declare section of the cursor by using cursor for loop in this case we must specify cursor select statement within parenthesis in place of cursor name within cursor for loop.

Syntax

```

for indexvarname in (select statement)
loop
 start;
end loop;

```

Ex:- begin

```

for i in (select * from emp) -
Loop
dbms_output.put_line(i.ename || i.sal);
end loop;
end;
/

```

Q. Write a PL/SQL cursor programs to display 5<sup>th</sup> record from emp table by using cursor for loop?

SQL> declare

cursor(c1) is select \* from emp;

begin

for i in c1 (

loop

If c1%rowcount = 5 then

dbms\_output.put\_line(i.ename || i.sal);

end if;

end loop;

end;

Output: martin 900

keyword

SPU SAIBHANAH ALI  
H.NO. 7-1-209/1, Ground Floor,  
Nimmita Towers, New Nizam Pura,  
Ameerpet, Hyderabad - 500 016.

(if C1 (cursor name)  
is there we must  
declare required  
section)

Q) Write a PL/SQL cursor program which is used to display total salary from emp table without using sum() function by using cursor for loop ?

SQL> declare

```
cursor c1 is select * from emp;
n number(10) := 0;
begin
for i in c1
loop
n := n + i.sal;
end loop;
dbms_output.put_line('total salary is:'||' '||n);
end;
```

Output - Total salary is : 42075

### ② Parameterized Cursor :-

\* In oracle we can also pass parameters to the cursor same like a subprogram "in" parameters. These parameters cursor are also called as parameterized cursor.

→ In parameterized cursor we are defining formal parameter when we are declaring the cursor and also pass actual parameters when we are open in the cursor.

#### Note :-

- In oracle whenever we defining formal Parameter in cursor, procedures, functions ~~are~~ then we are not allowed to use datatype size in formal parameter declaration.

#### Syntax -

```
cursor cursorname (Parametername datatype)
is select * from tablename -
where columnname = Parametername;
```

#### Syntax :

(or)  
open cursorname (actual parameter);

Ex 7

SQL> declare

```
cursor c1 (P_deptno number)
is select * from emp where
deptno = P_deptno;
i emp%rowtype;
begin
open c1(10); for i in c1 (P_deptno)
loop
fetch c1 into i;
exit when c1%notfound;
dbms_output.put_line(i.ename || '|| i.sal || '|| i.deptno);
end loop;
close c1;
end;
/
```

Output:

|        |      |    |
|--------|------|----|
| CLARK  | 2700 | 10 |
| MILLER | 7600 | 10 |
| JONES  | 2700 | 10 |

- Q. Write a PL/SQL Parametrized cursor for Passing Job is as parameter from emp table that display the employee working as 'CLERKS' or 'Analyst' and also display output statically based on following format -

Employee working as CLERKS

SMITH

MILLER

KING

:

Employee working as ANALYST

SCOTT

FORD

:

SQL> 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 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 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 then we must close the cursor properly otherwise oracle server returning an error.

error: ora-6511: cursor already open.

Note :-

when we are not open the cursor but we are try to perform operation to the cursor then oracle server returning an error, ora-1001 : invalid cursor.

SQL> 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;
/

```

```

SQL> declare
 cursor C1 (P_JOB VARCHAR2) is
 select * from emp where
 job = P_JOB;
 begin
 dbms_output.put_line ('Employee working as clerk');
 for i in C1 ('CLERK')
 loop
 dbms_output.put_line (i.ename);
 end loop;
 dbms_output.put_line ('Employee working as Analyst');
 for i in C1 ('ANALYST')
 loop
 dbms_output.put_line (i.ename);
 end loop;
 end;
/

```

NOTE:-

In all databases whenever we are passing data from one cursor into another cursor then receiving cursor must be a parameterized cursor.

- Q. write a PL/SQL cursor program which is used to retrieve all department no. from dept table by using static cursor and also pass this static cursor values into another parameterized cursor which is used to retrieve emp details from emp table?

```

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 (i.ename || '||' || i.sal || '||' || i.deptno);
end loop;
end loop;
end;
/

```

Output: my dept table deptno is : 10

|        |      |    |
|--------|------|----|
| CLARK  | 2800 | 10 |
| KING   | 7700 | 10 |
| MILLAR | 2800 | 10 |

My dept table deptno is : 20

|       |      |    |
|-------|------|----|
| SMITH | 3000 | 20 |
|-------|------|----|

#### NOTE:-

In cursor we can also use function, expression, group functions, in this case we must use alias name from those expression, function  
 also we must declare cursor record type variable in declare section of PI/SQL Block.

\* C1%rowtype;

#### Syntax:

variable name cursorname%rowtype;

Q) write a PI/SQL cursor program to display total salary from emp table by using sum() function.

```

declare
cursor c1 is select sum(sal) a from emp;
c1%rowtype;
begin
open c1;
fetch c1 into i;
dbms_output.put_line ('total salary is:' || i);
close c1;
end;
/

```

O/P:- Total salary is : 38525

Q) write PL/SQL program using parameterized cursor for user entered deptno then display no. of employees, min sal, max sal, total salary in that dept using following format -

Enter the value for deptno:

no. of employee's are :

minimum salary is :

maximum salary is :

Total salary is :

declare

cursor c1 ( P\_deptno number ) is select count(\*) a, min(sal) b, max(sal) c,  
sum(sal) d .

from emp where deptno = P\_deptno;

c1%rowtype;

begin

open c1 (&deptno);

fetch c1 into i;

dbms\_output.put\_line ('number of employee's are : ' || i.a);

dbms\_output.put\_line ('minimum salary is : ' || i.b);

dbms\_output.put\_line ('maximum salary is : ' || i.c);

dbms\_output.put\_line ('Total salary is : ' || i.d);

close c1 ;

end;

/

O/P: Enter the value for deptno: 10

no. of employee's are : 3

Minimum salary is : 2800

Maximum salary is : 7700

Total salary is : 13300

Note:-

In oracle we can also pass default values in parameterised cursor by using default (or) := operator.

Syntax

Parametername datatype default [:=] defaultvalue

- update, delete statement are used in cursors (without using where current of, for update clauses)

Q. Write a PL/SQL cursor program to update salary of the employee in emp table based on following cond'—

(a) if job = 'CLERK' then increment sal  $\rightarrow$  100;

(b) if job = 'SALESMAN' then decrement sal  $\rightarrow$  200;

SQL> declare

```
cursor c1 is select * from emp;
i emp%rowtype;
begin
open c1;
loop
fetch c1 into i;
exit when c1%notfound;
if i.job = 'CLERK' then
update emp set sal = sal + 100 where empno = i.empno;
elsif i.job = 'SALESMAN' then
update emp set sal = sal - 200 where empno = i.empno;
end if;
end loop;
close c1;
end;
/
```

SQL> select \* from emp;

~~Implicit Cursor attributes:-~~

For SQL statements returning single record, it is called implicit cursor.

Implicit cursor memory area is also called as ~~memory~~ context area.

~~Ex:-~~ SQL> declare

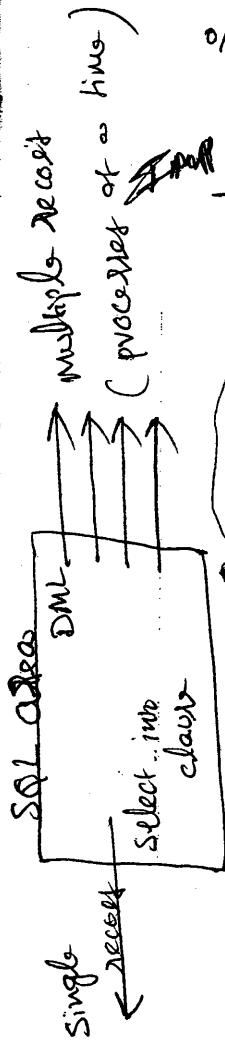
```
v_ename varchar2(10);
v_sal number(10);
begin
select ename,sal into v_ename,v_sal from emp
where empno = &n;
```

```

dbms_output.Put_line (v_ename || '||' || v_sal);
end;
/

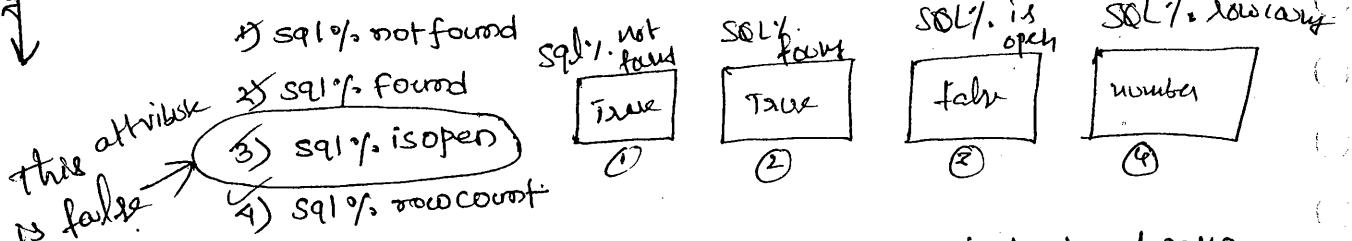
```

O/P:- Enter value for no: 7902  
FORD 3800



- In oracle whenever pl/sql block having select into clause or pure DML statements then oracle serves internally automatically allocated memory area this memory area is also called as SQL area or context area or implicit cursor.
- This memory area always returns single record when pl/sql Block having select...into clause.
- This memory area also returns multiple records when pl/sql block having pure DML statement but these DML statements are processed at a time by using SQL engine that's why in this case developer can't control individual records.
- Along with implicit cursor memory area oracle serves automatically allocated k-memory at locations. These memory location are identified through implicit cursor attributes. These memory locations behaves like a variable.

These implicit attribute are -



- In implicit cursor attribute sql% is open attribute is always return false where as sql% not found, sql% found attributes returns Boolean value either true or false and sql% rowcount attribute returns number data type.

```

SQL> set serveroutput on;
SQL> begin
 delete from emp where
 ename = 'welcome';
 end;
/

```

PL/SQL procedure successfully completed

- solution (using implicit cursor attribute)

```
SQL> begin
 delete from emp where
 ename = 'welcome';
 if sql%found then
 dbms_output.put_line ('your record deleted');
 end if;
 if sql%notfound then
 dbms_output.put_line ('Your record input doesnot exists');
 end if;
 end;
/
output: your record input doesnot exists.
```

```
SQL> begin
 update emp set sal = sal + 100
 where job = 'CLERK';
 dbms_output.put_line ('affected number of clerks are:'
 || sql%rowcount);
 end;
/
output: affected number of clerks are : 4
```

18/12/16

SRI SAI BHAVANI XEROX  
H.No. 7-1-209/1, Ground Floor,  
Ram Mirra Towers, Near Satyam Theatre,  
Ameerpet, Hyderabad-500 016.

Date  
22/Dec/15.

## Exception

- Exception is an error occurred in Run-time.
- whenever the run-time error is occurred use an appropriate exception name in exception handler under the exception section of the PL/SQL Block.
- oracle provided three types of exception these are -
  - 1) Predefine exception
  - 2) Userdefine exception
  - 3) Unnamed exception.

### Imp 1: Predefine exception:-

oracle provided 20 Predefine exception name for regularly occurred run-time error.

Whenever the run-time error is occurred use corresponding P.exception name in exception handler and exception section of PL/SQL Block.

Syntax - When predefined exception name<sub>1</sub> then  
stmt<sub>1</sub>;  
When predefined exception name<sub>2</sub> then  
stmt<sub>2</sub>;  
.....  
When others then  
stmt<sub>n</sub>;

### → Predefined exception name:-

- 1. no\_data\_found
- 2. Too\_many\_rows
- 3. zero\_divide
- 4. dup\_val\_on\_index
- 5. Invalid\_cursor
- 6. cursor\_already\_open

} But

- ✕ \*\*\*  
 7. invalid\_number [Bmp]  
 ✕ \*\*\*  
 8. value\_error

### ✓ 1) no\_data\_found →

whenever PL/SQL Block having select into clause and also ~~the~~ if requested data is not available then oracle server returns an error ora-1403: no data found. For handling this error oracle provided no-data-found exception name.

ex:- 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 doesn't exists');
end;
/

```

O/P:- enter value for no: 7902

FORD 3800

enter value for no: 11

your employee doesn't exists.

### ✓ 2) too\_many\_rows:-

whenever select into clause try to return multiple records or try to return multiple values in a single column at a time then oracle server returns an error ora-1422: Exact fetch returning more than requested number of rows.

for handling this type of error oracle provided too-many-rows exception name.

example -

```
declare
 v_sal number(10);
begin
 select sal into v_sal from emp;
 dbms_output.put_line(v_sal);
exception
 when too_many_rows then
 dbms_output.put_line('not to return multiple rows');
end;
/
```

output:- not to return multiple rows.

### ~~dup\_val\_on\_index~~ :-

In oracle when we try to insert duplicate value into primary key column or when we try to insert duplicate value or unique constraint column then oracle server returns an error ora-0001 : unique constraint violated for handling this error oracle provided dup\_val\_on\_index exceptionname.

ex:-

```
begin
 insert into
 emp(empno) values(7902);
exception
 when dup_val_on_index then
 dbms_output.put_line('not to insert duplicates data');
end;
/
```

O/P: not to insert duplicates data

### ~~10~~ Invalid\_cursor :-

In oracle when we are not open in cursor but we are try to perform operation of the cursor then oracle server returns an error ora-1001: invalid\_cursor for handling this error then we are using invalid\_cursor exception name.

ex:- declare

```
cursor c1 is select * from emp;
 i emp%rowtype;
begin
loop
 fetch c1 into i;
 exit when c1%not_found;
 dbms_output.put_line(i.ename || i.sal);
end loop;
close c1;
exception
when invalid_cursor then
 dbms_output.put_line('first we must open cursor');
end;
/
```

### ~~10~~ cursor\_already\_open :-

Before we are reopen in the cursor we must close the cursor properly otherwise oracle server returns an error; cursor\_already\_open for handling this error oracle provided cursor\_already\_open exception name.

ex:-

~~error~~ :- ora-6511: cursor already open

declare

```
cursor c1 is select * from emp;
 i emp%rowtype;
begin
open c1;
loop
 fetch c1 into i;
 exit when c1%not_found;
 dbms_output.put_line(i.ename || i.sal)
end loop;
open c1;
```

done  
2016/7/19

exception

when cursor\_already\_open then

dbms\_output.put\_line('First we must close the cursor before  
reopen the cursor');

end;

/

## 6) zero\_divide :-

In oracle when we are try to perform division by zero then oracle serves returning an error.

ORA-1476 : divisor is equal to zero

for handling this type of error then we are using zero\_divide exception.

ex:-

begin

dbms\_output.put\_line(3/0);

exception

when zero\_divide then

dbms\_output.put\_line('not to perform division of with  
zero');

end;

/

SRI SAI BHAVAN  
H.No. 7-1-1000/1, Ground floor,  
Ram Mirra Layout, Naer Ghati Thora,  
Ameerpet, Hyderabad-500 016

Fall \*\*\*  
(7,8)  
\* \* \*  
\* \* \*

## Invalid\_number, value\_error :-

In oracle when we try to convert string type to number type or "date string into date type" then oracle serves returning two types of errors.

a.) Invalid\_number

b.) Value\_error or numeric error

## a.) Invalid\_number -

when PL/SQL Block have a SQL statements and also those SQL statements try to convert So string type to number type or Date String into Date type then oracle serves returning an error : ora-1722 - Invalid number  
for handling this error oracle provided exception invalid\_number exception name.

ex:-

```
begin
 insert into
 emp (empno, ename, sal) values (1, 'Gokul', 'abc');
exception
 when invalid_number then
 dbms_output.put_line ('insert proper data only');
end;
/
```

(b) value\_error:-

Whenever PL/SQL block having procedural statements and also those statements find to convert string type to number type then oracle server returns an error

ora-6502 : numeric or value error: character to number conversion error

For handling this error oracle provided exception value\_error exceptionname.

ex:- begin

```
declare
 z number(10);
begin
 z := '&x' + '&y';
 dbms_output.put_line(z);
exception
 when value_error then
 dbms_output.put_line ('enter numeric data value for x & y only');
end;
/
```

SRI SAI BHAVANI XEROX  
H.No. 7-1-209/1, Ground Floor,  
Ram Mirra Towers, Near Satyam Timings,  
Ameerpet, Hyderabad-500 016

Output:- Enter value for x: 3  
Enter value for y: 2

z := 5

Enter value for x: a  
Enter value for y: b

error: enter numeric data value for x & y only.

### NOTE:-

- In oracle when we are try to store more data then the datatype size specified in varchar2 datatype then also oracle sever returns an error ora-6502- numeric or value error: character string buffer too small.  
for handling this error also we using value\_error exception.

```

declare
z varchar2(3);
begin
z := 'abcd';
dbms_output.put_line(z);
exception
when value_error then
dbms_output.put_line('invalid string length');
end;
/
output: invalid String length.

```

21/7/16

### Exception Propagation

- In pl/sql exception also occurred in declare section, executable section, exception section.  
when exception raised in executable section those exception are handle either in inner blocks either outer blocks.  
where as when exception are raised in declare section or in exception section those exception must be handled by using outer block only. This is called PL/SQL Exception propagation.

```

declare
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-6502: numeric or value error: character string  
buffer too small.

Sol:-

```

begin
declare
z varchar2(3) := 'abcd';
begin
dbms_output.put_line(z);
exception
when value_error then
dbms_output.put_line('invalid string length');
end;
exception
when value_error then
dbms_output.put_line('invalid string length handled through outer
block only');
end;
/

```

Output: Invalid string length handled through outer block only.

### \* User-defined exception :-

In oracle we can also create our own exception name and raised whenever necessary those exception explicitly these type of exception are also called as user-defined exception.

- In all DB's if we want to raise exception based on client business rule then only we are using user-defined exception.

### Handling user-defined Exception in oracle -

- Step 1 - declare ✓
- Step 2 - raise ✓
- Step 3 - handling exception ✓

#### Step-1 (declare) -

In declare section of the pl/sql block we are creating our own exception name by using exception predefined type through following syntax -

Syntax - userdefinedname exception;

Ex:-    SQL> declare  
          a exception;

### Step-2 (raise) :-

In raise statement we can raised user defined exception explicitly either executable section or a exception section of the PL/SQL Block.

Syntax - raise userdefined exception name;

### Step-3 (handling userdefine exception) :-

When we can also handle user defined exception same like a Predefined exception by using exception handler. In exception section of in PL/SQL Block.

Syntax - when userdefined Exception name1 then

Start;

when user defined exception name 2 then

Start;

.....

.....

when others then

Start;

Q) write a PL/SQL program which raise a user-defined exception on thursday?

declare

a exception;

begin

if to\_char(sysdate, 'DY') = 'THU'

then

raise a;

end if;

exception

when a then

dbms\_output.put\_line ('my exception raised on thursday');

end;

/

O/p: my exception raised on thursday.

```

declare
v_sal number(10);
a exception;
begin
select sal into v_sal from emp
where ename = 'KING';
if v_sal > 2000 then
raise a;
else
update emp set sal = sal + 100 where
ename = 'KING';
end if;
exception
when a then
dbms_output.put_line ('salary already high');
end;
/

```

- Q. write a PL/SQL program by using following table whenever user entered empno is more than 10 or less than 1 then raise user define exception otherwise user entered empno available on that emp table then display name of the emp from that table based on user entered empno.

SQL> create table test as select rownum empno, ename from emp where rownum <= 10;

SQL> select \* from test;

| EMPNO | ENAME  |
|-------|--------|
| 1     | SMITH  |
| 2     | MARTIN |
| 3     | ALLEN  |
| 4     | JONE   |
| 5     | BLAKE  |
| :     | :      |

SQL> declare
v\_sal number(10);
v\_ename varchar2(10);
v\_empno ~~number~~ number(10);
a exception;

```

begin
 v_empno := &no;
 if v_empno > 10 or v_empno < 1 then
 raise a;
 else
 select ename into v_ename from test where empno = v_empno;
 dbms_output.Put_line (v_ename || '||' || v_sal);
 end if;
 exception
 when a then
 dbms_output.Put_line ('Your empno is out of range');
 end;
/

```

Output: Enter the value for no: 90

Your empno is out of range

SQL> /

Enter the value for no: 2

ALLEN

#### NOTE:-

In all databases whenever condition is true then only database server raise exception procedure and also server doesn't execute remaining executable statement whereas when we are using normal procedure in place of exception procedure whenever condition is true then all the remaining statement is executed.

Ex:-

```

declare
 a exception;
begin
 if to_char(sysdate, 'DY') = 'SAT' then
 raise a;
 dbms_output.Put_line ('Control does not go to next line');
 end if;
 exception
 when a then
 dbms_output.Put_line ('My exception raised on Saturday');
 end;
/

```



Output: my exception raised on saturday

ex-2

```
begin
 if to_char(sysdate, 'DY') = 'SAT'
 then
 dbms_output.put_line ('my exception raised on saturday.');
 dbms_output.put_line ('control goes to next line');
 end if;
end;
/
```

Output: my exception raised on saturday.  
Control goes to next line.

NOTE:-

In oracle when a pl/sql block having cursor and also if the requested data not in the table is available then if we want to display proper message then we are using userdefine Exception with in pl/sql blocks.

eg- declare

```
cursor c1 is select.* from emp
where deptno = &deptno;
:emp%rowtype;
a exception;
begin
 open c1;
 fetch c1 into :i;
 if c1%rowcount = 0 then
 raise a;
 else
 while (c1%found)
 loop
 dbms_output.put_line (:i.ename||' || :i.sal || ' || :i.deptno);
 fetch c1 into :i;
 end loop;
 end if;
exception
 when a then
 dbms_output.put_line ('your requested deptno doesnot exist');
 /

```

D/P:

Enter value for deptno = 10 Enter value for deptno = 80

|        |    |
|--------|----|
| CLARK  | 10 |
| KING   | 10 |
| MILLER | 10 |

it does not exist

```

declare
cursor c1 is select * from emp
where deptno = &no;
i.emp%rowtype;
begin
open c1;
fetch c1 into i;
if c1%rowcount = 0 then
raise no_data_found;
else
while (c1%found)
loop
dbms_output.put_line (i.empname || '||' || i.deptno);
fetch c1 into i;
end loop;
end if;
close c1;
exception
when no_data_found then
dbms_output.put_line ('your deptno doesn't exists');
end;
/

```

Output: Enter the no : 20

| ename  | deptno |
|--------|--------|
| mark   | 20     |
| miller | 20     |
| jone   | 20     |

Enter the no : 90

your deptno doesn't exists.

(no-data-found)

Note :-

In oracle we can also raise predefined exception, excepting by using raise statement

Syntax of raise predefinedexceptionname;

Testing Exception Propagation by using userdefine exception:-

Exception raised in executable section:-

method-1: handled by using inner blocks:-

```
declare
 a exception;
begin
 raise a;
exception
when a then
 dbms_output.put_line ('handled by using inner blocks');
end;
/
output: handled by using inner blocks.
```

method-2: handled by using outer blocks:-

```
declare
 a exception;
begin
 begin
 raise a;
 end;
exception
when a then
 dbms_output.put_line ('handled by using outer blocks');
end;
/

```

output: handled by using outer blocks.

## Exception raised in exception section:-

In oracle when exception raised in declare section or exception section, those exception must be handled by using outer blocks only.

```
declare
 z1 exception;
 z2 exception;
begin
 raise z1;
exception
 when z1 then
 dbms_output.put_line ('z1 handled');
 raise z2;
 end;
 /

```

Output: z1 handled

Error: unhandled user-defined exception.

## Solution :-

```
declare
 z1 exception;
 z2 exception;
begin
 begin
 raise z1;
 exception
 when z1 then
 dbms_output.put_line ('z1 handled');
 raise z2;
 end;
 exception
 when z2 then
 dbms_output.put_line ('z2 handled');
 end;
 /

```

Output: z1 handled

z2 handled

### Un-named exception:-

- In oracle if we want to handle other than oracle 20 predefine exception name error then we must used un-named method.
- In this method we are creating our own exception name and they associates this exception name with appropriate error number by using EXCEPTION\_INIT function. This function accepts two parameter.
- **syntax -**

```
Pragma exception_init (userdefine exception name, error name);
```
- This function is used in declare section of the pl/sql block.

#### Note:-

Here Pragma is a compiler directive i.e. whenever we are using pragma oracle servers internally associates error no. with exception name at the time of compilation.

```
sql> begin
 insert into
 emp (empno,ename) values (null,'murali');
 end;
 /
```

```
[error: ORA-1400: Cannot insert NULL into EMPNO]
```

```
sql> declare
 a exception;
 Pragma exception_init (a,-1400);
 begin
 insert into
 emp (empno,ename) values (null,'murali');
 exception
 when a then
 dbms_output.put_line ('not to insert null values');
 end;
 /
```

Output: not to insert null values.

```
SQL> begin
 delete from dept where
 deptno=10;
 end;
 /
```

ORA- 2292 : Integrity Constraints violated - child record found

solution:-

```
SQL> declare
 a exception;
 pragma exception_init(a, -2292);
 begin
 delete from dept where deptno=10;
 exception
 when a then
 dbms_output.put_line('not to delete master records');
 end;
 /
```

output: not to delete master records-

- \*\*\* Q. write a pl/sql program to handle -2291 error no. by using exception\_init() based on emp, dept table ?.

```
SQL> declare
 a exception;
 pragma exception_init(a, -2291);
 begin
 insert into emp(empno,deptno) values(1,5);
 exception
 when a then
 dbms_output.put_line('not to insert other than
 primary key values');
 end;
 /
```

O/p:- not to insert other than primary key values.

26/7/16

7.5 U nicely w.

### raise\_application\_error()

- `raise_application_error` is a predefined procedure available in `dbms_standard package`.
- In oracle if we want to display userdefined exception messages in more descriptive form then only we are using this procedure. i.e. if we want to display user-defined exception messages as same as oracle errors displayed format then we must use `raise_application_error` procedure. This procedure is used in either in executable sections or the exception section of PL/SQL Block.

- This procedure accepts two parameters -

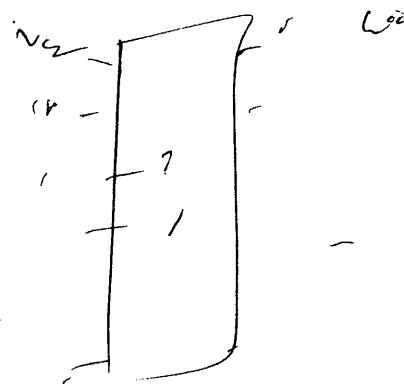
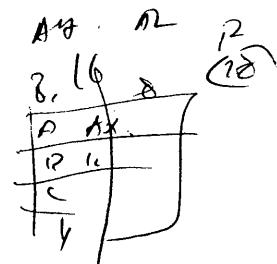
Syntax -

```
Raise_application_error (error number, Message);
 ↑ ↑
 -20,000 to -20,999 upto 512 character
```

ex:-

```
declare
 a exception;
begin
 if 'to_char(sysdate, 'DY') = 'TUE'
 then
 raise a;
 end if;
exception
 when a then
 raise_application_error (-20345, 'My exception raised on Tue');
end;
/
```

Output: ORA-20345: My exception raised on Tue.



2nd off

1st





### - Difference between dbms\_output, raise\_application\_error():-

- raise\_application\_error is an exception procedure the whenever condition is true raise this procedure and also control doesn't goes to remaining program. i.e this procedure regularly used in trigger and also this procedure stops invalid data entry based on condition.

- dbms\_output package having put\_line procedure. This procedure also displaying message. i.e. whenever condition is true it will be display message and also control goes to remaining program. That's why this procedure doesn't stops invalid data entry .

SQL> begin

  if to\_char(sysdate, 'DY') = 'WED'

  then

dbms\_output.put\_line ('My exception raised on wednesday');

dbms\_output.put\_line ('Control goes to next line');

  end if;

  end

  /

SQL> set serverpoint on;

O/P:- my exception raised on wednesday  
control goes to next line

SRI SAI SHASHI KALYAN,  
H.No. 7-1-209/1, Ground Floor,  
Sam Mira Towers, Near Satyam Bhawan,  
Amarpet, Hyderabad-500 014.

SQL> begin

  if to\_char(sysdate, 'DY') = 'WED' then

raise\_application\_error (-20123, 'my exception raised on wednesday');

    dbms\_output.put\_line ('control does not go to next line');

  end if;

  end;

  /

SQL> set serverpoint on;

SQL> /

O/P:- ORA-20123: my exception raised on wednesday.

## Error Trapping Functions :-

oracle having two predefined error trapping functions which is used to returns error numbers, error no. with error message.

Those are -

1) sqlcode

2) sqlerrm

- These functions are used in when 'others then' clause (or) used in our own exception handler .
- In oracle if we want to know the error no at runtime then only we are allowed to use sql-code function.

ex:-

```
SQL> declare
 v_sal number(10);
 begin
 select sal into v_sal from emp
 where deptno = &deptno;
 dbms_output.put_line (v_sal);
 exception
 when others then
 dbms_output.put_line (sqlcode);
 end;
 /
```

SQL> Enter value for deptno : 10  
-1422

```
SQL> /
Enter value for deptno : 'a'
-1722
```

NOTE:- Using sqlcode function we can also handle unnamed exception,

example:-

```
SQL> begin
 delete from dept where
 deptno=10;
 end;
 /
```

ORA-2292: ~~integrity constraint violated - child~~  
~~record found~~

solution:-

```
SQL> begin
 delete from dept
 where
 deptno=10;
```

SHRI GAI BHAVANI COLLEGE  
No. 7-1-209/1, Ground Floor,  
Ranjitha Towers, Nizampet, Telangana  
Ameerpet, Hyderabad 500 016

```

exception
when others then
if sqlcode = -2292 then
dbms_output.put_line ('not to delete master records');
end if;
end;
/

```

O/P: not to delete Master records.

#### NOTE:-

We are not allowed to use sqlcode, sqlerrm functions in DML statements to overcome this problem first we must declare variable declare section of PL/SQL blocks and then we must assign sqlcode, sqlerrm values into variable within exception handler and then only we are allowed to use these two variables in DML stmts.

- Q. write a PL/SQL program which is to store error number, err no. with error message of PL/SQL Block, into a table?

```

SQL> create table target (errno number(10), errmsg varchar2(100));
SQL> declare
 v_errno number(10);
 v_errmsg varchar2(100);
 v_sal number(10);
begin
 select sal into v_sal from emp;
exception
when others then
 v_errno := sqlcode ;
 v_errmsg := sqlerrm;
 insert into target values (v_errno,v_errmsg);
end;
/
SQL> select * from target;

```

| <u>Sqlcode return values</u> | <u>Meaning</u>         |
|------------------------------|------------------------|
| 0                            | no error               |
| -ve                          | oracle error           |
| 100                          | no data found          |
| 1                            | user defined exception |

SRI SAI BHAVANI COLLEGE OF ENGINEERING  
H.No. 7-1-209/1, Ground Floor,  
Ram Mirra Towers, Near Satyam Theatre,  
Ameerpet, Hyderabad-500 016.

- sqlcode functions returning numbers, whereas Sqlerrm returns error number with error messages.

SQL> declare

```
a exception;
begin
raise a;
exception
when a then
dbms_output.put_line (sqlcode);
dbms_output.put_line (sqlerrm);
end;
/
```

O/P : 1  
User defined Exception

NOTE:- we can also pass sqlcode (or) sqlcode return value in Sqlerrm function whenever we are passing like this in PL/SQL executable section ~~as~~ then only oracle server returns internal meaning of sqlcode return values.

SQL> begin

```
dbms_output.put_line (sqlerrm(sqlcode));
dbms_output.put_line (sqlerrm(100));
dbms_output.put_line (sqlerrm(1));
dbms_output.put_line (sqlerrm(-1722));
end;
/
```

Output :

```
ORA- 0000 : normal, successful completion
ORA- 01403 : no data found
ORA- 1 : User defined Exception
ORA -01722 : invalid number
```

27/17/16

SRI SAI BHAVAM  
F.I.No. 7-1-200/1, Ground Floor,  
Ram Mirra Towers, Near Satyam Building,  
Ameerpet, Hyderabad - 500 018.

## SUB-PROGRAMS

Date-31/Dec/15.

- Subprogram are named PL/SQL blocks which is used to solve particular task. All DB system having two types of subprograms These are -

- a) Procedure. [may or may not return value]
- b) Function. [must return a value]

### a) Procedure -

- procedure is a named PL/SQL blocks which is used to solve particular Task. and procedure may or may not return value.
- In Oracle whenever we are using create or replace keyword in front of the procedure. Then those procedures are automatically permanently stored in Database. That's why these procedures are also called as stored procedure.
- Generally procedures are used to improves performance of the application because procedures internally having one time compilation.

In all databases by default one time compilation programs you needs improves performance of the application.

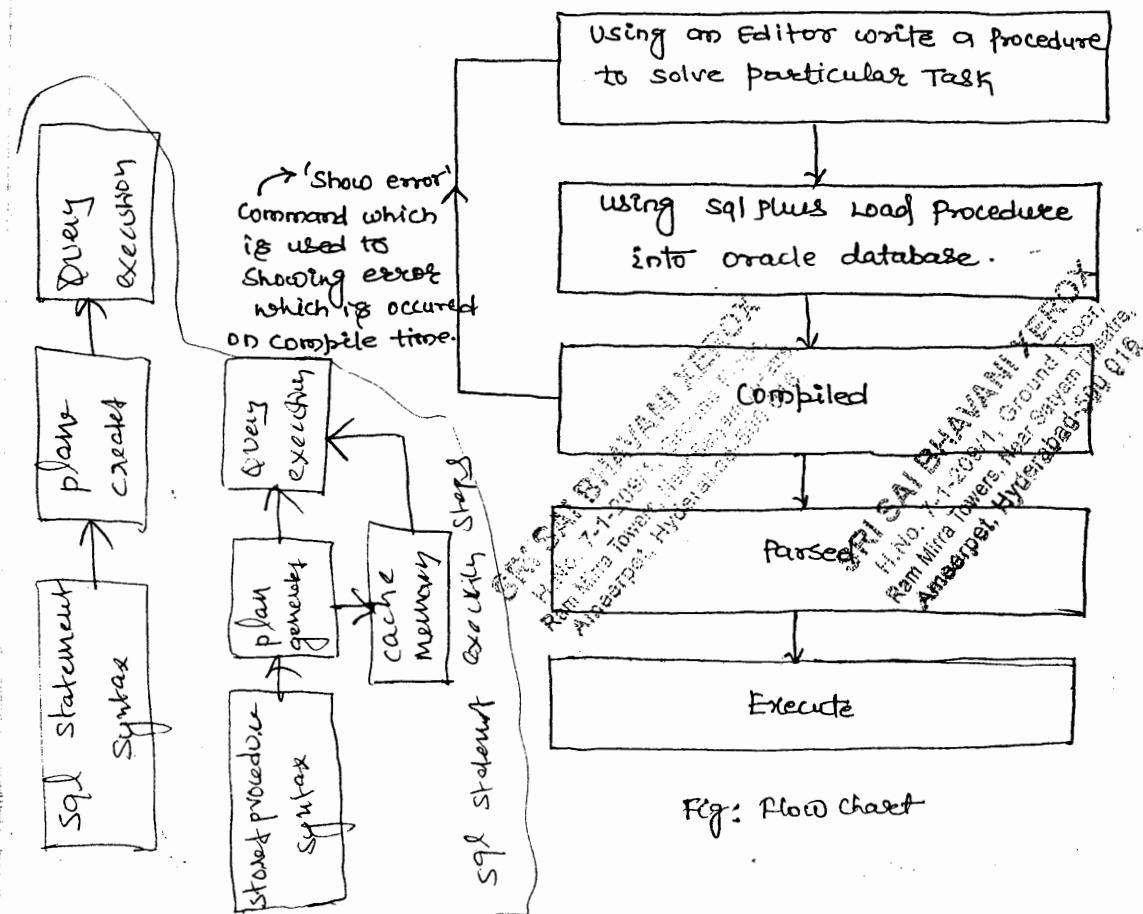


Fig: Flow chart

```
insert into test values ('india');
commit;
end;
/
```

```
SQL> begin
 insert into test values ('hyd');
 insert into test values ('mumbai');
 P1;
 rollback;
 end;
 /
output:- SQL> select * from test;
 Name
 india
```

Without using autonomous transaction :-

```
SQL> delete from test;
SQL> select * from test;
 no rows selected
SQL> drop procedure P1;
SQL> create or replace procedure P1
 as
 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;
```

O/p:-

| Name   |
|--------|
| hyd    |
| mumbai |
| india  |

SRI SAI EMART MART  
H.No. 7-1-209/1, Ground Floor,  
Ram Alpa Towers, Near Satyaniketan,  
Ameerpet, Hyderabad-500 011.

Ph: 9908195958

- In oracle whenever we are calling a procedure in main transaction it also commits and also that procedure having commit then that commit not only saves within procedure transaction but also saves all the above procedure transaction within main transaction. To overcome this problem oracle 8.1.6 introduced autonomous transaction within procedures.

### \* Autonomous transaction in Anonymous block:-

ex:-

```
SQL> conn scott/tiger
SQL> create table test (sno number(10));
```

Main Transaction -

```
SQL> insert into Test values(1);
SQL> insert into Test values(2);
SQL> select * from Test;
```

| SNO |
|-----|
| 1   |
| 2   |

Child Transaction (Autonomous Trans)

```
SQL> declare
 pragma Autonomous_Transaction;
 begin
 for i in 3..10
 loop
 insert into Test values(i);
 end loop;
 commit;
 end;
 /
```

```
SQL> select * from Test;
```

| SNO |
|-----|
| 1   |
| :   |
| 10  |

Main Transaction

```
SQL> rollback;
```

```
SQL> select * from Test;
```

| SNO |
|-----|
| 3   |
| 4   |
| :   |
| 10  |

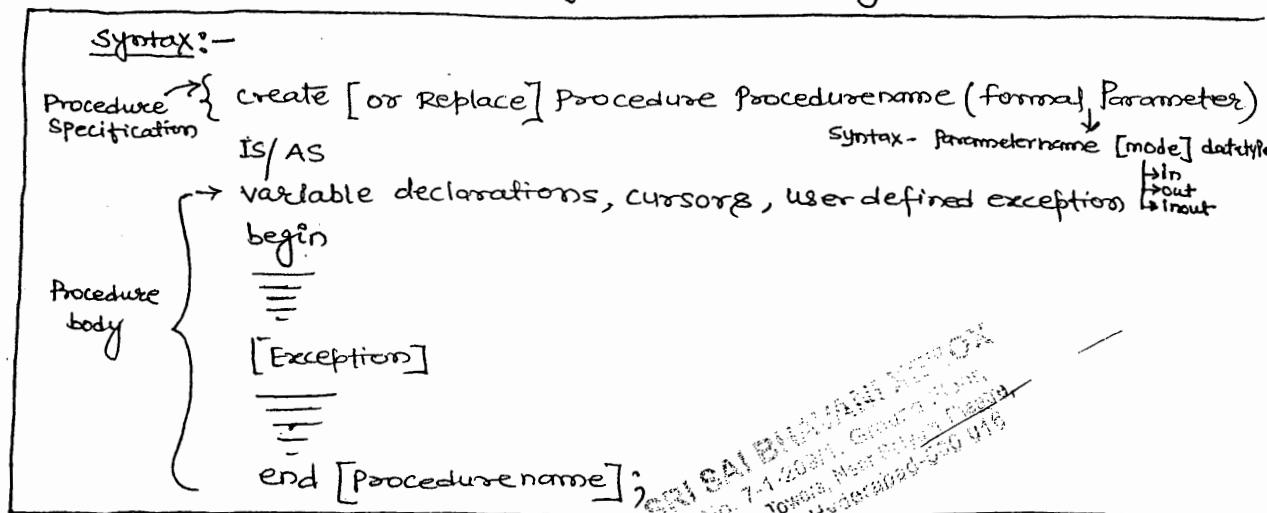
```
SQL> conn scott/tiger
```

```
SQL> select * from Test;
no rows selected
```

```
SQL> select * from Test;
```

| SNO |
|-----|
| 3   |
| 4   |
| 5   |
| 6   |
| 7   |
| 8   |
| 9   |
| 10  |

- Every procedure having two parts -
  - Procedure specification
  - Procedure body.
- In procedure specification we are specifying name of the Procedure and type of the Parameters.
- whereas in the procedure body we are solving actual task.



- to view errors:-

Syntax: SQL> show errors;

- Executing a procedure:-

Method 1 :-

Syntax:- SQL> exec Procedurename (actual Parameter);

Method 2 :- (using anonymous block)

Syntax:- SQL> begin  
Procedurename (actual Parameter);  
end;  
/

Method 3 : (using call statement)

Syntax: SQL> call  
Procedurename (actual Parameter);

Q) Write a PL/SQL stored procedure for passing emp no. as a parameter from emp table and display name of the employee and his salary

SQL> Create or replace procedure

```
P1 (P_empno number)
is
v_ename varchar2(10);
v_sal number(10);
begin
select ename, sal into
v_ename, v_sal from emp
where empno = P_empno;
dbms_output.put_line(v_ename || ' ' || v_sal);
end;
/
```

Execution - ① SQL> exec P1(7788);

O/P: SCOTT 4088

② SQL> begin P1(7788);
end;

/

O/P: SCOTT 4088

③ SQL> call P1(7788);

O/P: SCOTT 4088

- In oracle all procedure information stored under user\_procedures, user\_source data dictionaries.

- In oracle we want to view code of the procedure then we are using user\_source data dictionary.

SQL> desc user\_source;

SQL> select text from user\_source

where name = 'P1';

↑ capital letter.

SRI SAI BHAVAMI VENKOK  
Y-10, 7th floor, Ground Floor,  
Ranji Towers, Near Sanam Bhawan,  
Ameerpet, Hyderabad-500 015.

(Q) Write a PL/SQL stored procedure for passing deptno as a parameter from emp table then display employee details from emp table based on passed deptno?

```
SQL> Create or replace procedure P1 (P_deptno number)
 is
 cursor c1 is select * from emp
 where deptno = P_deptno;
 l_emp%rowtype;
 begin
 open c1;
 loop
 fetch c1 into l_i;
 exit when c1%notfound;
 dbms_output.put_line (l_i.empname || l_i.sal || l_i.deptno);
 end loop;
 close c1;
 end;
 /
```

Execution. SQL> exec P1 (10);

O/P:-

|        |      |    |
|--------|------|----|
| MILLAR | 3600 | 10 |
| KING   | 8600 | 10 |
| CLARK  | 4500 | 10 |

**MANOJ ENTERPRISES**  
Plot No: 40, Gayathri Nagar,  
Ameerpet, Hyderabad.  
Cell: 8125378496

### \* Procedure Parameter:-

- Parameter is a name which is used to pass the values into Procedure and return values from the procedure.
- Oracle procedure having two types of Parameters :-
  - ✓ 1.) Formal Parameter
  - ✓ 2.) Actual Parameter

#### 1) Formal Parameters:-

Formal Parameters specifies name of the Parameters and type of the Parameters.

Formal Parameters are defined in special Procedure Specification only.

NOTE:-

In Oracle whenever we are using cursor, procedures, functions then we are not allowed to use datatype size; in formal parameter declaration.

### Syntax :-

Parameter name [mode] datatype.

22nd

mode :- mode specifies purpose of the Parameter. Oracle procedure parameter having three types of mode. These are -

- ② in mode
  - ③ out mode
  - ④ in-out mode

a.) in mode:-

In mode is used to pass the values into procedure body.

- In oracle by default body is in mode only. in parameter behaves like a constants in procedure body.

## Syntax -

Parameter name in datatype

Q) write a pl/sql stored procedure to insert a record into dept table by using in parameter?

SQL> create or replace procedure

P\_deptno in number, P\_dname in varchar2, P\_loc in varchar2

i8

begin

insert into dept

values (P\_deptno, P\_dname, P\_loc);

```
dbms_output.put_line('your record is inserted');
```

end;

```
1 SQL> set serveroutput on;
```

execution - exec P, (1,'a','b');

Your record is inserted

SQL> select \* from dept;

## N & P A & P b> Out mode :-

- out mode is used to return value from the procedure.
- out parameter behaves like a uninitialized variable in procedure body.
- Here explicitly must specify out keyword.

### Syntax -

Parametername out datatype

e.g. create or replace procedure P,

(a in number, b out number)

i8 b  
begin  
    ~~a := a \* b;~~ ✓  
    b := a \* a; ✓  
end;  
/

~~a := a \* b;~~ X

- In oracle when procedure having out (or) in out parameters then those types of procedures are allowed to execute by using following two method:-

Method 1:- using bind variable. ✓

Method 2:- using anonymous block. ✓

### Method-1 - (using bind variable)

SQL> variable x number;

SQL> exec P1(7,:x);

SQL> print x;

O/P: x

49

GT-SAI BINNANI XEROX  
Plot No. 2007/1, Ground Floor,  
Sai Binnoo Towers, 10th Main Street,  
Amarpet, Hyderabad, 500016.

### Method 2 - (using anonymous block)

SQL> exec declare

```
z number(10);
begin
 P1(8,z);
 dbms_output.put_line(z);
end;
/
```

O/P:- 64

Q. Write a PL/SQL stored procedure for passing employee name as in parameter then return salary of the employee from emp table by using out parameter.

SQL) Create or replace procedure P<sub>i</sub>  
(P\_ename in varchar2, P\_sal out number)

```
is
begin
select sal into P_sal From emp
where ename = P_ename;
end;
/
```

Execution - I By using bind variable -

```
SQL> variable a number;
SQL> exec P_i('SMITH', :a);
SQL> print a;
```

O/P :-      A  
                4100

② By using anonymous block

```
SQL> declare
x number(10);
begin
P_i ('FORD', x);
dbms_output.put_line(x);
end;
/
```

O/P : 3100

Q. write PL/SQL stored procedure for passing deptno as in parameter then returns number of employees number from emp table by using out parameter.

SQL) Create or replace procedure P<sub>i</sub> (P\_deptno in number,  
P\_count out number)

```
is
begin
Select count(*) into P_Count from emp
where deptno = P_deptno;
end;
/
```

execute :-

SQL> variable a number;  
SQL> exec P1(10,:a);

'PL/SQL procedure successfully completed.

SQL> print a;

O/P: A  
3.

29/7/16

Q) write a PL/SQL stored procedure for passing deptno. as in Parameter then return dname, loc from dept table by using out Parameter ?

SQL> create or replace Procedure P1  
(P\_deptno in number, P\_dname out varchar2, P\_loc out varchar)  
is  
begin  
select dname, loc into P\_dname, P\_loc from dept  
where deptno = P\_deptno;  
end;  
/

Execution -

Method ① [using bind variable]

SQL> variable a varchar2(10);  
SQL> variable b varchar2(10);  
SQL> exec P1(20,:a,:b);

PL/SQL procedure successfully completed.

SQL> print a b;

O/P:- A ACCOUNTING B NEWYORK

Method ② [using anonymous block]

SQL> declare  
a varchar2(10);  
b varchar2(10);  
begin  
P1(20,a,b);  
dbms\_output.put\_line(a||' '||b);  
end;  
/

O/P:- SALES CHICAGO

## nocopy compiler hint

(08)

### Pass by value, Pass by reference :-

- whenever we are using Modular programming all languages supports two types of Passing Parameter Mechanism -
  - a) Pass by value
  - b) Pass by reference.
- These two Passing Parameter Mechanism specifies whenever we are modifying formal parameter then actual parameter are effected or not in Pass by value Method actual values does not change when we are modifying formal parameter also because in Pass by value Method internally copy of the values are Pass into calling Programs.  
If we want to effect actual parameter based on the formal parameters modification then we are using Pass by reference Method.
- oracle also supports these two Passing Parameters Mechanism internally when we are using subprogram parameters.  
In oracle by default all in parameters internally uses Pass by reference whereas all out parameters internally uses pass by value method.
- whenever we are returning large amount of data by using out parameter then oracle server degrades performance of the procedure because whenever we are using out parameters again copy of the values are created. To overcome this problem for improve performance of the oracle gi introduced "nocopy" compiler hint into the "out" parameter.

#### Syntax -

Parametername out nocopy datatype

ex:- say create or replace procedure P,  
(P\_ename in varchar2, P\_Sal out nocopy number)

is

begin

Select sal into P\_Sal from emp

where ename = P\_ename;

end;

/

BAE  
2017/18

### c) in out mode

This mode is used to pass the value into procedure and also return values from procedure.

This mode behaves like a ~~college~~ constant, initialized variable in procedure body. Here also explicitly we must specify in out keyword.

Syntax :-

Parametername in out datatype

```
SQL> create or replace procedure
 P1(a in out number)
 is
 begin
 a := a*a ;
 end;
```

!

Execution :- using bind variable -

```
SQL> variable z number;
```

```
SQL> exec :z := 8;
```

```
SQL> exec P1(:z);
```

```
SQL> print z;
```

O/P :- 
$$\frac{z}{64}$$

2) Method - using anonymous block -

```
SQL> declare
 x number(10) := &x;
 begin
 P1(x);
 dbms_output.Put_line(x);
 end;
```

!

O/P: Enter the value for x: 8

64

## In Parameter execution Method:-

oracle procedure in parameter having three types of execution method these are -

- a) Positional notations
- \* b) named notations ( $\Rightarrow$ )
- c) Mixed notations.

### ex:- (a) Positional notation

SQL> exec P<sub>i</sub>(1, 'a', 'b');  
error

SQL> exec P<sub>i</sub>('a', 1, 'b');  
successfully executed /

name, no, loc,

### (b) named notation -

SQL> exec P<sub>i</sub>(P\_name  $\Rightarrow$  'a', P\_loc  $\Rightarrow$  'b', P\_no  $\Rightarrow$  10);  
successfully executed

(c) Mixed notation - It is combination of Positional, named notation. Here after positional there can be all named notation, but after named there can't be positional.

eg - exec P<sub>i</sub>(10, P\_loc  $\Rightarrow$  'b', P\_name  $\Rightarrow$  'a');  
successfully executed .

exec P<sub>i</sub>( P\_loc  $\Rightarrow$  'b', P\_name  $\Rightarrow$  'a', 10);

error [after named notation can't be used  
positional notation].

NOTE:- we can also pass default values in procedure, function parameter by using IN mode.

If we want to pass default value then we are using default (or) := operator.

Syntax :- Parametername in datatype [default (or) :=]  
defaultvalue;]

Q. Write a PL/SQL stored procedure by using in out Parameter for passing employee number then return salary of the employee from emp table ?

SQL> create or replace procedure P1(P\_x in out number)

is

begin

select sal into P\_x from emp

where empno = P\_x;

end;

/

Execution- By using Bind variable -

SQL> Variable z number;

SQL> exec :z:=7369

SQL> exec P1(z);

SQL> print z;

            
          z  
        4200

=====

default example-

SQL> create or replace procedure

P1(P\_deptno in number; P\_dname in Varchar2, P\_loc in Varchar2

default 'hyd')

is

begin

insert into dept

values( P\_deptno, P\_dname, P\_loc);

end;

/

Execution:-

SQL> exec P1(1,'a');

SQL> select \* from dept;

## \* \* \* autonomous transactions:-

- autonomous transactions are independent transaction used in anonymous blocks, procedures, functions, triggers.
- generally we are defining autonomous procedure in child procedure.
- whenever we are calling autonomous procedure in main transaction and also main transaction TCL commands never affected on autonomous TCL commands & procedure, because these are independence procedure.
- If we want to procedure autonomous then we are using autonomous\_transaction Pragma, commit i.e. in declare section of the procedure we are defining autonomous transaction pragma and also we must use commit in procedure coding.

Syntax:-

Pragma autonomous\_transaction;

Syntax - (Procedure)

Create or replace procedure ProcedureName (Formal Parameters);

is/as

Pragma autonomous\_transaction;

begin

....

....

commit

CGPSA1 EVA44444444  
H.No. 7-12081, Ground floor,  
Ran Mirra Towers, Near Surya Plaza,  
Ameerpet, Hyderabad-500 016.

[Exception]

....

end [ProcedureName];

e.g:- SQL create table test (name varchar2(10));

With using autonomous Transaction

SQL create or replace procedure P,

is

Pragma autonomous\_transaction;

begin insert into test values ('india');

commit;

end //

## accessible by clause :-

- oracle 12c introduced accessible by clause stored procedure is in there. This clause is used in procedure specification.
- accessible by clause provides security of the procedure.

### Syntax:-

```
create or replace procedure
ProcedureName (formal parameters)
accessible by (anotherprocedurename1, anotherprocedurename2,
 anotherprocedurename3)
is/as
. . .
. . .
begin
. . .
[Exception]
. . .
end [ProcedureName];
```

- accessible by procedure are allowed to call within specific procedure specifying accessible by clause.

```
SQL> create or replace procedure P1
 accessible by (P2)
 is
 begin
 dbms_output.put_line ('my proc');
 end P1;
 /
```

```
SQL> create or replace procedure P2
 is
 begin
 P1;
 end;
 /
```

O/P :- error

SQL> create or replace procedure

```
P3
is
begin
P1;
end P3;
/
```

SQL> exec P3;

O/P: my proc

~~use~~ ~~where~~ authid current\_user :-

When a Procedure have a authid current\_user clause then those procedures are allowed to execute only owner of the procedure. These procedures are not allowed to executes by another user if any user giving permission also.

Generally whenever we are reading data from table and performing some DML operations then only data security folks, view developer uses this clause in procedures.

This clause are used in procedures specification only.

Syntax -

```
create or replace procedure procedurername(formal Parameters)
authid current_user
is /as

begin

[exception]

end [procedurername];
```

SRISAI BHAVANI INSTITUTE OF  
H.O. 7-1203/H, Ground Floor,  
Ram Mira Towers, Near Satya Narayan Temple,  
Ameerpet, Hyderabad - 500 013.

NOTE:- In oracle if we want to execute Procedures, functions, predefined Packages into another user then we must use execute object privilege.

## Cursors Used in Functions :-

- we can also use cursor within userdefine function i.e. In oracle if you want to implement user define aggregate function those function returns multiple values. Then only allowed to use cursors within user-defined cursor and also these user-defined function within statement same like aggregate functions.

Q. write a PL/SQL userdefine function which behaves like a predefine aggregate function for passing deptno, then return employee name for each department by using emp table ?

```
SQL> create or replace function
 f1 (P_deptno number)
 return varchar2
 l8
 a varchar2 (200);
 cursor C is select ename from emp where deptno = P_deptno;
 begin
 for i in C
 loop
 a := a || i.ename;
 end loop;
 return a;
 end;
 /
```

### Execution:-

```
SQL> select deptno, f1(deptno)
 from emp group by deptno;
```

Note:- oracle 10g introduced `wm_concat()` predefine aggregate function which returns multiple values group by when we are using this function within group by clause.

This function accepts all datatype, columns.

Syntax:-

```
wm_concat (columnname)
```

SQL> select deptno, wfm\_concat(ename)  
From emp group by deptno;

SQL> select job, wfm\_concat(ename) from emp group by job;

O/P:-

| JOB       | WM_CONCAT (ENAME)           |
|-----------|-----------------------------|
| ANALYST   | SCOTT, FORD                 |
| CLERK     | SMITH, JAMES, ADAMS, MILLER |
| MANAGER   | JONES, CLARK, BLAKE         |
| PRESIDENT | KING                        |
| SALESMAN  | ALLEN, WARD, TURNER, MARTIN |

NOTE:-

We can also use wfm\_concat() function without using group by clause in this case this function gets no. of values from columns and returns into no. of columns with comma separated.

Ex:- SQL> select wfm\_concat(ename) from emp;

O/P:-

wfm\_concat(ename)

SMITH, ALLEN, WARD, JONES, MARTIN, BLAKE, CLARK,....

- In oracle all stored functions information stored under user\_procedures user\_source data dictionaries. If we want to view code of the function then we are using user\_source data dictionaries.

Ex:-

SQL> desc user\_source;

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

We can also drop function by using

drop function function\_name;

Syntax :-

```
grant execute on procedurename to username1,username2,...;
```

```
SQL> conn scott/tiger;
```

```
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 mureli;
```

```
SQL> conn mureli/mureli;
```

```
SQL> set serveroutput on;
SQL> exec scott.P1(7566);
error: table or view does not
exist.
```

Handled (or) unhandled exception in procedures:-

In oracle whenever we are calling inner procedure into outer procedure then we must handle inner procedure exception within the inner procedure only otherwise oracle server ~~exception~~ executes after procedure default Exception handler.

inner procedure-

```
SQL> create or replace procedure P1
(a in number, b in number)
is
begin
dbms_output.put_line (a/b);
exception
when zero_divide then
dbms_out.put_line ('b cannot be zero');
end;
/

```

### Outer procedure -

```

SQL> create or replace procedure P2
is
begin
 f1(8,0);
exception
 when others then
 dbms_output.put_line ('any error occurred');
end;
/

```

### Execution:

SQL> exec P2;

O/P :- b cannot be zero.

We can also drop a procedure using -

drop procedure procedutename;

3/8/16

### b-> Function:-

- Function is a named PL/SQL block which is used to solve particular task and also function must ~~must~~ return a value.
- Function also having two parts :-
  - a) function specification
  - b) function body
- In function specification we are specifying name of the function and type of the parameters.
- Whereas in function body we are solving actual task.

#### Syntax:-

```

create [or Replace] function functionname(formal parameter)
return datatype
is/as
→ variable declarations, cursors, user defined exceptions;
begin
 return expression;
[Exception]
end [functionname];

```

## executing a function:-

- Method-1 :- (using select statement)

Syntax -

```
SQL> select functionname(actual parameters) from dual;
```

NOTE:- In oracle when a function having all in parameters or function doesn't have parameters then those functions only allowed to execute by using select statement.

- Method-2 :- (using anonymous block)

Syntax -

```
SQL> begin
 variable name := Functionname(actual parameters);
 end;
 /
```

Q:- SQL> create or replace function

```
f1(a varchar2)
return varchar2
is
begin
 return a;
end;
/
```

### execution -

- Method-1 :- (using select statement)

```
SQL> select f1('hi') from dual;
```

O/P :- hi

### Method-2 using anonymous block

```
SQL> declare
z varchar2(10);
begin
 z := f1('welcome');
 dbms_out.put_line(z);
end;
/
```

O/p : welcome

Q) write a pl/sql stored function for passing no. as a parameter  
then return a message either even or odd based on passed number.

SQL> create or replace function

```
f1(a number)
return varchar2
is
begin
if mod(a,2)=0 then
return 'even';
else
return 'odd';
end if;
end;
```

/

Execution:- ① using select statement

SQL> select f1(8) from dual;

O/P:- f1(8)  
even

② using anonymous block

```
SQL> declare
x varchar2(10);
begin
x:=f1(7);
dbms_output.put_line(x);
end;
```

/

O/P:- odd

③ using bind variable-

SQL> variable z varchar2(10);

```
SQL> begin
:z := f1(4);
end;
```

/

SQL> print z;

O/P:- even

④ exec  
dbms\_output.put\_line (f1(g));

O/P :- Odd

⑤ begin  
dbms\_output.put\_line (f1(g));  
end;  
/

O/P :- Odd

NOTE:-

In oracle we can also use userdefine functions in DML statements

ex:- SQL> create table test (msg varchar2(10));

SQL> insert into test values (f1(4));  
1 row created

SQL> select \* from test;

O/P:- msg  
even

NOTE:- In oracle we can also use predefined function within user defined function and also executes these userdefined function within same table or different table.

ex:- SQL> create or replace function f1  
return number  
is  
v\_sal number (10);  
begin  
select max (sal) into v\_sal from emp;  
return v\_sal;  
end;  
/

execution. SQL> select f1 from dual;  
O/P: 8800

execution. SQL> select ename, sal, f1 from emp;

Q. write a pl/sql stored function for passing employee name in a parameter then return job of the employee from emp table based on passed emp name ?

SQL> create or replace function

f\_1(p\_ename varchar2)

return varchar2;

is

v\_job varchar2(10);

begin

select job into v\_job from emp

where ename = p\_ename;

return v\_job;

exception

when no\_data\_found then

return 'your ~~the~~ employee name not exists';

end;

/

execution- By using select statement -

SQL> select f\_1('SMITH') from dual;

O/P :- CLERK

By using anonymous block -

SQL> declare

x varchar2(10);

begin

x := f\_1('abc');

dbms\_output.put\_line(x);

end;

/

O/P : your employee name not exists  
error.

8/8/16

A

- (Q) Write a PL/SQL stored function for passing emp. number as a parameter from emp table, return 1 if the emp. salary is more than the avg salary of there department, otherwise return 0. return 1.

SQL> create or replace function

f1 (P\_empno number)

return number

is/as → Keyword

v\_sal number(10);

v\_avgSal number(10);

v\_deptno number(10);

begin

select sal, deptno. into v\_sal, v\_deptno from emp

where empno = P\_empno;

select avg(sal) into v\_avgSal from emp

where deptno = v\_deptno;

if v\_sal > v\_avgSal then

return 1; ↗ exception

else ↗ which no data found then

return 0; ↗

end if; ↗

end; ↗

/ ↗

Execution: SQL> select f1(7902) from dual;

O/P :- 1

- Q. write a query to display the employee who are getting more salary then the average salary to the dept no from dept table by using above function.

SQL> select ename, sal from emp

where f1(empno) = 1;

(Q/P)

ENAME

BLAKE

SCOTT

KING

F1 (EMPNO)

1

1

1

Q) write a PL/SQL stored function for passing empno, date as parameters then return no. of years that employee working from emp table based on passed date?

SQL> create or replace function

f1 (P\_empno number, P\_date date)

return number

is

x number(10);

begin

select

(date1, date2)

months\_between(P\_date, hiredate)/12 into x from emp  
where empno = P\_empno;

return round(x);

end;

Here using round keyword

O/P: 10.802

round → 10 ✓

Execution-

SQL> select f1(7566, sysdate) from dual;

O/P:- 35

SQL> select empno, ename, hiredate f1(empno, sysdate) || ' ' ||  
'years' . "Exp" from emp where empno = 7566

O/P:

| EMPNO | ENAME | HIREDATE  | EXP      |
|-------|-------|-----------|----------|
| 7566  | JONES | 02-APR-81 | 35 Years |

concatenation  
operator

NOTE:-

Oracle 11g introduced named, mixed notation. When a subprogram is executed by using select statement.

ex:- (oracle 11g onwards working)

SQL> select empno, ename, hiredate, f1(P\_empno => empno,

P\_date => sysdate) || ' ' || "years" . "Exp" from emp.

where empno = 7566;

outmode:- this mode is used return more than one value

from function that is we can also used out parameter, if functions but we are not allowed to execute these functions

by using select statements.

- Q. write a pl/sql stored procedure for passing employee no's  
 then return tax of the employee based on following cond:-
- (i) if annual More than 10,000 then tax > 10% of annual.
  - (ii) if annual More than 15,000 then tax > 20% of annual.
  - (iii) if annual More than 20,000 then tax > 30% of annual  
 else tax > 0.

Sol:-

SQL

```
- Create or replace function
 tax(P_empno number)
 return number
 is
 v_sal number(10);
 annual number(10);
 z number(10);
 begin
 Select sal into v_sal from emp
 where emp_no = P_empno;
 annual := v_sal * 12;
 if annual > 10000 and annual <= 15000
 then
 z := annual * 0.1;
 elsif annual > 15000 and annual <= 20000
 then
 z := annual * 0.2;
 elsif annual > 20000 then
 z := annual * 0.3;
 else
 z := 0;
 end if;
 return z;
 end;
 /
```

Execution:- SQL> select tax(7902) from dual;

## DML statements are used in function:-

& write a PL/SQL stored function for passing empno. as a parameter from emp table then delete that record & and also returns deleted no. of records number.

```
SQL> create or replace function
f1 (P_empho number)
return number
is
v_count number(10);
begin
delete from emp where
empno = P_empho;
v_count := sql%rowcount;
return v_count;
end;
/
```

O/P: Function created.

- In oracle we can also use DML statements in user define functions but those functions are not allowed to ~~not~~ execute select statement but allowed to execute by using anonymous block.

### Execution:-

Method 1 - (using select statement)

```
SQL> select f1(1) from dual;
```

Error:- cannot perform a DML operation inside a query.

Method 2 - (using anonymous block)

```
SQL> declare
a number(10);
begin
a := f1(1);
dbms_output.put_line(a);
end;
/
```

Output: 0

### NOTE:-

In oracle we can also execute DML function within select statement oracle 8.0.6 onwards i.e. whenever we are using autonomous transaction function then only we're allowed to execute those user-defined function in select statements. In this case we must declare autonomous transaction program and also used commit in function body.

```
SQL> create or replace function
f1 (P_emphno number)
return number
is
v_count number(10);
pragma autonomous_transaction;
begin
delete from emp where
empno = P_emphno;
v_count := sql%rowcount;
commit;
return v_count;
end;
/
```

Execution:- SQL> select f1(1) from dual;  
O/P : 0

SRI SAI BHAKTI AC. AKA.  
H.NO. 7-1-208/1, Ground Floor,  
Ram Niwas Towers, Near Saifee Hospital,  
Ameerpet, Hyderabad - 500 011  
Ph: 9908195958

### Out mode:-

We can also use out mode parameter in function, but these functions are not allowed to execute by using select statement. If we want to return more no. of values from a function then only we're allowed to use out parameters. Here also out parameters behaves like a uninitialized variable.

- Q) write a PL/SQL stored function for passing deptno. as in parameter from dept table then return dname, location from dept table by using out parameters?

```
SQL> create or replace function
f1 (P_deptno.number, P_dname out varchar2, P_loc out varchar2)
return varchar2
is
begin
```

```
SQL> Select dname, loc into p_dname, p_loc from dept
where deptno = p_deptno;
return p_dname;
end;
/
```

In oracle when a function have a out or in parameter, those function also allowed to execute by following two methods—

- 1> using bind variables ✓
- 2> using anonymous block. ✓

executing :-

1> using bind variable:-

```
SQL> Variable a Varchar2(10);
SQL> Variable b Varchar2(10);
SQL> Variable c Varchar2(10);
SQL> begin
:a := f1(10,:b,:c);
end;
/
```

```
SQL> print b,c;
```

O/P :-      B  
                Accounting      C  
                                  New York

when to use procedures, when to use functions—

- In oracle when application request to return multiple value then we must use procedures by using out parameter.
- Whenever an application requires DML statements then also must use procedures.
- whenever application request to execute by using select statement then must use function and also if we want to calculate value then also must use functions.

8/8/16

## PACKAGES

18-Jan-16

- Package is a database objects which encapsulate global variables, constraints, cursor, procedure, functions into single unit.
- generally packages also used to improves performance of the applications because when-ever we are calling packages subprograms first time then automatically total package load into Memory area. whenever we are calling subsequent procedures and functions then oracle server call those subprograms from memory area ~~which~~. This process automatically Reduces Disk I/O, that's why packages improves performance of the applications.
- Package also having two parts -
  - Package specification (declaration)
  - Package body (Implementing)
- In package specification we are declaring global variable, constraints, types, cursor, procedures, functions, where as package body we are implementing procedures and functions.
- In oracle by default package specification objects are public whereas package body objects are private.

package declaration parameters

- (1) does not accept parameters
- (2) does not be nested
- (3) does not invoked directly

### Syntax - (Package specification)

create or replace package packagename

is/as → keywords

- global variable declaration, constraint declarations;
- cursor declaration, types declaration, procedure declaration;
- function declaration;

end;

### Syntax - (Package body) :-

create or replace package body packagename  
is/as

Procedures implementation;

Function implementation;

end [Packagename];

## executing package SubPrograms:-

### 1. calling package procedure —

#### method - 1

Syntax :-

SQL> exec

packagename. Procedure name (actual Parameters);

#### method 2 ~~&~~ using anonymous blocks

Syntax :-

SQL> begin

packagename. Procedure name (actual Parameters);

end ;

/

### 2) calling package functions —

#### method I (using select statement)

Syntax :-

SQL> select

packagename. function name (actual Parameters)

from dual ;

#### method 2 (using anonymous blocks)

Syntax :-

SQL> begin

varname := packagename. function name (actual  
parameters);

end ;

/

↳ advantage of package

↳ disadvantage of package

NOTE:-

→ generally Packages doesn't accept parameters and also Packages  
can't nested and also can't invoked directly.

O/P: After deleting last two elements -

10  
20  
30  
40

After deleting second element

10  
30  
40

Q) write a PL/SQL program which is used to transfer all employee name from emp table into nested table and also display content from nested table ?

```
declare
 type t1 is table of varchar2(10);
 v_t t1 := t1();
 cursor c1 is select ename from emp;
 n number(10) := 1;
begin
 for i in c1
 loop
 v_t.extend;
 v_t(n) := i.ename;
 n := n + 1;
 end loop;
 for i in v_t.first .. v_t.last
 loop
 dbms_output.put_line(v_t(i));
 end loop;
end;
```

- oracle 8i introduced bulk collect clause which is used to transfer data from table into collection.
- whenever we are use bulk collect clause we are not allowed to use extend collector method in nested table (explicitly)

```
SQL> 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;
/
```

\* Varray :-

oracle 8.0 introduced varray this is a bounded table which is used to store upto 2GB data Here also, always indexes are starts with 1, and also these indexes are consecutive.  
Before we are storing data in varray we must to use constructor.

This is an user-defined type so, we are creating a two-step process, i.e. first we are creating type then only we are only creating variable from that type.

Syntax :-

- ① TYPE Typename IS varray (maxsize) OF datatype (size);
- ② VariableName Typename := Typename();

SQL> declare

```
type t1 is varray (10) of varchar2(10);
v_t t1 := t1('a','b','c','d');
begin
dbms_output.put_line(v_t.limit);
dbms_output.put_line(v_t.count);
dbms_output.put_line(v_t.first);
dbms_output.put_line(v_t.last);
dbms_output.put_line(v_t.prior(3));
dbms_output.put_line(v_t.next(3));
v_t.extend(4,3);
v_t.trim;
dbms_output.put_line(v_t(i));
end loop;
v_t.delete;
dbms_output.put_line(v_t.count);
end;
/
```

```

SQL> begin
 p23.g := 7;
 end;
 /
SQL> begin
 dbms_output.put_line (p23.g);
 end;
 /
output: 50

```

- State of the cursor:-

```

SQL> create or replace package p24
 is
 cursor c1 is select * from emp;
 pragma serially_reusable;
 end;
 /

```

```

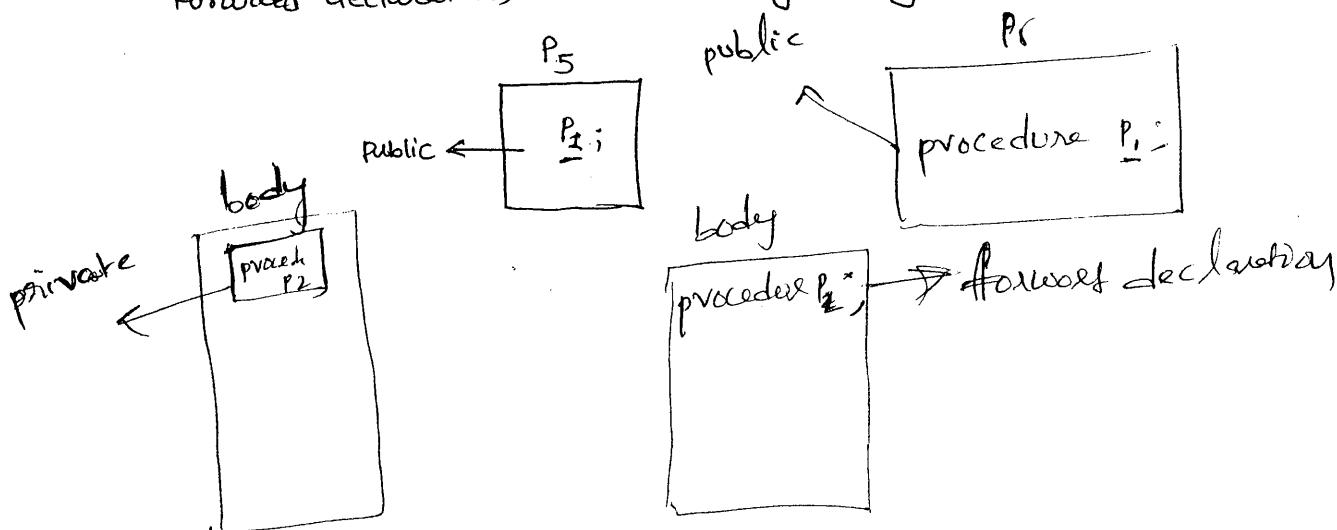
SQL> begin
 open p24.c1;
 end;
 /
SQL> begin
 open p24.c1;
 end;
 /

```

~~vv  
Tmt~~

\* Forward declaration -

→ In oracle declaring Procedures within Package body is called forward declaration generally before we're calling private procedures into public procedure first we must implements private procedure within package body otherwise use a forward declaration within package body.



Ex:

ex:- sql> Create or replace package PZ5

is

procedure P1;  
end;  
/

sql> Create or replace package body

PZ5

is

procedure P2;

Procedure P1,

is

begin

P2;

end P1;

Procedure P2

is

begin

dbms\_output.Put\_line('Private Proc');

end P2;

end;

/

declaration procedure is  
present in package body then  
it is called Forward declaration

execution. sql> exec PZ5.P1;

O/P :- Private Proc

### Overloading Procedures

overload is refers to same name can be used for different purpose.  
In oracle we can also implement overloading procedure through  
package. overloading procedure having same name with  
diff type or diff no. of parameters.

Ex:

sql> Create or replace package PJ1

is

procedure P1(a number, b number);

procedure P1(x number, y number);

end;

/

Same name

Different parameters

overloading procedures

→ In oracle overriding procedures having Some no. of parameter with same type than those types of procedures allowed to execute by using name masking only.

SQL> create or replace package P1  
is

procedure P1;

procedure P2;

end;

/

SQL> create or replace package body P1  
is

procedure P1

is

begin

dbms\_output.put\_line ('First Proc');

end P1;

procedure P2

is

begin

dbms\_output.put\_line ('Second Proc');

end P2;

end;

/

Execution - SQL> set serveroutput on;

SQL> exec P1.P1;

O/P :- First proc

~~global variables:-~~

In oracle we are declaring global variables in Package specification only.

SQL> create or replace package P22

is/~~as~~ → keywords

g number (10) := 800;

procedure P1;

function f1(a number)

return number;

end;

/

```

SQL> create or replace package body PZ2
is
procedure P1
is
x number(10);
begin
x := g/2; x := g*2;
dbms_output.put_line(x);
end P1;
function f1(a number) return number
is
begin
return a*g;
end f1;
end;
/

```

Execution - SQL> exec PZ2.P1;

O/P : 400

SQL> select PZ2.f1(3) from dual;

O/P : 2400

\* \* \* State of the global variable —

In oracle if we want to maintain state of the global variable of state of the cursor then we must use serially-reusable Pragma in packages.

Syntax -

pragma serially\_reusable;

Ex:-

```

SQL> create or replace package PZ3
is
g number(10) := 50;
pragma serially_reusable;
end;
/

```

O/P : Package created

```

SQL> create or replace package body PJ1
is
procedure P1(a number, b number)
is
c number(10);
begin
c := a+b;
dbms_output.Put_line(c);
end P1;
procedure P1(x number, y number)
is
z number(10);
begin
z := x-y;
dbms_output.Put_line(z);
end P1;
end;
/

```

Execution:- SQL> exec PJ1.P1(9,4);

error: too many declarations of "P1" match this call.

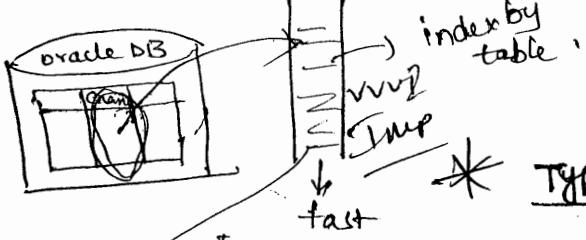
NOTE:-

In Oracle whenever overloading procedure having same no. of Params. and also same type then those type of procedures are allow to executes by using named notations only.

Execution. SQL> exec PJ1.P1(a=>9, b=>4);  
o/p : 13

SQL> exec PJ1.P1(x=>10, y=>8);  
o/p : 2

SRI SAI BHAVANI XEROX  
H.No. 7-1-209/1, Ground Floor,  
Ram Mitra Towers, Near Siddhartha Hospital,  
Ameerpet, Hyderabad - 500 013  
Ph: 9908195958



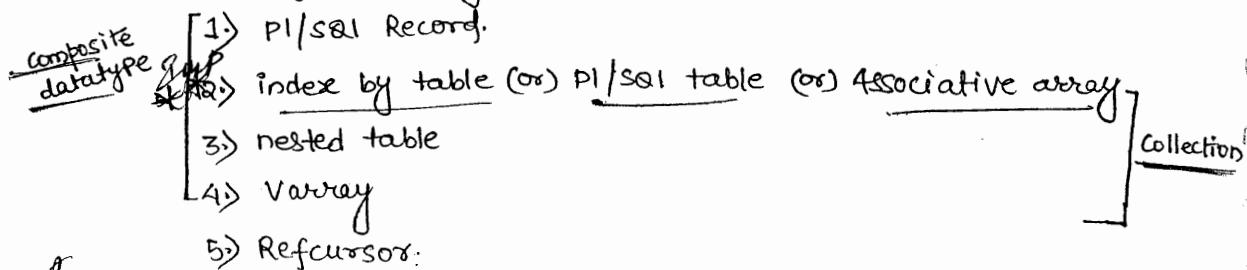
HTM CSJ

## \* Types used in Packages. \*

20-Jan-16.

- In oracle we are creating our own datatype by using type keyword.
- In oracle we are creating user-defined types in two step process i.e. RAM memory first we are creating our own userdefined datatype from appropriate area. Syntax then only we are creating a variable from that user-defined type.

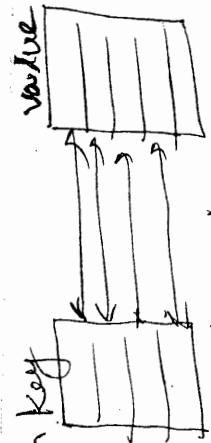
- PL/SQL having Following userdefined types -



10/8/16  
VVVVV  
Just  
dotted

### index by table (or) PL/SQL table (or) Associative array :-

- Index by table is an un-define type which is used to store no. of date items in single unit.
- Basically Index by table is an un-bound table.
- Basically Index by table having key, value pairs i.e. here value field stores actual data whereas key field stores indexes.
- These indexes are either integer (or) characters. These integers are positive, negative numbers. This key behaves like a primary key.
- Generally, index by table are used to improve performance of the applications because these tables are automatically stored in RAM memory area that's why these tables are also called as memory tables.



- Index by table having exist, first, last, prior, next, count, delete (index), delete (one index, another index), delete collection methods. Using this collection method we are operating data within index by table.
- For improve performance of the index by table oracle provided Special datatype Binary-integer for the key field.
- This tables are also called Memory table.

- This is an user defined type so, we are creating in two step process.  
i.e. first we are creating type then only we are creating a variable from that type.

~~Exmp~~

~~syntax:~~

- 1) TYPE Typename . is table of datatype (size) index by binary\_integer;
- 2) variable name Typename;

ex:- 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));
```

```
dbms_output.put_line(v_t.first);
```

```
dbms_output.put_line(v_t.last);
```

```
dbms_output.put_line(v_t(4));
```

```
dbms_output.put_line(v_t.next(3));
```

```
dbms_output.put_line(v_t.count);
```

```
v_t.delete(1,3);
```

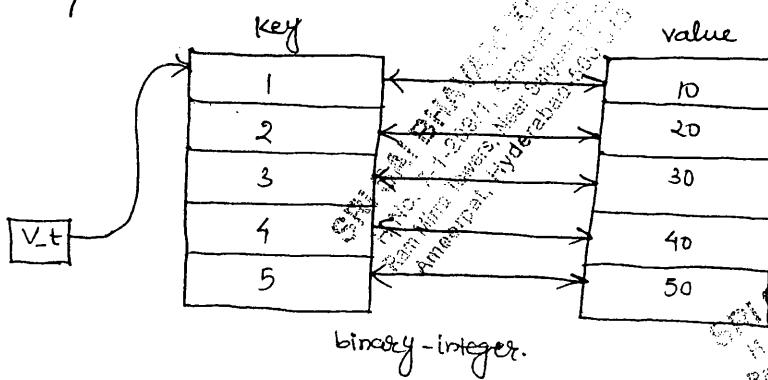
```
dbms_output.put_line(v_t.count);
```

```
v_t.delete;
```

```
dbms_output.put_line(v_t.count);
```

```
end;
```

```
/
```



SPIKA BIKE RENTAL SERVICE  
Ranbir Towers, 1st Floor,  
Ameerpet, Hyderabad - 500 016.

Ques

Q. Write a PL/SQL program to transfer all employee name from emp table into index by table and also display content from index by table?

SQL> declare

```
type t1 is table of varchar2(10) index by binary_integer;
```

```
v_t t1;
```

```
cursor c1 is select ename from emp;
```

```
m number(10):=1;
```

```
begin
```

```
open c1;
```

```
loop
```

```
fetch c1 into v_t(m);
```

```
exit when c1%not found;
```

```
m := m+1;
```

```
end loop;
```

```
close c1;
```

```
for i in v_t first..v_t.last
```

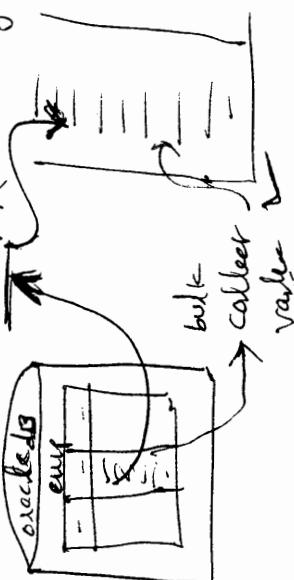
```
loop
```

```
dbms_output.put_line(v_t(i));
```

```
end loop;
```

```
end;
```

```
/
```



Q1.8/16

- whenever resource table having large amount of data and also when we are transfer data into collection by using cursor then those type of applications degrades performance.

To overcome this problem oracle 8i introduced bulk collect clause.

- cursor degrades performance because cursor internally uses record by record process. whereas bulk collect clause improves performance because whenever we are using bulk collect clause oracle server selects column data at a time on storing into collection.

Syntax:-

```
Select * bulk collect into collection varname
From tablename where condition;
```

This clause is used in executable sections of PL/SQL Block.

```

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) Write a PL/SQL program which stores next 10 days into index by table and also display content of index by table.

```

SQL> 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;
/

```

O/P :-  
 12-8-16  
 13-8-16  
 14-8-16

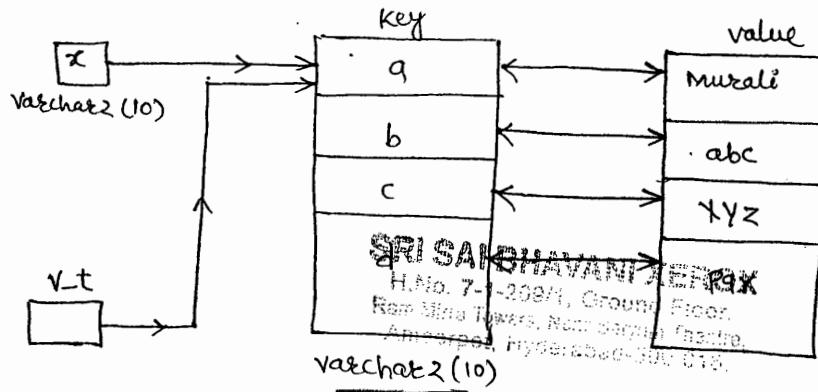
- Q) write a PL/SQL program to transfer all employee's joining date from emp table into index by table and also display content from index by table ?

```

SQL> 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;
/

```

→ Index by table we can also character data type in key field but in this case we are not allowed to use for loops to display all data index by table because in for loop cursor variable behaviour like a integer variable.



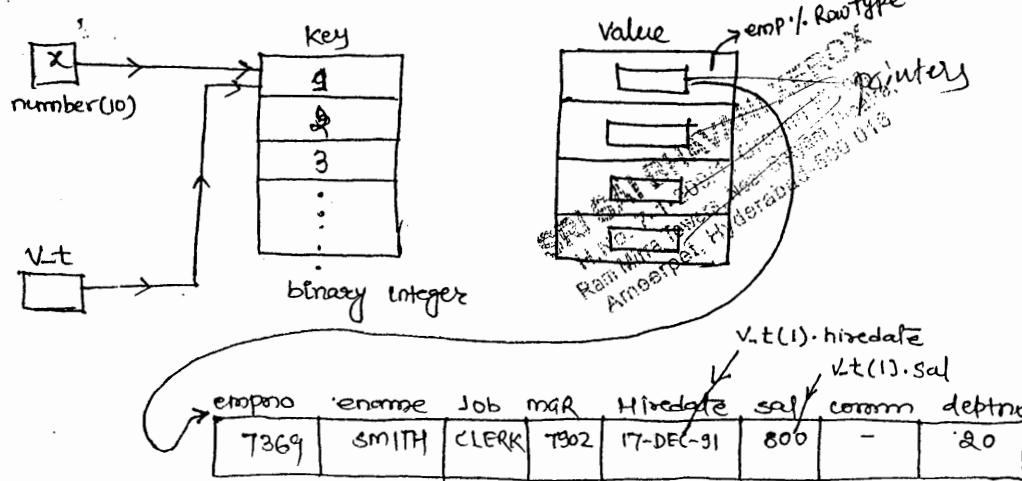
SQL declare

```

type t1 is table of varchar2(10)
index by varchar2(10);
v_t t1;
x varchar2(10);
begin
 v_t('a') := 'murali';
 v_t('b') := 'abc';
 v_t('c') := 'xyz';
 v_t('d') := 'pqrs';
 x := 'a';
 loop
 dbms_output.put_line(v_t(x));
 x := v_t.next(x);
 exit when x is null;
 end loop;
end;
/

```

\* Using recordtype datatype (%rowtype) in value field:-



```

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).hire
date);
 x := v_t.next(x);
 exit when x is null;
 end loop;
end;
/

```

(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).
job);
 end loop;
end;
/

```

return result set :— If we want to return large amount of data from oracle database server into client application. Then first we must develop database server application and then execute this application by using client application.

- In oracle by using following two methods we are returning large amount of data from database server resultset client application. These are –

Method 1 :- Using index by Table

Method 2 :- Using ref cursor.

**MANOJ ENTERPRISES**  
 Plot No: 40, Gayathri Nagar,  
 Ameerpet, Hyderabad.  
 Cell: 8125378496

### Method-1 using index by table -

In oracle generally we are implementing database server application function  
(or) by using procedure out parameter.

### database server application -

```
sql> create or replace package p_j_1
```

```
is
 type t_1 is table of emp%rowtype
 index by binary_integer;
 function f_1 return t_1;
end;
/
```

```
sql> create or replace package body p_j_1
```

```
is
 function f_1 return t_1
 is
 v_t t_1;
 begin
 select * bulk collect into v_t
 from emp;
 return v_t;
 end t_1;
end;
/
```

### executing by using pl/sql client :-

```
sql> declare
```

```
 x p_j_1.t_1;
begin
 x := p_j_1.f_1;
 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 used in index by table -

Exists collection method is used in index by table, nested table, varray, Exist collection method always return boolean value either true or false this collection method is used to test whether request data is available in collection or not.

Syntax -

```
CollectionVarname.exists (IndexName)
```

ex:- 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
z := v_t.exists(2);
if z = true then
dbms_output.put_line('your requested index 2 exists
with having no element 11' || v_t(2));
else
dbms_output.put_line('index 2 does not exists');
end if;
dbms_output.put_line(v_t.first);
dbms_output.put_line(v_t.last);
for i in v_t.first .. v_t.last
loop
dbms_output.put_line(v_t(i));
end loop;
end;
/

```

O/P: Your requested index 2 exists with having no element 20

1  
4  
10  
20  
30

ex:- SQL> declare

```

type t1 is table of varchar2(10)
index by binary_integer
v_t t1;
begin
select ename bulk collect into v_t from emp;
v_t.delete(3);
for i in v_t.first .. v_t.last
loop
dbms_output.put_line(v_t(i));
end loop;
end;
/

```

O/P : SMITH  
ALLEN  
ORA-01403: no data found

whenever index by table or nested table having gaps and also if have try to display those collection data then oracle server returns an error.

ORA-01403: No data found

To overcome this problem we must use exists collection Method.

Solution - SQL> 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;
/
```

### ✓ nested table, Varray :-

oracle 8.0 introduced nested, Varray.  
These collections also userdefined types which is used to store no. of data items in a single unit.

Before we are storing data into these two collections then we must initialize through constructor. Here Constructor name are also same as typename.

### Nested table :-

This is an unbound table, which is used to store no. of data items in single unit.

Nested table doesn't have key-value pairs.

Generally we are not allowed to store Index by table permanently to the database. To overcome these problems oracle 8.0 introduced extension of the index by table, which is used to store permanently to the DB by using SQL.

Generally in index by table we can't add or remove indexes whereas a nested table we can add or remove indexes by using extend, trim, collection Method.

Here indexes are always starts with 1 and also these indexes are consecutive.

nested table have a `extend`, `trim`, `exists`, `first`, `Last`, `Prior`, `next`, `count`, `delete(index)`, `delete(index, anotherindex)`, `delete collection` Method 8.

- This an userdefine type so, we are creating it two step process.
    - ① We are creating that type then only we are creating a variable from that type.

## Syntax -



## Index by table

e.g. SQL declare type +

type t, is table of number(10)

index by binary-integer;

$\sqrt{t - t_1}$

begin

$$v_t(500) = 30;$$

```
dbms_output.put_line(v_t(500));
```

end;

1

O/P :- 80

index by table are basically sparse i.e. explicitly we are not allowed to allocate memory. oracle server only automatically reserve the memory upto those indexed.

### nested table -

SQL> declare

type t, 18 table of number (10);

$$v - t \cdot t_1 := t_1(\cdot)$$

begin

$$V-t(500) := 30;$$

dbmng\_output.Put\_line(v\_t(500));

end;

1

error: subscript beyond count

- Nested tables are basically tensed i.e. we must reserve the memory explicitly by using extend collection method upto that indexes.

solution.

sql> declare

```
type t1 is table of number(10);
v_t t1 := t1();
begin
 v_t.extend(500);
 v_t(500) := 30;
 dbms_output.put_line(v_t(500));
end;
/
```

O/P :- 30

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;
 v_t(4) := 40;
 dbms_output.put_line(v_t(1));
end;
/
```

O/P: 10

NOTE:- In nested table we can also store data directly without using extend collection method, in this case we must specify actual data within constructor itself.

eg:- sql> declare

```
type t1 is table of number(10);
v_t t1 := t1(10, 20, 30, 40, 50);
begin
 for i in v_t.first .. v_t.last
 loop
 dbms_output.put_line(v_t(i));
 end loop;
 end;
/
```

SQL declare

```
type t1 is table of number(10);
v_t1, t1;
v_t2 t1 := t1();
begin
if v_t1 is null then
dbms_output.put_line('v_t1 is null');
else
dbms_out.put_line('v_t1 is not null');
end if;
if v_t2 is null then
dbms_output.put_line('v_t2 is null');
else
dbms_output.put_line('v_t2 is not null');
end if;
end;
```

/

Output:-

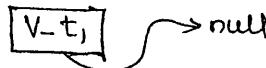
v\_t<sub>1</sub> is null

v\_t<sub>2</sub> is not null

① declare

```
Type t1 is table of number(10);
```

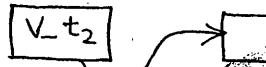
v\_t<sub>1</sub>, t<sub>1</sub>;



② declare

```
Type t1 is table of number(10);
```

v\_t<sub>2</sub> t<sub>1</sub> := t<sub>1</sub>();



③ declare

```
Type t1 is table of number(10);
```

v\_t t<sub>1</sub> := t<sub>1</sub>();

begin

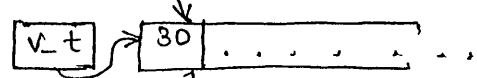
v\_t.extend;

v\_t(1) := 30;

dbms\_output.put\_line(v\_t(1));

end;

/



SQL> declare

```
type t1 is table of number(10);
V_t t1 := t1(10,20,30,40);
begin
 dbms_output.put_line(v_t.first);
 dbms_output.put_line(v_t.last);
 dbms_output.put_line(v_t.prior(3));
 dbms_output.put_line(v_t.next(2));
 dbms_output.put_line(v_t.count);
 v_t.extend;
 v_t(5) := 50;
 v_t.extend(3,4);
 v_t.trim;
 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;
 end;
 /
```

which value of  
the index

extend (3,4)

index

- Difference between trim, delete collection Method:-

SQL> declare

```
type t1 is table of number(10);
V_t t1 := t1(10,20,30,40,50,60);
begin
 v_t.trim(2);
 dbms_output.put_line('after deleting last two elements');
 for i in v_t.first .. v_t.last
 loop
 dbms_output.put_line(v_t(i));
 end loop;
 v_t.delete(2);
 dbms_output.put_line('after deleting second element');
 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;
 /
```

### NOTE:-

In varrays we are not allowed to delete particular elements or range of indexes by using delete collection method but we can delete all the elements by using delete collection method; that's why Varray doesn't have any gaps.

- Q. Write a PL/SQL program which is used to transfer first-10 employee names from emp table in Varray and also display content from Varray.

```
SQL> declare
 type t1 is varray(10) of varchar2(10);
 v_t t1 := t1();
 begin
 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;
 /
```

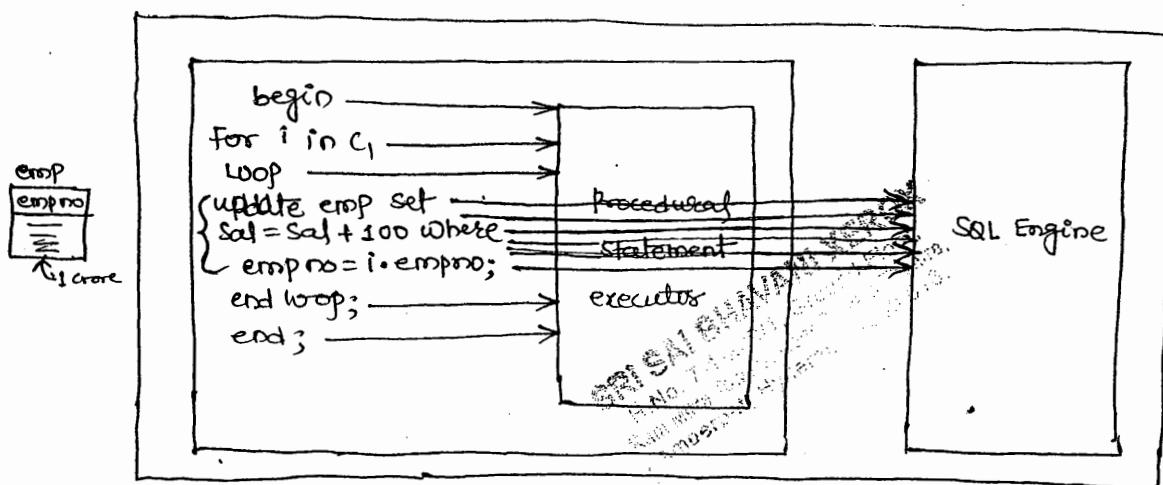
### Difference between Index by table, Nested table, Varray :-

| <u>Index by table</u>                                                                                                                       | <u>Nested table</u>                                                                                                                                        | <u>Varray</u>                                                                                            |
|---------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------|
| 1; Index by tables is an unbounded table having key-value pairs.                                                                            | 1; gt is unbound table                                                                                                                                     | 1; Varray is an bounded table which stores aGB data.                                                     |
| 2; Index by tables are not allowed to store permanently in oracle database.                                                                 | 2; Nested tables are allowed to store permanently in oracle database by using SQL.                                                                         | 2; we can also stores Varray permanently into database by using SQL.                                     |
| 3; we cannot add (or) remove indexes                                                                                                        | 3; we can add (or) remove indexes by using extend, trim collection method.                                                                                 | 3; we can add (or) remove indexes by using extend, Trim collection methods.                              |
| 4; Here indexes are either integers (or) characters and also +ve, -ve numbers.                                                              | 4; Here always indexes are integer starts with 1.                                                                                                          | 4; Here always indexes are integer starts with 1.                                                        |
| 5; Index by table having exists, first, last, prior, next, count, delete(index), delete (oneindex,anotherindex), delete collection methods. | 5; Nested table having exists, extends, trim, first, last, prior, next, count, delete (index), delete (one index, another index) delete collection method. | 5; Varray having exists, Limit, extend, trim, first, last, prior, next, count, delete collection method. |

whenever we are submitting a PL/SQL block into the oracle server then all sql statements are executed, within sql engine and also all procedure statement are executed, within procedural statement executor under PL/SQL engine these type of execution methods are also called as context switching execution methods.

whenever PL/SQL block having more no: of sql, procedural statements then these type of context switching execution methods degrades performance of the application to overcome this problem, For improve performance of the application oracle 8i introduced bulk-bind concept, through collections.

without using bulk bind (Performance Penalty for many context switching)

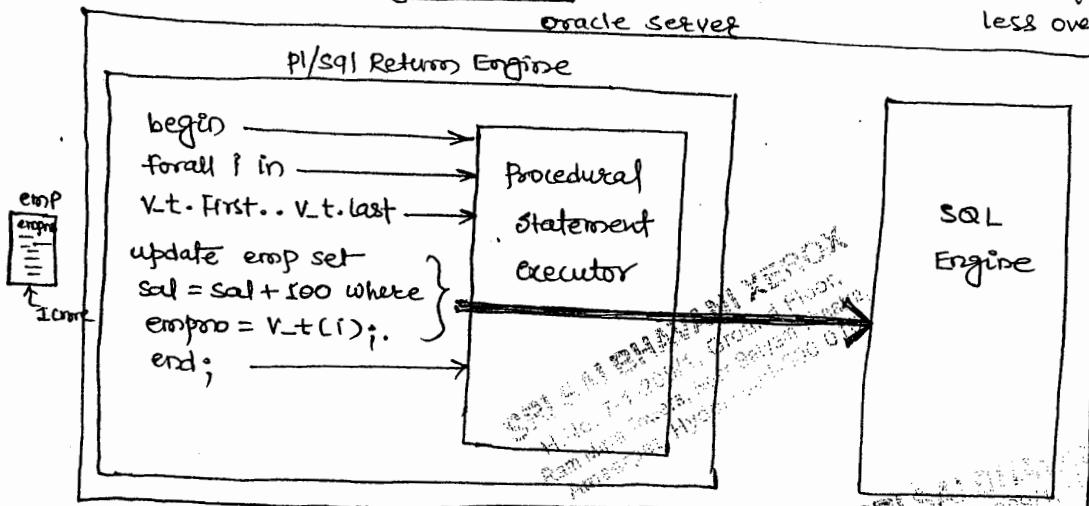


- In bulk bind process we can execute multiple DML statement at a time, bulk bind process always improves performance of the application, because it reduces no: of context switches i.e. in bulk-bind process we are using for all statement, through this for all statement we can execute all DML statements at a time by using sql engine. i.e. in bulk-bind process first we are collecting data from table into collection by using bulk-collect clause and then execute all these collection data at a time by using sql engine through for all statements.

#### Syntax:-

```
forall indexvalname in collectionvarname.first..collectionvarname.last
 DML statement where columnname = collectionvarname
 (indexvalname);
```

using bulk bind (improve Performance because of much less overhead)



- Basically bulk bind is a two step process -

Step-1 : Fetching data from table into collection by using bulk collect clause.

Step-2 : process all data in a collection at a time by using sql engine through forall statement. (actual bulk bind).

#### Step-1 :-

Before we are using forall statement in bulk-bind process first we must fetch data from resource into collection by using bulk collect clause. In oracle bulk collect clause used is -

- 1) select ... into clause
- 2) cursor.:fetch.:statement
- 3) DML... returning ... into clause.

#### 1) Bulk collect clause used in select... into clause -

##### Syntax :-

```
select * bulk collect into
collectionvarname from tablename
where condition;
```

e.g:- SQL> declare

```
type t_1 is table of emp%rowtype
index by binary integer;
v_t t_1;
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 —

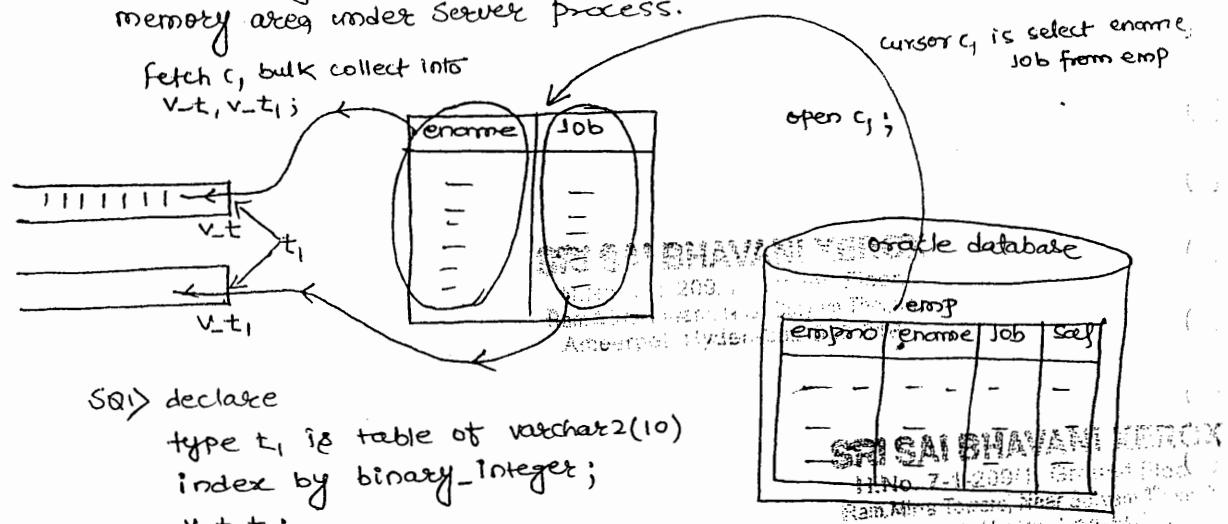
Syntax:-

Fetch cursorname bulk collect into collectionvarname [limit anynumber]

NOTE:-

Here limit is an optional clause, which is used to restrict no. of rows within PGA memory area.

In oracle by default all collections are executed with "PGA" memory area under Server process.



```
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;
```

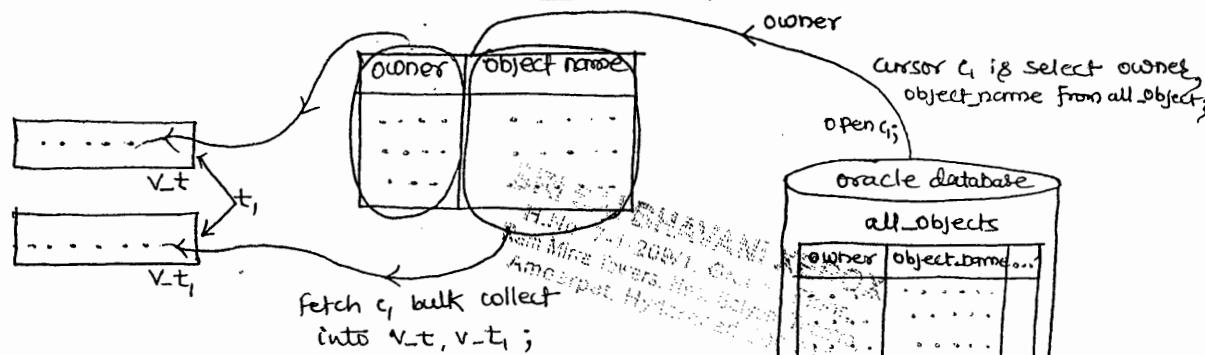
```
/
```

## Calculating Elapsed time in PL/SQL block:-

In PL/SQL if we want to calculate elapsed time within PL/SQL block, then we must use `get_time` function from `dbms_utility` package.  
This function always returns number data-type.

Syntax :-

```
vartime := dbms_utility.get_time;
```



```
SQL> declare
 type t1 is table of
 all_objects.owner%type
 index by binary_integer;
 v_t t1;
 v_t1 t1;
 cursor c1 is select owner, object_name from all_object;
 z1 number(10);
 z2 number(10);
 begin
 z1 := dbms_utility.get_time;
 for i in c1
 loop
 null;
 end loop;
 z2 := dbms_utility.get_time;
 dbms_output.put_line('Elapsed time for normal Fetch' || '||'
 (z2-z1) || '||' 'hsecs');
 z1 := dbms_utility.get_time;
 open c1;
 fetch c1 bulk collect into v_t, v_t1;
 close c1;
 z2 := dbms_utility.get_time;
 dbms_output.put_line('Elapsed time for bulk fetch' || '||'
 (z2-z1) || '||' 'hsecs');
 end;
 /
```

### Output -

Elapsed time for normal  
Fetch 76 hsecs

Elapsed time for bulk  
Fetch 66 hsecs

3) Bulk collect clause used in dml ... returning ... into clause -

We can also store DML transaction values into variables by using returning into clauses.

example- SQL> variable a varchar2(10);

SQL> update emp set sal = sal + 100

where ename = 'KING' returning job into :a;

SQL> print a;

O/P :- A  
PRESENT

- Oracle 8i onwards we can also use bulk collect clause within DML returning into clauses which is used to store multiple transactional values into collections.

eg:- 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;  
/

Output:

affected number of clerks are: 9

1200

4600

1300

5100

## Step-2 :-

Process all data in a collection at a time by using sql engine through forall statements (actual bulk bind).

- once data is there in a collection, through this collection data we're using bulk update, bulk delete, bulk inserts, by using forall statements. this is called "bulk-bind Process". In this process, all the DML statement are executing at a time by using sql engine.

### Syntax -

```
forall IndexVarName in collectionVarname • first..collectionVarname
 • last
 dml statement where columnname = collectionVarname(indexVarname);
```

e.g:- SQL> 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 which is used to transfer all employee no. from emp table into index by table by using bulk collect clause and also modify salaries of all the emp at a time by using bulk bind process?

SQL> declare

```
type t1 is table of number(10)
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;
/
```

**MANOJ ENTERPRISES**  
 Plot No: 40, Gayathri Nagar,  
 Ameerpet, Hyderabad.  
 Cell: 8125378496

### bulk insert -

- Q. write a pl/sql program which is used to transfer all empno from emp table into target table by using bulk bind process. In this process modify some emp names within a collection?

```

SQL> Create table target (name varchar2(10));
SQL> declare
 type t_ is table of varchar2(20)
 index by binary_integer;
 v_t t_;
begin
 select ename bulk collect into v_t from emp;
 v_t(1) := 'abc';
 v_t(2) := 'xyz';
 v_t(3) := 'pqr';
 for all i in v_t.first .. v_t.last
 insert into target values(v_t(i));
 end;
/
SQL> select * from target;

```

eg - SQL> declare

```

type t_ is table of number(10)
index by binary_integer;
v_t t_;
begin
 select empno bulk collect into v_t from emp;
 v_t.delete(3);
 forall i in v_t.first .. v_t.last
 update emp set sal = sal + 100
 where empno = v_t(i);
 end;

```

error: element at index [3] does not exist.

- whenever index by table (or) nested table having gaps then we're not allowed to use bulk bind process (forall statements)

To overcome this problem then we're using varray because varray doesn't have any gaps; but using varray we can store upto 2GB data, to overcome these problems oracle has introduced indices of clause within forall statement.

Syntax :-

forall indexvaluename in indices of collectionvarname

DML statement where columnname = collectionvarname(indexvaluename);

solution (using indices of 10g)

SQL> declare

type t1 is table of number(10)

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;

/.

before oracle 10g

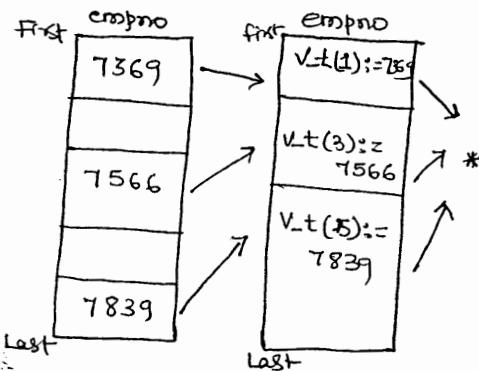
| empno |
|-------|
| 7369  |
| 7902  |
| 7566  |
| 7788  |
| 7839  |

first

last

update all rows  
in an array  
In order of array  
listing.

oracle 10g (using indices & clause)



SRI SAI BHAVANI XEROX  
H.No. 7-1-209/1, Ground Floor  
Ram Mira Towers, New Gachibowli  
Ameerpet, Hyderabad-500 013.

\*- update all rows within an  
array in order of array  
variable.

## bulk delete -

```
SQL> 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
 delete from emp where
 deptno = v_t(i);
 end;
 /
```

## SQL%bulk\_rowcount -

oracle introduced sql%.bulk\_rowcount attribute which is used to count affected number of rows within each group in bulk bind process. (forall statement).

sql%.bulk\_rowcount (indexvarname) → syntax

eg - SQL> 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);
```

```
for i in v_t.first..v_t.last.
```

```
loop
```

dbms\_output.put\_line ('affected number of rows in  
dept no');

|| v\_t(i) || 'is' || '|| sql%.bulk\_rowcount(i);

```
end loop;
```

```
end;
```

```
/
```

output - affected number of rows in deptno 10 is 3  
affected number of rows in deptno 20 is 5

⋮

## Forall and dml errors :-

(or)

### bulk exception -

- In oracle forall statement is an important performance enhancement in PL/SQL.
- We can use forall statement whenever we're executing DML statement inside the loop.
- forall statement typically executes multiple DML statements.
- whenever an exception occurs in one of the DML statements the default behaviour is that statement is rollback and also forall statement execution stops; second one is - all previous successful statements are never rollback.  
If we want to continue forall processing even if one of the DML statement having error also then we must add some exception clause within forall statements.
- whenever we are using some exception clause then oracle server stores exception information and continue processing of DML statements. These exceptions are stored in predefined, Pseudo collection: `sql%bulk_exceptions` index by table. This index by table having only one collection method `count` that's why this is also called as Pseudo collection. This predefined pseudo collection having two fields -
  - ① `error_index`
  - ② `error_code`

Here `error_index` field stores index number of the exception, and `error_code` field stores oracle error code of the exception.



eg- SQL> Create table target (sno number(10) not null);

SQL> declare

type t1 is table of number(10);

V\_t t1 := t1 (10, 20, 30, 40, 50);

begin

V\_t(3) := null;

V\_t(4) := null;

```

forall i in v_t.first .. v_t.last
 insert into target
 values (v_t(i));
end;
/

```

Error: ORA-01400: cannot insert NULL into sno

If we want to handle bulk exceptions in forall statements then we must use following two steps.

Step-I - Add a save exceptions clause in forall statement :-

Syntax

```

forall indexvarname in collectionvarname.first ..
collectionvarname.last save exceptions

```

```

DML statement where columnname = collectionvarname
(indexvarname);

```

Whenever we are using save exception clause, oracle server save those exception in predefine sql%bulk\_exceptions collection.

Step-II :- handling exception in exception handler :-

We must catch exceptions from sql%bulk\_exceptions index by table through count collection method by using following syntax.

Syntax

```

varname := sql%bulk_exceptions.count;

```

Here count collection method always count no. of exception occurred in the process.

SQP:-

```

SQL> declare
 type t1 is table of number(10);
 v_t t1 := t1(10, 20, 30, 40, 50);
 z number(10)
 begin
 v_t(3) := null;
 v_t(4) := null;
 forall i in v_t.first .. v_t.last save exceptions
 insert into target values (v_t(i));
 exception
 when others then
 z := sql%bulk_exceptions.count;
 end;

```

```

dbms_output.Put_line(z);
end;
/

```

outputs:- 2

SQL> select \* from target;

| SNO |
|-----|
| 10  |
| 20  |
| 50  |

NOTE:- If we want to display exception indexes and error numbers, within bulk bind process, then we are using following syntax.

**Syntax**

sql%> bulk\_exceptions(indexvarname).error\_index

**Syntax**

sql%> bulk\_exceptions(indexvarname).error\_code

eg:- SQL> declare

```

type t1 is table of number(10);
v_t t1 := t1(10, 20, 30, 40, 50);
z number(10);
begin
v_t(3) := null;
v_t(4) := null;
forall i in v_t.first .. v_t.last
save exceptions
insert into target
values (v_t(i));
exception
when others then
z := sql%bulk_exception.count;
dbms_output.Put_line(z);
for i in 1..z
loop
dbms_output.Put_line(sql%bulk_exceptions(i).error_index
|| sql%bulk_exceptions(i).error_code);
end loop;
end;
/

```

| Output: | 2      |
|---------|--------|
|         | 3 1400 |
|         | 4 1400 |

## dbms\_utility Package -

This Predefine package internally having Predefine index by table and also this package following two procedure-

1) comma\_to\_table

2) table\_to\_comma

1) comma\_to\_table :- This procedure is used to convert comma separated string into index by table values.

This procedure accepts three parameters.

Syntax → dbms\_utility.comma\_to\_table ( string varname, binary\_integer index\_by\_table varname )

2) table\_to\_comma - This procedure is used to convert index by table values into comma separated strings.

Syntax - dbms\_utility.table\_to\_comma ( index\_by\_table binary\_integer string varname , varname , varname )

Before using these procedure we must declare index by table variable by using `umcl_array` type from `dbms_utility` package in declare section of the PL/SQL blocks.

Syntax - Index by table dbms\_utility.umcl\_array;  
varname

Q. Write a PL/SQL Program which is used to transfer comma separated string into index by table using `dbms_utility` package and display content from index by table ?

SQL> declare

v\_t dbms\_utility.umcl\_array;

x binary\_integer;

str varchar2(200);

begin

str := 'a, b, c, d, e, f';

dbms\_utility.comma\_to\_table(str, x, v\_t);

for i in vt.first .. vt.last

loop

dbms\_output.put\_line(v\_t(i));

end loop;

end;

O/P:  
a  
b  
c  
d  
e  
f

Q. write a PL/SQL program which is used to retrieve all dept\_name from dept table and displayed into comma separated string by using dbms\_utility package.

```
SQL> declare
 v_t dbms_utility.varcarray;
 x binary_integer;
 str varchar2(200);
begin
 select dname bulk collect into v_t from dept;
 dbms_output.put_line(str);
 dbms_output.put_line(to_comma(v_t,x,str));
end;
/
```

Output:

ACCOUNTING, RESEARCH, SALES, OPERATIONS

## REF CURSOR (or) CURSOR VARIABLE (or) DYNAMIC CURSOR

- oracle 7.2 introduced ref cursor, this is an user defined type which is used to process multiple records and also this is a record by record process.
- In static cursor database servers executes only one select statement at a time for a single active set area where in ref cursor database servers executes no. of select statement dynamically for a single active set area that's why those cursor are also called as dynamic cursor.

generally we are not allowed to pass static cursor as parameters to use subprograms where as we can also pass refcursor as parameter to the subprograms because basically refcursor is an userdefined type in oracle we can also pass all user defined type as parameter to the subprograms.

generally static cursor does not return multiple record into client application where as refcursor are allowed to return multiple records into client application (Java, .Net, PHP, VB, C++, ...)

This is an userdefined type so we are creating it in 2 step process i.e. first we are creating type then only we are creating variable from that type that's why this is also called as cursor variable.

In all databases having 2 ref cursors.

1. Strong ref cursor
2. Weak ref cursor

Strong ref cursor is a ref cursor which have return type, whereas weak ref cursor has no return type.

Syntax

- ① type Typename is ref cursor return recordtypeldatatype;
- ② variablename typename;  
    ^ strong ref cursor variablename

Syntax

- ① type Typename is refcursor
- ② variablename Typename;  
    ^ weak ref cursor variablename

- In ref cursor we must specify select statement by using open for clause this clause is used in executable section of the pl/sql block.

Syntax :-

```
open refcursorVariableName for select * from tablename
condition;
```

- Q. using ~~strong~~ ref cursor to refer employees and department details

```
procedure P_get_employees (Pi_deptno in integer,
Po_results out refStrong_cmrtyp)
```

is

begin

open Po\_results for

select \* from emp e

where e.deptno = Pi\_deptno;

Exception

when others then

raise;

End P\_get\_employees;

### Examples Based on ref cursors -

Q. write PL/SQL programs using ref cursor for employee having salary is more than equals to 2000; then display employee name and salary from emp table?

Ex:- declare

```
type t1 is ref cursor;
v_t t1;
i emp%.rowtype;
begin
open v_t for select * from emp
where sal > 2000;
loop
fetch v_t into i;
exit when v_t%notfound;
dbms_output.put_line (i.empname || i.sal);
end loop;
close v_t;
end;
/
```

Q. write a PL/SQL program using ref cursor for user entered dept no=10 then display 10<sup>th</sup> dept. detail from emp table whenever user entered dept no=20 then display 20<sup>th</sup> dept details from dept table?

```
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.sal || '|| i.deptno);
end loop;
elseif v_deptno = 20 then
open v_t for select * from dept where deptno=20;
loop
fetch v_t into j;
exit when v_t%notfound;
dbms_output.Put_line (j.deptno || '|| j.dname || '|| j.loc);
end loop;
close v_t;
end if;
end;
/

```

output:

Enter value for deptno : 10

|        |      |    |
|--------|------|----|
| CLARK  | 4800 | 10 |
| KING   | 9900 | 10 |
| MILLAR | 6200 | 10 |

SQL> /

output: Enter value for deptno : 20

|    |          |        |
|----|----------|--------|
| 20 | RESEARCH | DALLAS |
|----|----------|--------|

#### - sys\_refcursor

oracle gi introduced sys\_refcursor predefine type in place of weak refcursor.

|                                                 |
|-------------------------------------------------|
| <u>syntax</u> → refcursorVarName sys_refcursor; |
|-------------------------------------------------|

```

SQL> declare
 v_t sys_refcursor;
 i emp%rowtype;
begin
 open v_t for select * from emp;
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 refcursor as Parameter to the stored procedure :-

1) Passing sys\_refcursor as in parameter to the stored procedure:-

Q: write a PL/SQL stored procedure for passing sys\_refcursor as in parameter then display emp detail from emp table ?

```
SQL> create or replace procedure P1 (v_t in sys_refcursor)
 IS
 i emp%rowtype;
 begin
 loop
 fetch v_t into i;
 exit when v_t%notfound;
 dbms_output.put_line (i.empname || ' ' || i.sal);
 end loop;
 end;
 /
```

NOTE:-

In all databases whenever we are passing refcursor as IN Param. to the subprograms then we're not allowed to use open for stats. within subprograms because IN Parameter is used to pass the value into procedure body.

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;
/
```

SRI SAI BHAVANI XEROX  
H.No. 7-1-209/1, Ground Floor,  
Ram Niwas Towers, Near Satyam Theatre,  
Ameerpet, Hyderabad-500 016.

2) Passing sys\_refcursor as out parameter to the stored procedure

Q. write a pl/sql stored procedure for passing sys\_refcursor as out parameter which returns emp details from emp table ?

```
SQL> create or replace Procedure P,
 (v_t out sys_refcursor)
 is
 begin
 open v_t for select * from emp;
 end;
 /
```

Execution (by using pl/sql client) -

```
SQL> declare
 v_t sys_refcursor;
 i emp%rowtype;
 begin
 P_1(v_t);
 loop
 fetch v_t into i;
 exit when v_t%notfound;
 dbms_output.put_line ('-'||ename||'.'||i.sal);
 end loop;
 close v_t;
 end;
 /
```

Q. write pl/sql stored procedure function using sys\_refcursor as return type which returns Employee details from emp table ?

```
SQL> create or replace function f,
 return sys_refcursor
 is
 v_t sys_refcursor;
 begin
 open v_t for select * from emp where
 sal > 2000;
 return v_t;
 end;
 /
```

Execution : SQL> select f, from dual;

## \* Strong refcursor:-

```
SQL> create or replace package PJ1
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 PJ1
is
procedure P1(v_t1 out t1)
is
begin
open v_t1 for select * from emp;
end P1;
procedure P2(v_t2 out t2)
is
begin
open v_t2 for select * from dept;
end P2;
end;
/
```

execution:- (by using bind variable)

```
SQL> variable a refcursor;
SQL> variable b refcursor;
SQL> exec PJ1.P1(:a);
SQL> exec PJ1.P2 (:b);
SQL> print a b;
```

### NOTE:-

In all databases we are not allowed to use refcursor variable within Packages.

eg:-    SQL> create or replace package PJ2  
is  
type t1 is ref cursor;  
v\_t t1;  
end;  
/

Warning: Package created with compilation errors.

SQL> show errors

Error: cursor variables cannot be declared as part of a package.

## Local SubPrograms

04/feb/16.

Local subprograms are named pl/sql block which is used to solve particular task.

Oracle having two types of subprograms -

- (a) Local Procedures
- (b) Local Functions

These subprograms are not stored permanently to the DB.

These subprograms doesn't have create or replace keyword.

- In oracle local subprograms must be defined in bottom section of anonymous block and also declare section of the stored procedure and also call those local subprograms in immediate executable section.

Syntax:

```
declare
 → variable, constant declaration;
 → Types declaration;
 → cursors declaration;
 → Procedure procedurename (formal Parameter)
 is

begin

end [Procedure name];
→ Function functionname (formal Parameters) return datatype
 is/as

begin

 Return exp;
end [function name];
begin
 procedurename (actual Parameter);
 variablename := functionname (actual Parameter);
end;
```

```
SQL> declare
 procedure P1
 is
 begin
 dbms_output.Put_line ('local proc');
 end P1;
 begin
 P1;
 end;
 /
```

```
SQL> create or replace procedure P2
 is
 Procedure P1
 is
 begin
 dbms_output.Put_line ('Local proc');
 end P1;
 begin
 P1;
 end;
 /
SQL> exec P2;
o/p: Local proc
```

eg:- SQL> declare  
 type t1 is ref cursor return  
 emp%rowtype;  
 v\_t t1;
 procedure P1 (P\_t in t1)
 is
 i emp%rowtype;
 begin
 loop
 fetch P\_t into i;
 exit when P\_t%notfound;
 dbms\_output.put\_line(i.ename || '||' i.sal);
 end loop;
 end P1;
 begin
 open v\_t for select \* from emp
 where rownum <= 10;
 P1(v\_t);
 close(v\_t);

CRY SAJ BHAVANI VENKATESH  
W No. 7-1-2004, Ground Floor,  
Ram Mirra Towers, Near Gachibowli  
Amaravati, Hyderabad - 500 070.  
Ph: 9908959556

```

open v_t for select * from emp
where sal > 3000;
P,(v_t);
close v_t;
open v_t for select * from emp
where job = 'CLERK';
P,(v_t);
close v_t;
END;
/

```

### SHRI SAI BHAVANI KERALA

H.No. 7-1-200/1, Ground Flr.  
Rani Mirra Towers, Near Salyam Tin  
Ameerpet, Hyderabad - 500 016

Date  
5<sup>th</sup> FEB 16.

## UTL\_FILE Package

- oracle 7.3 introduced utl\_file package. This package is used to write data into an O.S. file and also read data from O.S. file.
- If we want to write data into O.S. file then we are using putf() procedure from utl\_file package. Whereas if we want to read data from file then we are using get\_line() procedure from utl\_file package.

In oracle before we are using utl\_file package or LOB then we must create alias directory related to physical directory by using following syntax.

Syntax -

```

create or replace directory
DIRECTORY NAME as 'path';

```

- Before we are creating alias directory that database administrator creates any directory system privileges to user otherwise oracle server returns an error, insufficient privileges.

Syntax:-

```

grant create any directory to username;

```

eg:-    SQL> conn sys as sysdba;  
         Enter password: sys

SQL> grant create any directory to scott;

SQL> conn scott/tiger;

SQL> create or replace directory

XYZ as 'C:\';

always capital letter we use'

Before we are performing read, write operations then we must use read, write object privileges on alias directory by using following syntax -

grant read, write on directory  
directoryname to username;

ex:- SQL> conn sys as sysdba;  
Enter password: sys

SQL> grant read, write on directory  
XYZ to scott;

SQL> conn scott/tiger;

### ① writing data into an O.S. file

Step-1:- Before we are opening the file we must create file pointer variable by using file\_type from utl\_file package in declare section of the pl/sql block.

Syntax - filePointerVarname utl\_file.file\_type;

Step-2:- Before we are writing data into file then we must open the file by using fopen() function from utl\_file Package. This function is used in executable section of pl/sql block, this function accepts three parameters and returns file\_type datatype.

Syntax - filePointerVarname := utl\_file.fopen('alias directoryname',  
'filename', 'mode');

mode

- W
- R
- A

w - write  
r - read  
a - alias

Step-3:- If we want to write data into file then we are using putf() procedure from utl\_file Package.

Syntax utl\_file.putf(filePointerVarname, 'content');

Step-4:- After writing data into file then we must close the file by using fclose procedure from utl\_file Package.

Syntax: utl\_file fclose(filePointerVarname);

eg:- SQL> declare

Step-1      fp utl\_file.file\_type ;  
begin  
Step-2      fp := utl\_file.fopen ('xyz', 'file1.txt', 'w');  
Step-3      utl\_file.putf (fp, 'abcdefghijklmn');  
Step-4      utl\_file.fclose (fp);  
end;  
/

Q. write a PL/SQL program which is used to retrieve all employee name from emp table and storing into an o.s. file by using utl\_file package.

SQL> declare

```
fp utl_file.file_type;
cursor c1 is select ename from emp;
begin
fp := utl_file.fopen ('xyz', 'file2.txt', 'w');
for i in c1
loop
utl_file.putf (fp, i.ename);
end loop;
utl_file.fclose (fp);
end;
/
```

|        |
|--------|
| Output |
| SMITH  |
| ALLEN  |
| CLARK  |
| BLAKE  |
| .....  |
| .      |

when we are try to store column data into O.S. file by using putf procedure from utl\_file package then oracle server stores column data in horizontal manner within O.S. file. To overcome this problem if we want to write data into file in our own format then we must use %'s access specifier within second parameter of the putf procedure and also if we want to display column data vertically then we must use \n along with access specifier %'s.

Syntax -

```
utl_file.putf (filepointername, 'format\n', variablename);
```

```

SQL> declare
 fp utl_file.file_type;
 cursor c1 is select ename from emp;
begin
 fp:=utl_file.fopen ('XYZ', 'file2.txt', 'W');
 for i in c1
 loop
 utl_file.putf (fp, '%s\n', i.ename);
 end loop;
 utl_filefclose(fp);
end;
/

```

NOTE:-

- we can also write data into os file through put\_line procedure from utl\_file package.
- This function accepts two parameter.

|                                                                                     |
|-------------------------------------------------------------------------------------|
| <b>Syntax</b><br><code>utl_file.put_line (filepointerVarname, variableName);</code> |
|-------------------------------------------------------------------------------------|

eg. SQL> declare

```

 fp utl_file.file_type;
 cursor c1 is select * from emp;
begin
 fp:=utl_file.fopen ('XYZ', 'file3.txt', 'W');
 for i in c1
 loop
 utl_file.put_line (fp, i.ename || '||' || i.sal || '||' || i.job);
 end loop;
 utl_filefclose(fp);
end;
/

```

reading data from OS file:-

If we want to read data from file then we are using get\_line procedure from utl\_file package before we are using this procedure we must specified read mode within fopen function from utl\_file package.

|                                                                                      |
|--------------------------------------------------------------------------------------|
| <b>Syntax</b><br><code>utl_file.get_line (filepointerVarname, bufferVarname);</code> |
|--------------------------------------------------------------------------------------|

Q. write a PI/SQL programs to read data from file1.txt by using utl\_file package and also display content from file.

```
SQL> declare
 fp utl_file.file_type;
 x varchar2(3000);
begin
 fp := utl_file.fopen ('XYZ', 'file1.txt', 'r');
 utl_file.get_line (fp, x);
 dbms_output.put_line ('data from file' || ' ' || x);
 utl_file.fclose(fp);
end;
/
```

Output: data from file abcdefghijklmn

Q. write a PI/SQL programs which is used to read multiple data item from file2.txt by utl\_file package; and also display multiple data item?

```
SQL> declare
 fp utl_file.file_type;
 x varchar2(2000);
begin
 fp := utl_file.fopen ('XYZ', 'file2.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;
/
```

whenever we are reading multiple data items from a file by using utl\_file package then oracle server returns an error ora-1403: No data found when control reach end of file to overcome this problem we must used no\_data\_found exception name within PI/sql block.

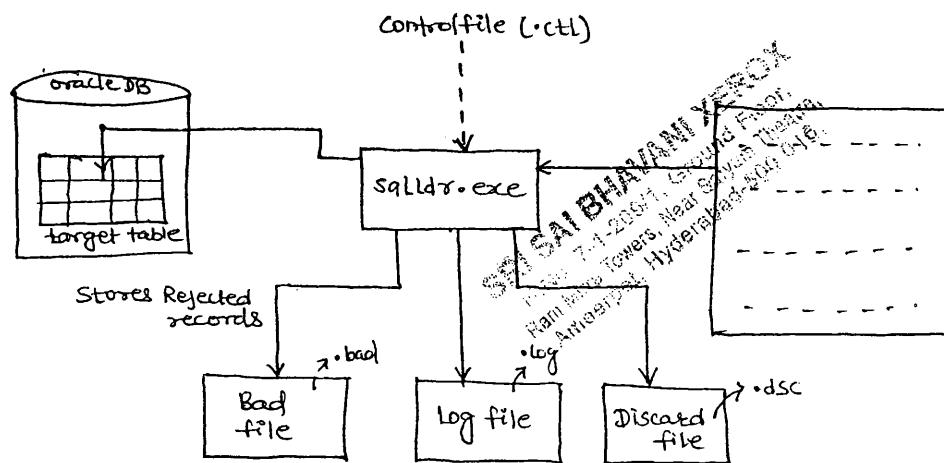
## SQL-LOADER

Date  
06/feb/16

- SQL-Loader is an utility program which is used to transfer data from flat into oracle database.
- SQL Loader always executes control file based on the type of flat file we are creating control file and then submit control file to SQL Loader then only SQL-Loader transfer file into flat file into oracle DB during this file some other files also created:-
  - ① Logfile
  - ② badfile
  - ③ discardfile.

① Logfile - This file stores all other files information and also some loaded, Rejected no. of files or records & store oracle error no; error message.

Based on some reason, some records are rejected. These rejected records are stored in Bad file and Discard file.



### I. Flat File -

flat file having NO. of Records, these file is structured file.

All languages having two types of flat File -

(a) Variable Record flat File

(b) Fixed Record Flat file

#### (a) Variable Record Flat File -

A flat File which having delimiters in between Fields is called variable Record Flat file.

ex:-      101, aab, 2000  
           102, mkl, 3000  
           103, xyz, 4000

## (b) Fixed Record Flat file -

A flat file which does not have delimiters is called Fixed Record flat file.

ex:- 101 abc 2000  
102 mno 3000  
104 xyz 4000

### Control File -

These file extension is .ctl, SQL Loader executes Control file only based on the type of flat file we are creating control file then only SQL-Loader transfer data into oracle DB sys.

#### • Creating a control file based on variable record flat file -

Step-1. always control file execution starts with Load data clause, after Load data clause we must specify path of the flat file by using infile clause.

|        |                                |
|--------|--------------------------------|
| syntax | load data                      |
|        | infile 'path of the flat file' |

#### NOTE:-

we can also specify flat file data into control file itself, in this case we must use \* instead of path of flat file within 'infile' clause & also use a 'begindata' clause in the above of the flat file data.

ex:- load data  
infile \*  
...  
...  
begindata  
101,abc,2000  
102,xyz,3000  
103,bbb,7000

Step-2- If we want to load data into oracle DB then we must use 'into table tablename' clause within Control File.

In the above of the into table tablename clause we can also use Insert/append/truncate/replace

when target table is an empty table then use Insert clause, by default insert clause is there.

Step 3: Based on the flat-file data we are using following clauses 'after into table tablename' clause. These clauses are -

- 1) fields terminated by  
'delimitername'
- 2) optionally enclosed by  
'delimitername'
- 3) trailing nullcols.

and last only we must specify ~~target~~ <sup>target</sup> table columns within Parenthesis.

#### control file (.ctl)

##### Syntax -

Load data  
infile 'Path of flatfile'  
badfile 'Path of badfile'  
discardfile 'Path of discardfile'  
insert/append/truncate/replace into table tablename  
fields terminated by  
'delimitername'  
optionally enclosed by  
'delimitername'  
trailing nullcols  
(cols, col<sub>2</sub>, ...)

#### Invoking SQL Loader (Executing) -

Start → run → cmd → C:\>sqlldr  
userid = scott/tiger  
control = path of the control file

##### Flat file

file1.txt

101,abc,2000  
102,xyz,3000  
103,mnm,4000

SQL> create table target (empno number(10), ename varchar2(10),  
sal number(10));

### control file (filename.ctl)

Load data

```
infile 'c:\file1.txt'
insert
into table target
Fields terminated by ','
(empno, ename, sal)
```

### Execution -

```
C:\>sqlldr userid=scott/tiger
control=c:\murali.ctl
```

O/P : Commit point reached [successful]

### To check

```
SQL> select * from target;
```

- During this process SQL-Loader automatically creates a Log file as same as name of control file. This Log file stores all ~~log file~~ other files information, also stores oracle errors no. and message.
- Log file stores Loaded Records, Rejected No. of Records Number.
- This log file also stores elapsed time of the operation performed.

(or)

```
SQL> load data
infile *
insert into table target
Field terminated by ','
(empno, ename, sal)
begin data
101, abc, 2000
102, xyz, 7000
```

Constant, filler clauses are used in Control File -

If we want to store default values into Oracle DB by using SQL Loader tool, then we must use constant clause.

Syntax -

columnname constant defaultValue

When flat file having less no. of fields and also target table requires more no. of fields then only we are allowed to use constant clause.

- If we want to skip columns from the table then we are use filler clause.

Syntax -

anyname filler

when flat file having more no. of fields and target table requires less no. of fields then only we are use filler clause.

Ex:-

file1

101,abc

102,xyz

103,bbb

SQL> create table target (emphno.number(10), loc varchar2(10));

murali.ctl

load data

infile 'c:\file1.txt'

insert

into table target

Field terminated by ','

(emphno,ename filler, loc constant 'mumbai').

MR. SAI BHAVANI REDDY  
A. NO. 74, 2nd Floor, 2nd Main,  
Rajajinagar, Bangalore - 560010.  
Mobile: 98800 44444  
Airtel: 98800 44444

SQL> select \* from target;

| Emphno | Loc    |
|--------|--------|
| 101    | mumbai |
| 102    | mumbai |
| 103    | mumbai |

Functions are used in Control file -

- we can also use oracle functions within control file, in this case
- we must use (specify) function functionality within double quotes
- and also we must use colon (:) in front of the columnname within function functionality.

Syntax -

columnname "functionname(:columnname)"

eg:- file1

101, abc, m

102, xyz, f

103, bbb, m

104, jjj, f

- Q. Create a control file for the above flat file which is used to convert m into male and f into female within gender column in oracle database.

SQL> create table target (empno number(10), ename varchar2(10), gender varchar2(10));

mutali.ctl

Load data

infile 'c:\file1.txt'

insert

into table target

fields terminated by ','

(empno,ename,gender

"decode (:gender,'m','male','f','female'))")

Bad File :-

- This file extension is .bad.
- Bad file stores rejected record based on -
  - datatype mismatch
  - Business rule violation.
- Bad file is automatically created as same name as flat file, we can also create bad file explicitly by using badfile clause within control file.

## 1) data type Mismatch -

### file1.txt

101, abc  
 '102', xyz  
 103, bbb  
 '104', jjj

SQL> create table target (empno number(10), ename varchar2(10));

### murali.ctl

Load data  
 infile 'c:\file1.txt'  
 Insert into table target  
 Fields terminated by ','  
 (empno, ename)

### file1.bad

'102', xyz  
 '104', jjj

SQL> select \* from target

O/P: 101, abc  
 103, bbb

## 2) Business rule violation -

### file1.txt

101, abc, 3000  
 102, xyz, 7000  
 103, bbb, 2000  
 104, jjj, 9000

SQL> create table target (empno number(10), ename varchar2(10),  
 sal number(10). check(sal > 5000));

### murali.ctl

Load data  
 infile 'c:\file1.txt'  
 Insert into table target  
 Field terminated by ','  
 (empno, ename, sal)

### file1.bad

101, abc, 3000  
 103, bbb, 2000  
 SQL> select \* from target  
 102, xyz, 7000  
 104, jjj, 9000

SPD SYSTEMS  
 100'ft. Main Road,  
 Opp. TCS, Ground Floor,  
 Ramnagar, Secunderabad,  
 Hyderabad, Telangana, India.  
 Ph: 9968195958

### NOTE -

In flat file when trailing fields having null values then automatically those null value records are rejected and also those records are stored in flat file. To overcome this problem if we want to store these records into oracle database then we must use trailing nullcols clause, within control file.

ex:-

#### file1.txt

```
101,abc,3000
102,xyz
103,jjj
104,nnn,4000
```

```
sql> create table target (empno number(10), ename varchar2(10),
sal number(10));
```

#### control.ctl

```
Load data
infile 'c:/file1.txt'
insert
into table target
fields terminated by ','
trailing nullcols
(empno,ename,sal)
```

### RECNUM -

This clause is automatically assign number to the loaded rejected record.

#### Syntax -

```
columnname renum
```

#### file1.txt

```
101,abc
'102',xyz
'103',jjj
104,nnn
```

```
sql> select * from target;
o/p: empno ename rno
```

```
101 abc 1
104 nn 4
```

```
sql> create table target (empno number(10),
ename varchar2(10), rno number(10)),
control.ctl
```

```
Load data
infile 'c:/file1.txt'
insert
into table target
field terminated by ','
(empno,ename,rno renum)
```

## Discard file :-

- This file extension is .dsc
- Discard file we must specify within control file by using discard file clause.
- Discard file also stores rejected record based on when clause condition within control file. This condition must be satisfied into table tablename clause.

### Syntax

when condition

### File1.txt

```
101,abc,10
102,xyz,20
103,yyy,10
104,nnn,30
105,KKK,10
```

### file2-dsc

```
102,xyz,20
104,nnn,30
```

```
SQL> create table target (empno number(10), ename
 varchar2(10), deptno number(10));
```

### Mutali.ctl

```
Load data
infile 'c:\file1.txt'
discardfile 'c:\file2.dsc'
Insert
into table target
when deptno = '10'
Fields terminated by ','
(empno, ename, deptno)
```

### NOTE -

- In Control file when clause condition value must be specified within single quotes (' )
- In when clause we are not allowed to use other than  $=$ ,  $<$   $>$  relational operators.
- In when clause we are not allowed to use logical operator "or" but we are allowed to use logical operator "and".

### DATES used in Control file -

method 1 : using datatype

method 2 : using to\_date() function.

### Method 1

#### Syntax

```
Columnname date "flatfile dateformat"
```

### file1.txt

```
101, abc, 170809
102, xyz, 200407
103, jij, 250708
```

SQL> create table target (empno number(10), ename varchar2(10), col3 date);  
murali.ctl

Load data

infile 'c:\file1.txt'

Insert

into table target

Fields terminated by ','

(empno, ename, col3 date "DDMMYY")

### Method 2:

syntax

"to\_date (:columnname, 'flatfiledate Format')"

### murali.ctl

Load data

infile 'c:\file1.txt'

Insert

into table target

fields terminated by ','

(empno, ename, col3 "to\_date (:col3, 'DDMMYY')")

### Sequences are used in control file -

We can also use oracle sequences within control file, in this case we must specify sequence pseudo columns within double quotes (" ") .

Syntax -

columnname "sequence name . nextval"

of: file1.txt

101

102

103

104

105

murali.ctl

load data

infile 'c:\file1.txt'

insert

into table target

fields terminated by ',' (sno, "s1.nextval").

SQL> create table target (sno number(10));

SQL> create sequence s1;

SQL> select \* from target;

of:  
1  
2  
3  
4  
5

## Creating a record file

### Creating a control file for fixed record flatfile -

A flatfile which have no any delimiter is called fixed record flatfile.

When we are using fixed record flatfile then we must use "Position" clause within control file. In position clause we must specify starting and ending position of the every field by using ":" operator. Along with position clause we must specify sqlLoader datatypes. sqlLoader having 3 datatypes these are -

- ① integer external
- ② decimal external
- ③ char

#### Syntax -

columnname : position (startingPosition : endingPosition) sqlloader datatype

#### NOTE:-

Whenever we are using functions or expressions then we are not allowed to use sqlLoader datatype. In place of this one we must use functions functionality within double quotes (" ") and also use colon operator (:) in front of columnname.

#### Syntax -

columnname position (startingPosition : endingPosition) "functionname (: colname)"

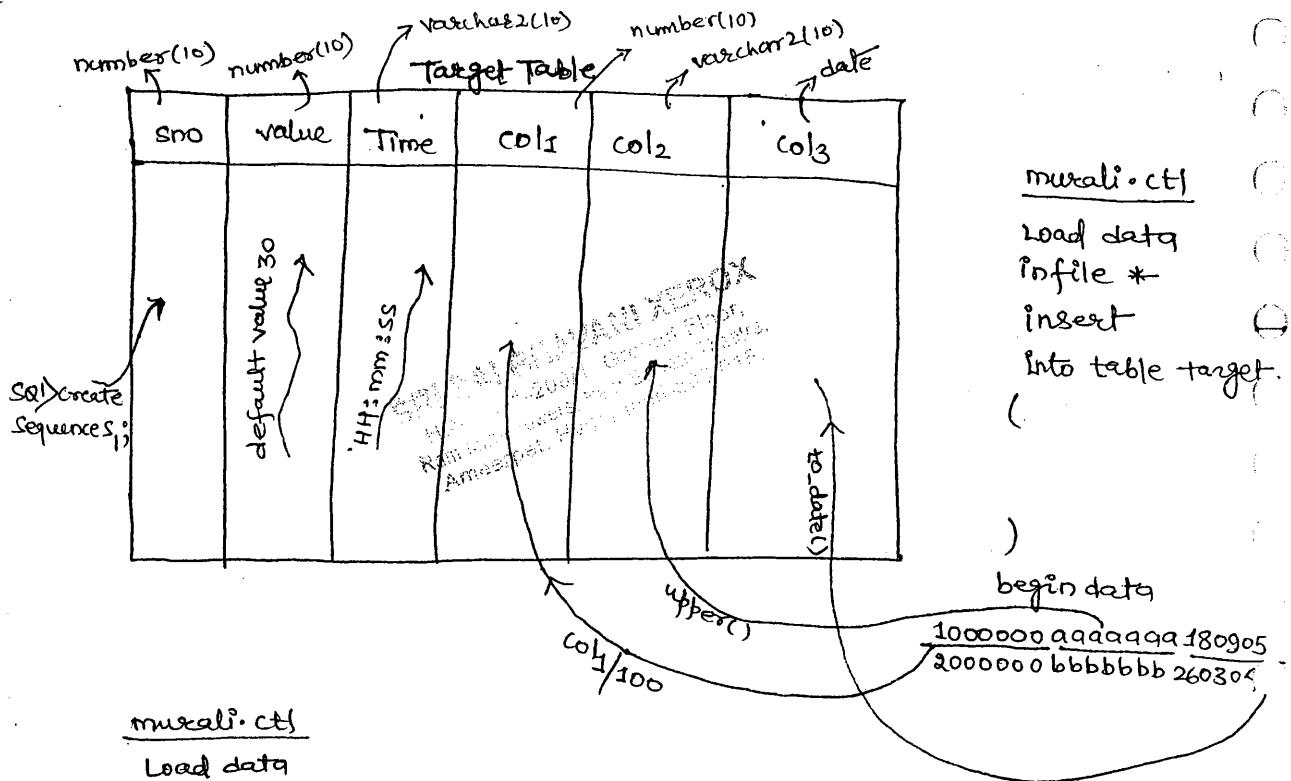
Eg:- file1.txt

101 abc 2000  
102 xyz 4000  
103 mmm 9000  
104 KKK 7000

SQL> create table target (empno number(10),  
ename varchar2(10), sal number(10));

#### murali. CTJ

```
Load data
infile 'c:\file1.txt'
insert
into table target
(empno position(01:03) integer external, ename position(04:06)
char, sal position(07:10) integer external)
```



### murali.cts

Load data

infile \*

insert

into table target

```
(sno "S1.nextval", value constant '50', Time "TO_CHAR(SYSDATE,'HH:MM:SS')"
 col1 position (01:07) ":" col1/100" col2 position (08:14) "UPPER(:col2)"
 col3 position (15:20) "TO_DATE (:col3, 'DD/MM/YY'))"
```

begin data

1000000aaaaaaaaa 180905

2000000bbbbbbb 260304

NOTE:- In oracle in front of sysdate, userfunctions we are not allowed to use ';' operator.

- Using sqlloader we can also transfer no. of flatfiles data into single target table in this case we must use no. of infile clause.
- Using sqlloader we can also transfer single flatfile data into multiple target tables, in this case we must use no. of into table tablename clauses, but using sqlloader we aren't allowed to transfer when resource as different databases data and also when resources as combination of flatfile, databases data. In this case generally we are using ETL tools.

## Triggers

12/feb/16.

- Trigger is also same as stored procedure & also it will automatically invoked whenever DML operation performed against table or view.

There are two types of triggers supported by PL/SQL -

- Statement level trigger.
- Row-level trigger.

In statement level trigger, trigger body is executed only once for DML statement, whereas in row level trigger, trigger body is executed by each row for DML statement.

**STYL GAI BHUVANAH YERDAM**  
 501 GAI BHUVANAH YERDAM,  
 H.No. 7-1-200/1, Ground Floor,  
 Ram Niwas Towers, Near Sardar Bhawan,  
 Annapet, Hyderabad 500 016

Syntax :-

```

Trigger specification { Create or replace trigger triggername
 { before/after insert/update/delete on tablename
 [for each row] { Trigger events
 [when condition] { Trigger Timing
 [declare]
 { variable declaration, cursors, userdefined exceptions;
 begin
 =====
 [exception]
 =====
 end;

```

Difference between statement level trigger and Row level Triggers -

e.g.: SQL> create or replace trigger t1  
 after update on emp  
 begin  
 dbms\_output.put\_line ('Statement level');  
 end;  
 /

Statement level trigger



testing : SQL> update emp set sal = sal + 100  
 where deptno = 10;

Statement level.

3 rows updated

SQL> update emp set sal = sal + 100  
where deptno = 80;  
Statement level  
0 rows updated

SQL> drop trigger th1.

Row level trigger -

SQL> create or replace trigger th2  
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

SQL> update emp set sal = sal + 100  
where deptno = 80;  
0 rows updated.

### ● ROW LEVEL TRIGGER :-

In row level trigger, trigger body is executed for each row DML Stmt. That's why we are using for each row clause in trigger specification and also DML transaction values are internally stored in two rollback segments qualifiers, these qualifiers :old,:new, are also called as record type variables.

These qualifiers are used in trigger specification, trigger body.

|                    |
|--------------------|
| Syntax :-          |
| : old . columnname |
| Syntax             |
| : new . columnname |

- when we are use this 'qualifiers' in trigger specification then we are not allowed to use ":" in forms of the qualifiers names.

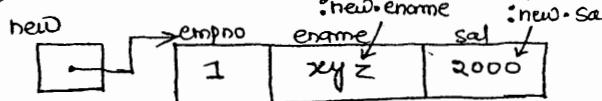
Ex:-

| emp   |       |      |
|-------|-------|------|
| empno | ename | sal  |
| 1     | xyz   | 2000 |

SQL STATEMENT  
T-1200/1, Ground Floor,  
No. 7-1200/1, Ground Floor,  
Ram Mira Towers, New Satyanagar,  
Ameerpet, Hyderabad-500 016.

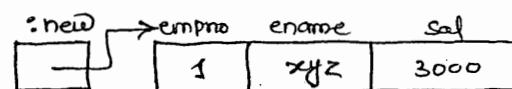
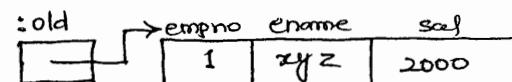
### Insertion:

SQL> insert into emp values (1,'xyz', 2000);



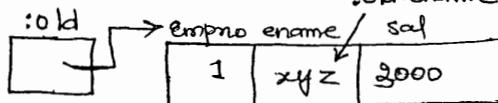
### Update:

SQL> update emp set sal = 3000 where empno=1;



### Deletion:

SQL> delete from emp where empno=1;



### Conclusion:

|      | insert | update | delete |
|------|--------|--------|--------|
| :new | ✓      | ✓      | X      |
| :old | X      | ✓      | ✓      |

Q. Write a pl/sql row level trigger on emp table whenever user inserting data into emp table then salary should be more than 5000?

SQL> create or replace trigger t1

before insert on emp

for each row

begin

if :new.sal < 5000 then

raise\_application\_error(-20123, 'salary should be above 5000');

end if;

end;

Testing:- SQL> insert into emp (empno, ename, sal) values (1, 'abc', 3000);  
o/p: ORA-20123: salary should be above 5000

SQL> insert into emp (empno, ename, sal) values (1, 'abc', 8000);  
o/p: 1 row created.

Q. write a pl/sql row level trigger on following table not allowed to insert duplicate data (it acts like a primary key)?

SQL> create table test (sno number(10))  
SQL> insert into test values (...);  
SQL> select \* from test;

| SNO |
|-----|
| 10  |
| 10  |
| 10  |
| 20  |
| 20  |
| 30  |
| 30  |

SQL> create or replace trigger tJ1  
before insert on test  
for each row  
declare cursor c1 is select \* from test;  
begin  
for i in c1  
loop  
if i.sno = :new.sno then  
raise\_application\_error (-20345, 'not allowed to insert  
duplicate data');  
end if;  
end loop;  
end;  
/

Testing:- SQL> insert into test values (20);

ORA-20345: not allowed to insert duplicate data

SQL> insert into test values (90);  
1 row created.

SQL> select \* from test;

MANO'S ENTERPRISES  
Plot No: 10, Ghati Nagar,  
Ameerpet, Hyderabad.  
Cell: 9125378496

Q. Write a PL/SQL row level trigger on the above table 'not allowed to insert duplicate data without using cursor?

```
SQL> create or replace trigger TJ2
before insert on test
for each row
declare
 v_count number(10);
begin
 select count(*) into v_count from test
 where sno = :new.sno;
 if v_count >= 1 then
 raise_application_error(-20123, 'we can't insert duplicate');
 elseif v_count = 0 then
 dbms_output.put_line('your record inserted');
 end if;
end;
```

/

Testing -

```
SQL> insert into test values(10);
ORA-20123: we can't insert duplicate
SQL> insert into test values (50);
o/p : your record inserted
```

Q. write a PL/SQL row level trigger on the emp table implementing following business rule -

rule - company does not allow any bonus to job as ANALYSTS.

```
SQL> create or replace trigger TJ3
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 values (1, 'ANALYST', 300);  
1 row created

```
SQL> select * from emp;
o/p : No ename comm
 1 ANALYST —
```

MANOJ ENTERPRISES  
Plot No: 40, Gayathri Nagar,  
Ameerpet, Hyderabad.  
Cell: 8125378496

## Updation :-

Q. write a pl/sql row level trigger on emp table whenever user modifying a salary then automatically display old salary, new salary, salary difference ?

```
SQL> create or replace trigger t4
after update on emp for each row
declare z number(10);
begin
z := :new.sal - :old.sal;
dbms_output.put_line('old salary is:'||:old.sal);
dbms_output.put_line('new salary is:'||:new.sal);
dbms_output.put_line('difference on salary is:'||z);
end;
/
```

### Testing:-

```
SQL> update emp set sal = sal + 100 where ename = 'SMITH';
```

O/P:-      Old salary is : 2650  
              New salary is : 2750  
              difference on salary is : 100

Q. write a pl/sql row level trigger on dept table whenever user modifying deptno on dept table then automatically those modification is effected in emp table ?

```
SQL> create or replace trigger t5
after update on dept
for each row
begin
update emp set
deptno = :new.deptno where
deptno = :old.deptno;
end;
/
```

Testing:-    SQL> update dept set deptno = 1 where deptno = 10;

```
SQL> select * from emp;
```

```
SQL> select * from dept;
```

## Deletion:-

Q. write a pl/sql row level trigger using emp, dept table implement undelete cascade concept without using ondelete cascade clause ?

```
SQL> create or replace trigger tk2
 after delete on dept
 for each row
 begin
 delete from emp where
 deptno = :old.deptno;
 end;
 /
```

Testing:-

```
SQL> delete from dept where deptno=10;
SQL> select * from dept;
SQL> select * from emp;
```

## Conclusion -

|      | Insert | update | Delete |
|------|--------|--------|--------|
| :new | ✓      | ✓      | ✗      |
| :old | ✗      | ✓      | ✓      |

## Row level trigger Application:-

### 1) auditing a column -

In all databases whenever we are modifying columns data then, those transaction are automatically stored in another table i.e called auditing a column.

In oracle we are implementing auditing a column by using update of clause in trigger specification.

|                       |
|-----------------------|
| Syntax :-             |
| update of column name |

Q. write a pl/sql row level trigger on emp table whenever user modifying salary then those transaction are automatically stored in another table ?

```
SQL> create table target (old.empno number(10), oldename varchar2(10),
 old.sal number(10), newsal number(10), transdate date,
 username varchar2(10));
```

```
SQL> create or replace trigger tk3
 after update of sal on emp
 for each row
 begin
 insert into target
 values (:old.empno, :old.ename, :old.sal, :new.sal,
 sysdate, user);
 end;
 /
```

Testing:- SQL> update emp set sal = sal + 100 where deptno = 10;  
SQL> select \* from target

#### NOTE:-

In oracle we are not allowed to use :old, :new qualifiers in front of the sysdate, user functions.

#### \*\*\* Auto Increment:-

In all databases generating primary key value automatically is called auto increment concept.

In oracle we are implementing auto increment concept by using row level triggers, sequences. i.e. here we are creating sequence in sal and use this sequence in pl/sql row level trigger.

```
SQL> create table test (sno number(10) primary key, name varchar2(10));
```

```
SQL> create sequence s1;
```

```
SQL> create or replace trigger tk4
 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 : XYZ

```
SQL> /
```

Enter value for name : PQR

```
SQL> /
```

Enter value for name : Sudhir Sastry

```
SQL> /
```

Enter value for name : Shalini

```
SQL> select * from test;
```

O/P:-

| SNO | NAME          |
|-----|---------------|
| 1   | XYZ           |
| 2   | PQR           |
| 3   | Sudhir Sastry |
| 4   | shalini       |

### NOTE:

Oracle 11g introduced variable assignment concept when we are using sequences in pl/sql block.

i.e. Here not required to use dual.table, select into clause.

### Syntax -

```
begin
 varname := sequencename.nextval;
end;
```

## oracle 11g

```
SQL> create or replace trigger tk4
```

before insert on test

for each row

```
begin
```

```
:new.sno := s1.nextval;
```

```
end;
```

```
/
```

© 2013, All rights reserved.  
H. R. K. Venkateswaran, General Manager,  
Ganapathy Towers, Near Satyam Theatre,  
Amarpet, Hyderabad-500 016.

→ Generating Primary key values automatically in varchar2 datatype col.

```
SQL> create table test (sno varchar2(20) primary key, name varchar2(10))
SQL> create sequence s1;
SQL> create or replace trigger tJ1
before insert on test
for each row
begin
select 'ABC' || lpad (s1.nextval, 10, '0') into :new.sno from dual,
end;
/
```

Testing :-

```
SQL> insert into test(name) values ('&name');
```

Enter the value for name : ZZZ

```
SQL> /
```

Enter value for name : XXX

```
SQL> /
```

Enter value for name : YYY

```
SQL> select * from test;
```

| SNO              | SNAME |
|------------------|-------|
| ABC 000000000001 | ZZZ   |
| ABC 000000000002 | XXX   |
| ABC 000000000003 | YYY   |

```
SQL> create or replace trigger tJ1
after insert on test
```

For each row

begin

```
select 'ABC' || lpad (s1.nextval, 10, '0') into :new.sno from
dual;
```

end;

/

error: cannot change NEW values for this trigger type.

- trigger timing (before/after)

oracle DML triggers having 2 timing triggers -

- ① before timing
- ② after timing

Whenever we are using before timing internally trigger code executed first then only DML statements are executed.

In oracle whenever we are assigning new values or whenever we are modifying new qualifiers values then we must use before timing otherwise oracle server returns an ~~error~~ error. "cannot change NEW values for this trigger type" because in before timing DML transaction values are not effected directly in database those transaction values are first effected in trigger then only those values are affected in database.

syntax

:new.columnname := value ;

- generally if we want to restrict invalid data entry then also we are using before timing.
- whenever we are use after timing first DML statements are executed within DB then only trigger code is executed.  
generally in oracle whenever we are performing some operation on the table those transactional values are stored in other table or affected on another table then only we are use after timing.

Q. write a PL/SQL rowlevel trigger on emp table, whenever user inserting data into ename column then automatically user inserted data convert into uppercase within ename column?

SQL> create or replace trigger tij

before insert on emp

for each row

begin

:new.ename:= upper (:new.ename);

end;

/

Testing : SQL> insert into emp(empno,ename) values(1,'murali');  
SQL> select \* from emp;

- Statement level trigger :-

In statement level trigger, trigger body is executed only ones per DML statement. In statement level trigger we are not allowed to use old, new qualifiers and also statement level trigger doesn't have for row.

In all databases if we want to implement time compliment appl'd through triggers then we must use statement level trigger

Q. Write a PL/SQL statement level trigger on emp table, not allowed to perform DML operations on (SAT,SUN) ?

```
SQL> create or replace trigger tJ1
before insert or update or delete on emp
begin
if
to_char(sysdate,'DY') in ('SAT','SUN') then
raise_application_error(-20123,'we cannot perform
DML on SAT,SUN');
end if;
end;
/
```

Testing :- SQL> delete from emp where sal > 2000;  
ORA:-20123 : We cannot perform DML on SAT,SUN

\* [SYSDATE: 20-Feb-16 (sat)]

NOTE:-

• We are not allowed to use when clause in statement level trigger.

```
SQL> create or replace trigger tJ2
before insert or update or delete on emp
when
for each row
when (to_char(sysdate,'DY') in ('SAT','SUN'))
begin
raise_application_error(-20123,'we cannot perform dml
on sat,sun');
end;
/
```

### NOTE:-

In oracle we can also convert statement-level triggers into row-level trigger and vice-versa.

And also by default statement level trigger performance is very high compare to row-level trigger.

- Q Write a pl/sql statement level trigger on emp table not to perform DML operations in last day of the month?

```
SQL> create or replace trigger tk1
 before insert or update or delete
 on emp
 begin
 if sysdate = last_day(sysdate)
 then
 raise_application_error (-20123, 'we cannot perform DML
 on lastday of the month');
 end if;
 end;
 /
```

Testing : ~~SQL> delete from emp where deptno=10;~~  
ORA-20123: we cannot perform DML on lastday of the month  
[sysdate - 29-Feb-2016]\*

### Triggering Events:-

(or) Trigger Predicate clauses:-

If we want to perform multiple operations in different tables then we must use triggering events within trigger body. These are inserting, updating, deleting clauses. These clauses are used in statement, row-level trigger. These triggers are also called as trigger predicate clauses.

#### Syntax :-

```
if inserting then
 stmts;
else if updating then
 stmts;
elseif deleting then
 stmts;
end if;
```

Q. write a pl/sql statement level trigger on emp table, not to perform DML operations in any days by using triggering events ?

```
SQL> Create or replace trigger tgi
before insert or update or delete
on emp
begin
if inserting then
raise_application_error(-20123,'we can't Perform insertion');
else if updating then
raise_application_error(-20145,'we can't Perform update');
else if deleting then
raise_application_error(-20150,'we can't Perform deletion');
end if;
end;
```

Testing :

```
SQL> insert into emp values (1,'RAM','CLERK',5000);
error - ORA-20123: we can't perform insertion.
```

- SQL> create table target (msg varchar2(20));  
SQL> create or replace trigger tJ1  
after insert or update or delete  
on emp  
declare  
x varchar2(10);  
begin  
if inserting then  
x := 'rows inserted';  
else if updating then  
x := 'rows updated';  
else if deleting then  
x := 'rows deleted';  
end if;  
insert into target values (x);  
end;

Testing -

```

SQL> insert into emp(empno) values (1);
SQL> insert into emp (empno) values (2);
SQL> update emp set sal = sal + 100
 where deptno = 10;
SQL> delete from emp where empno in (1,2);

SQL> select * from target;

o/p :- m89
 rows inserted
 rows inserted
 rows updated
 rows deleted

```

### ● Trigger Execution Order :-

- ① before statement level
- ② before row level
- \* \*\* ③ after row level
- \* ④ after statement level

eg:-

```

SQL> create table test (sno number(10));
SQL> create or replace trigger th1
 after insert on test
 for each row
 begin
 dbms_output.put_line ('after row level');
 end;
 /

```

```

SQL> create or replace trigger th2
 before insert on test
 for each row
 begin
 dbms_output.put_line ('before row level');
 end;
 /

```

```

SQL> create or replace trigger th3
 after insert on test
 begin
 dbms_output.put_line ('after statement level');
 end;
 /

```

SGT CAI EDUCATIONAL INSTITUTE  
 H.NO. 71-209/1, Ground Floor,  
 Ram Mira Towers, Near Sezai Masjid,  
 Ameerpet, Hyderabad-500 010

```

SQL> create or replace trigger th4
 before insert on test
begin
 dbms_output.put_line ('before statement level');
end;
/
SQL> set serveroutput on;
SQL> insert into test values(70);
O/P :- before statement level
 before row level
 after row level
 after statement level

```

- Follows clause :- (oracle 11g)

oracle 11g introduced follows clause in triggers.  
 generally in all databases we can't control execution order of the trigger when we are using same level of triggers in same table. To overcome this problem, oracle 11g introduced follows clause in triggers specification. which is used to control execution order of the triggers explicitly, when we are using same level of triggers in same table.

Through the follows clause, we can provide guarantee of the follows clause in the trigger.

Syntax -

```

create or replace trigger triggername
before/after insert/update/delete
on tablename

```

[for each row]

[when condition]

[follows another triggername]

[declare]

.....

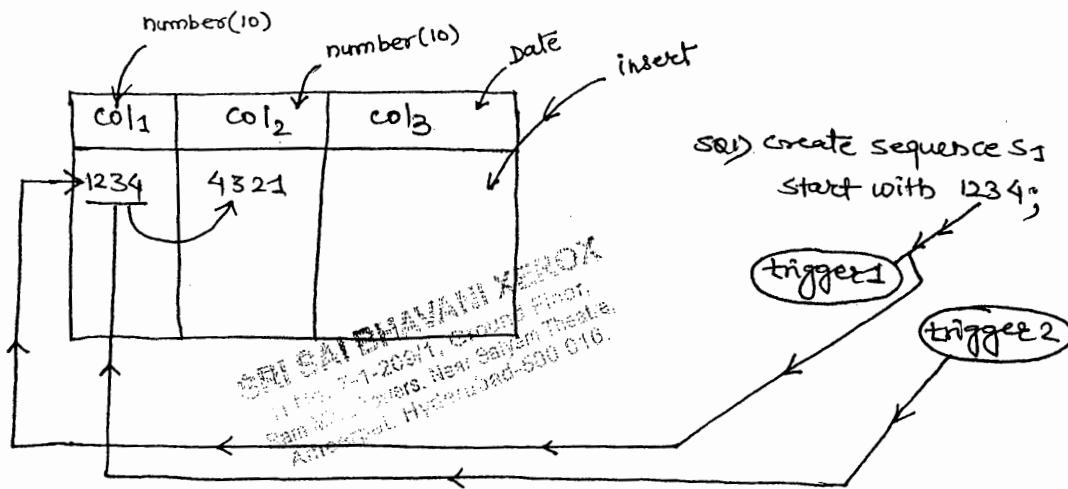
.....

begin

.....

end;

**SRI SAI BHAVANI XEROX**  
 H.No. 7-1-209/1, Ground Floor,  
 Ram Mitra Towers, Near Satyan Theatre,  
 Ameerpet, Hyderabad-500 016.



eg. SQL> create table test (col1 number(10), col2 number(10), col3 date);  
SQL> create sequence s1 start with 1234;

SQL> create or replace trigger tr1

before insert on test

for each row

begin

select s1.nextval into

:new.col1 from dual;

dbms\_output.put\_line ('trigger1 fired');

end;

/

SQL> create or replace trigger tr2

before insert on test

for each row

begin

Select

reverse (to\_char (:new.col1)) into :new.col2 from dual;

dbms\_output.put\_line ('trigger2 fired');

end;

.

/

Testing : SQL> set serveroutput on;

SQL> insert into test (col3) values (sysdate);

O/P: trigger2 fired

trigger1 fired

SQL> select \* from test;

O/P:

| Col1 | Col2 | Col3      |
|------|------|-----------|
| 1234 |      | 19-FEB-16 |

solution (using follows clause)

```
SQL> create or replace trigger tr1
before insert on test
for each row
begin
select s1.resetval into :new.col1 from dual;
dbms_output.put_line ('trigger1 fired');
end;
/
```

```
SQL> create or replace trigger tr2
before insert on test
```

for each row

follows tr1

```
begin
select reverse (to_char (:new.col1)) into
:new.col2 from dual;
dbms_output.put_line ('trigger2 fired');
end;
/
```

Testing -

```
SQL> set serveroutput on;
```

```
SQL> Insert into test (col3) values (sysdate);
```

O/P :- trigger1 fired

trigger2 fired

```
SQL> select * from test;
```

O/P :-

| col1 | col2 | col3      |
|------|------|-----------|
| 1234 | 4321 | 19-FEB-16 |

Q. write a pl/sql trigger on emp table whenever user deleting records from emp table automatically display remaining no. of existing record number in bottom of the delete statement?

```
SQL> create or replace trigger tJ4
after delete on emp
declare
z number(10);
begin
select count(*) into z from emp;
dbms_output.put_line(z);
end;
/
SQL> delete from emp where empno=7901;
o/p : 13
```

ex:- SQL> create or replace trigger tJ4  
after delete on emp  
for each row  
declare  
z number(10);  
begin  
select count(\*) into z from emp;  
dbms\_output.put\_line(z);  
end;  
/  
SQL> delete from emp where empno=1;  
error: ora-4091: table scott.emp is mutating

#### • Mutating :-

Int. a row level trigger based on a table trigger body cannot read data from same table and also we cannot perform DML operation on same table.

If we are trying this oracle server returns an error  
ora-4091: table is mutating.

This error is called mutating error, and this trigger is called mutating trigger, and table is called mutating table.

mutating errors are not occurred in statement level trigger because through these statement level trigger when we are performing DML operations automatically data committed into the database, whereas in row level trigger when we are performing transaction data is not committed and also again we are reading this data from the same table then only mutating errors is occurred.

To avoid mutating errors we are using autonomous transaction in trigger; But autonomous transaction gives previous result.

Sol:- (using autonomous transaction)

```
SQL> create or replace trigger t1
 after delete on emp
 for each row
 declare
 z number(10);
 pragma autonomous_transaction;
 begin
 select count(*) into z from emp;
 commit;
 dbms_output.put_line(z);
 end;
```

/

```
SQL> delete from emp where empno = 2;
```

O/P : 16

- ddl trigger:-

- Database administration only creates ddl triggers, i.e. these trigger are created on schema level.
- These trigger are also called as system triggers because administration creates trigger on DB level.

Syntax: create or replace trigger triggername  
 before/after create/alter/drop/truncate/ rename  
 on username.schema  
 declare  
 → variable declaration;  
 begin  
 ...  
 end;

SRI SAI BHAVAHINI  
H.NO. 7-1-203/1, Ground floor,  
Ram Niwas Towers, Near Satyaniketan,  
Amaripat, Hyderabad-500 016.

Q. write a pl/sql trigger on scott schema not to drop emp table?

SQL> create or replace trigger trs

before drop on scott.schema

begin

if ora-dict-obj-name = 'EMP' and

if ora-dict-obj-type = 'TABLE' then

raise\_application\_error(-20543, 'we can't drop emp table');

end-if;

end;

o/p: SQL> drop table emp;

ora-20543. we can't drop emp table.

- Therefore

- There are 12-types of trigger supported by oracle based on before, statement, row, update, after, delete, insert stat ---> after/before → insert/update/delete and also oracle supports instead of triggers on views and also supports ddl triggers on schema level, system triggers and database level.

Enable/disable triggers:-

Enable/disable single trigger:-

syntax: alter trigger triggername  
 enable/disable

Enable/disable all trigger in a table -

Syntax - alter table tablename  
enable/disable all trigger;

ex: alter table emp disable all trigger;

→ we can also drop trigger using

drop trigger triggername

→ All trigger information stored under user-trigger data dictionary.

sql> desc user-trigger;

## Dynamic SQL

oracle 7.1 introduced dynamic SQL.

It is the combination of SQL, PL/SQL. i.e. SQL statement are executed dynamically with PL/SQL block using execute immediate clause.

Generally in PL/SQL block we are not allow to use DDL, DCL statement. using dynamic SQL DDL, DCL statement within PL/SQL block.

Syntax

begin

execute immediate 'sql statement';

end;

/

ex: begin

execute immediate 'create table X, (sno number(10))';

end;

/

## Passing values into Dynamic SQL Statement -

Through "using" clause we are passing values into dynamic SQL statement.

Here we are passing values either statically or dynamically using placeholders.

In oracle placeholders are represented using colon operator.

Q. write a dynamic SQL program to insert a record in dept table ?

```
SQL> declare
 v_deptno number(10) := 5;
 v_dname varchar2(10) := 'a';
 v_loc varchar2(10) := 'b';
 begin
 execute immediate 'insert into dept values (:1,:2,:3)'
 using v_deptno, v_dname, v_loc;
 end;
 /
```

## Retrieving values from Dynamic SQL statements:-

through "into" clause we are retrieving values from dynamic statement.

Q. write a dynamic SQL program to display no. of records no. from emp table ?

```
SQL> declare
 z number(10);
 begin
 execute immediate 'select count(*) from emp'
 into z;
 dbms_output.put_line(z);
 end;
 /
```

SRI SAI BHAVANI XEROX

H.No. 7-1-309/1, Ground Floor,  
Ramdasp Towers, Banj City, N. Main Road,  
Ameerpet, Hyderabad-500 019.

Q. write a dynamic SQL program to retrieve all employee names from emp table and store it into index by table & also display the content from index by table?

```
SQL> declare
 type t1 is table of varchar2(10)
 index by binary_integer;
 v_t t1;
 begin
 execute immediate 'select ename from emp'
 bulk collect into v_t;
 for i into v_t.first .. v_t.last
 loop
 dbms_output.put_line(v_t(i));
 end loop;
 end;
```

NOTE:-  
when a dynamic SQL statements contains using, into clauses always into clauses precedes using clause.

Q. write a dynamic SQL program for passing deptno 20  
retrieve deptname, loc from dept table?

```
SQL> declare
 v_deptno number(10) := 20;
 v_dname varchar2(10);
 v_loc varchar2(10);
 begin
 execute immediate 'select dname, loc from
dept where deptno = '||v_deptno' into v_dname, v_loc
using v_deptno';
 dbms_output.put_line(v_dname||' '||v_loc);
 end;
 /
```

SRI SAI BHAVANI KENDRA  
H.No. 7-1-2034, Ground floor,  
Ram Mehta Layout, Near A.R.D. College,  
Amarapur, Hyderabad - 500 073.

ph: 9908195958

## Oracle-11g (2007)

- continue statement used in PL/SQL loops.
- read only tables
- simple integer datatype in PL/SQL
- Virtual columns
- Pivot() function
- follows clause in trigger.
- compound trigger
- enable/disable triggers are used in trigger specification
- sequences are used in PL/SQL blocks without using dual tables.

11g introduced read only tables along with alter command.

Syntax alter table tablename read only;

Syntax alter table tablename write read;

Virtual columns. Before 11g if we want to store stored exp<sup>n</sup> in DB we are using function based indexes, view.

But 11g introduced virtual columns using generated always as clause directly storing stored expression within databases.

Syntax: columnname datatype(size) generated  
always as expression [virtual]

ex- SQL> create table W, (a number(10), b number(10), c number(10)  
generated always as (a+b) virtual);

SQL> insert into W, values (10,20);

SQL> select \* from W,;

If we want to view Virtual Expression we are using data-default property from user\_tab\_columns datadictionary.

SQL> desc user\_tab\_columns;

SRI SAI BHAVANI KEPG.  
H.No. 7-1-2007, Ground floor,  
Ram Mohan Towers, Near Satyam Plaza,  
Ameerpet, Hyderabad, 500 016

SQL> select data\_default From user\_tab\_columns where tablename = 'wi';

Simple integer data in PL/SQL.

This datatype performance is very high compare to binary-integer, pls\_integer\_datatype.

This datatype cannot accept null values.

ex:- SQL> declare

```
a simple_integer := 50;
begin
dbms_output.put_line(a);
end;
/
```

Oracle 11g introduced continuous statement in PL/SQL  
Loops as same as c-language continuous statement.

oracle 11g introduced follows clause in triggers.

oracle 11g introduced variable assignment concept when we use sequence in PL/SQL blocks.

oracle 11g introduced enable, disable keyword within trigger specification itself

oracle 11g introduced named, mixed notation in function calling in select statement.

—————p—————p—————

The end ☺

SRI SAI BHAVANI XEROX  
H.No. 7-1-209/1, Ground floor,  
Ram Mirra Towers, Near Satyanarayana  
Ameerpet, Hyderabad - 500 016.

ph: 9908195958  
**MANOJ ENTERPRISES**  
Plot No: 40, Gayathri Nagar,  
Ameerpet, Hyderabad.  
Cell: 8125378496

SRI SAI BHAVANI XEROX  
H.No. 7-1-209/1, Ground floor,  
Ram Mirra Towers, Near Satyanarayana  
Ameerpet, Hyderabad - 500 016.  
Ph: 9908195958

[ I-# ] \*\*\*

WHERE CURRENT OF , FOR UPDATE CLAUSE USE IN CURSOR:-

[ OR ]

UPDATE DELETE STATEMENTS ARE USED IN SONSOLS:-

[ OR ]

CURSOR LOCKING MECHANISM:-

SRI SAI BHAVANI XEROX  
H.No. 7-1-209/1, Ground Floor,  
Ramnara Towers, Near Satyan Theatre,  
Ameerpet, Hyderabad-500 016

- ① write a PL/SQL consol programme to modify salaries of the employees from emp table based on following condition.
- (a) IF job = 'CLERK' Then increment sal → 100
- (b) IF job = 'SALESMAN' Then decrement sal → 200 .

SQL) declare

cursor c1 is select \* from emp;

i emp%rowtype;

begin

open c1;

loop

fetch c1 into i;

exit when c1%notfound;

if i.job = 'CLERK' Then

update emp set sal = sal + 100 where empno = i.empno;

elsif i.job = 'SALESMAN' Then

update emp set sal = sal - 200 where empno = i.empno;

end if;

end loop;

close c1;

end;

MANOJ ENTERPRISES  
Plot No: 40, Gayathri Nagar,  
Ameerpet, Hyderabad.  
Cell: 8125378496

SRI SAI BHAVANI XEROX  
H.No. 7-1-209/1, Ground Floor,  
Ramnara Towers, Near Satyan Theatre,  
Ameerpet, Hyderabad-500 016

Ph: 9908195758

- In all databases whenever we are using delete stmts database servers internally uses default locks
- If you want to establish locks before update before delete then we must use explicit locking mechanism using consols.

- In Oracle if you want to Locks set of Row through Consol Then we must use For Update clause.
  - whenever we are opening The consol Then only opening The cursor internally uses Exclusive locks for update clause is used in consol select stmts only.
- Syntax: consol consolName is select \* from TableName  
where condition for update [Nowait];  
using [or] Not using  
No problem

### WHERE CURRENT OF :-

where current of clause is used to uniquely identifying a record because where current of clause internally uses Rowid where current of clause is used in update delete stmt.

#### Syntax:-

update TableName set of colName = New value  
where current of consolName;

#### Syntax:-

Delete from TableName where current of consolName;

- whenever we are using where current of clause The we must use For update clause in consol Stmt.  
\*(I.②)                                                  { Describe what current of clause.}

#### NOTE:- [This one Tell Interview]

- where current of clause is used To update or delete Latest Fetched Row From the cursor.
- Generally when tables Not having primary key Table data If You want To modify (or) delete Through The cursor Then we must use where current of clause.

- After processing we must release the locks using commit.

- Write PL/SQL cursor programme To increment salary of the Employee From The Following Table without using where current of clause.

```
Sql> Create Table Test1 (ename varchar2(10),sal Number(10));
```

```
Sql> insert into Test values (- - -);
```

```
Sql> select * from Test;
```

| ENAME | SAL  |
|-------|------|
| a     | 1000 |
| b     | 2000 |
| a     | 3000 |
| b     | 4000 |
| c     | 5000 |

```
Sql> declare
```

```
cursor c1 is select*from Test;
```

```
begin
```

```
for i in c1
```

```
loop
```

```
Update Test set sal = sal + 100 where
ename = i.ename
```

```
end Loop;
```

```
/
```

```
Sql> select * from Test;
```

| Name | SAL  |
|------|------|
| a    | 3000 |
| b    | 4000 |
| a    | 5000 |
| b    | 6000 |
| c    | 6000 |

In the above programme Returns wrong Result Because Here where conditional column having duplicate data . to overcome these problem we must used where current of Elsewhere in update statements. whenever we are using where current of clause then clause server updates latestly fetched Row from the cursor If Resource table having duplicated data also because where current of clause internally using Row Id.

~~SQL> declare~~  
cursor C1;

SQL> Rollback;  
SQL> select \* from test1;  

| NAME | SAL  |
|------|------|
| a    | 1000 |
| b    | 2000 |
| a    | 3000 |
| b    | 4000 |
| c    | 5000 |

;

Q. Write a pl/sql program to increment salary of emp. from  $\uparrow$  (using where current of clause) :-

SQL> declare  
cursor C1 is select \* from test1 for  
update;  
begin  
for i in C1  
loop  
update test1 set sal = sal + 1000  
where current of C1;  
end loop;  
commit;  
end  
/

Q1) Select \* from test1;

| ENAME | SAL  |
|-------|------|
| a     | 2000 |
| b     | 3000 |
| a     | 4000 |
| b     | 5000 |
| c     | 6000 |

Q:- Write a PL SQL cursor programme to increment salary of the.

5th Record by 100 in emp table.

SQL declare

(where c1 is select \* from emp for update);

```
begin
 for i in c1
 loop
 if c1%rowcount = 5 then
 update emp set sal = sal + 100
 where current of c1;
 end if;
 end loop;
 commit;
end;
```

Q:- Write a PL SQL cursor programme by using following table updating column C based on following condns.

- 1) For first row update row -> update C=a+b
- 2) For second row -> update C=a-b.
- 3) For third Row -> update C=a\*b.
- 4) For Fourth row -> update C=a/b.

| a | b | C |
|---|---|---|
| 5 | 4 |   |
| 5 | 5 |   |
| 5 | 5 |   |
| 5 | 4 |   |

SQL> Create table test1 (a number(10), b number(10), c number(10));

SQL> insert into test1  
values(5,4, null);

SQL> /

1 row created.

SQL> /

1 row created.

SQL> commit;

SQL> select \* from test1;

| A | B | C |
|---|---|---|
| 5 | 4 |   |
| 5 | 4 |   |
| 5 | 4 |   |
| 5 | 4 |   |

SQL> declare

create c1 is select \* from test1 for update

begin

for i in c1

loop

if c1%rowcount = 1 then update test1

set c = a+b where current of c1;

else if c1%rowcount = 2 then update test1

set c = a\*b where current of c1;

else if c1%rowcount = 3 then update test1 set

c = a+b where current of c1;

elseif c1%rowcount = 4 then update test1 set

c = a/b where current of c1;

end if;

end loop;

commit;

end;

SQL> select \* from test\_1;

| A | B | C  |
|---|---|----|
| S | 4 | 9  |
| S | 4 | 1  |
| S | 4 | 20 |
| S | 4 | 1  |

### LOBs ( Large objects ) :-

Oracle 8.0 introduces <sup>large</sup> LOBs objects These are predefine define datatype which is used to store large amount of data or binary data into database.

In oracle if you want to store more than 2000 bytes of alphanumeric bytes of data then we are using ~~varchar2~~ datatype these datatype stores upto 4000 bytes IF you want to store more then 4000 bytes are alphanumeric data then they are using Long datatype store upto 2 G.B data but there can be only one long column for a table . and also we are not allow to create primary key on long column . to overcome these problems Oracle 8.0 introduces CLOB datatype .

Syntax:- Columnname long

Ex:-

SQL> Create table test1 (col1 long , col2 long );

ERROR:- a table may contain only one column of type LONG .

SQL> Create table test1 (col1 long primary key )

ERROR:- key column cannot be of LONG datatype .

If you want store binary data then we are using raw datatype this datatype stores upto 2000 bytes.

Syntax:-

(columnname raw(size))

If you want to store more then 2000 bytes of binary data then we are using long raw datatype.

These datatype stores upto 2gb data. but there can be one + in a table to overcome these problems Oracle 8.0 introduces "BLOB" datatype.

Syntax:-

(columnname long raw).

SQL> create table test1 (col1 raw (100));

SQL> insert into test1 values ('1010101');

SQL> select \* from test1;

SQL> create table test2 (col2 long raw);

SQL> desc test2;

All database system having two types of large object

1) Internal Large Objects.

2) External Large Objects.

Internal Large Objects are stored within database, all database systems having 2 types of internal large objects.

- 1) Clob (character large object)
- 2) Blob (binary large object)

Syntax:-

(columnname clob)

Syntax:-

(columnname blob)

External Large object:

Are stored in outside of the database. Create having External large object bfile.

Syntax:

columnname bfile

9-4-2016

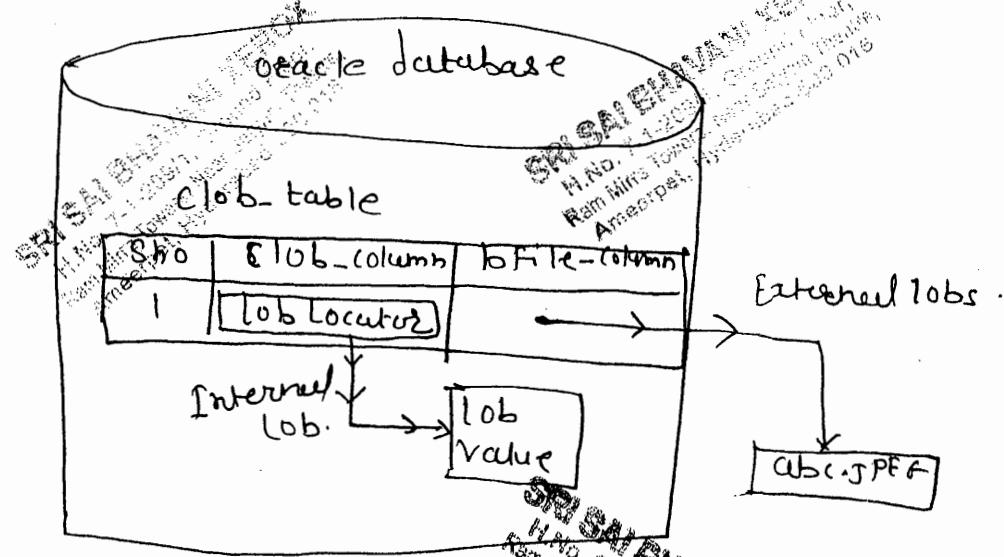
Difference betn long, lob datatypes:-

long

1. It stores upto 2 gb data.
2. A table can contain only one long column.
3. Subquery cannot select a long datatype column.

lob

- 1) It stores up to 4 gb data.
2. A table can contain more than lob columns.
3. subquery can select lob column.



## LOB Locator:

Whenever we are storing data into LOB the data directly not stored same like a other column within database instead of this one a locator is created within table column that locator points to where the actual data stored ~~elsewhere~~ in the database this locator behaves like a pointer. In Internal LOB these locator also called as LOB locator.

## empty\_clob, empty\_blob:-

These predefine funt's are part of the SQL DML these functions are used to initialize LOB locator into empty locator then we are using insert, update

Note:- Before we are start writing data into LOBS using OCI (Oracle Call Interface) or DBMS\_LOB package then we must initialize to LOB locator into empty locator.

## \*Difference b/w null, empty:-

SQL> create table test (sno number(10), col2 blob);

SQL> insert into test values(1,'abc');

SQL> insert into test values(2, empty\_clob());

SQL> insert into test values(3, null);

SQL> select \* from test;

| SNO | NAME |
|-----|------|
| 1   | abc  |
| 2   |      |
| 3   |      |

SQL> select \* from test where col2 is null;

| SNO | COL2 |
|-----|------|
| 3   |      |

SQL) select \* from test where col2 is not null;

| O/P:- | SNO | COL2 |
|-------|-----|------|
|       | --- | ---  |
|       | 1   | abc  |

2   some pointer is there. It displays.

Storing large amount of data into clob column (or) storing an image into blob column by using.

### forms lob package:

Step 1: Before we are handling lobs we must create alias directory Related to physical directory by using following syntax:-

#### Syntax:

Create or replace directory dirname as 'path';

Before we are created alias directory then database Administrator it gives

Create any directory system privilege to use other oracle returns in error.

#### Syntax:

grant create any directory to username;

Step 2: create a relational table by using lob

columns. Syntax:

Syntax: <sup>ex:-</sup> create table tablename (col1 clob, col2 blob);

ex:-

Q1:

SQL> create table test (sno number(10), col2(job));

SQL> conn sys as sysdba;

Enter password:sys

SQL> grant create any directory to scott;

SQL> create or replace directory xyz as 'c:\';

Step3: Develop a pl/sql block which stores data into lobs.

Step1: In declare section of PL block we must declares lob variables.

Syntax:

```
clobvariablename clob;
blobvariablename blob;
bfilevariablename bfile;
```

Step2: Before we are storing data into lobs by using dbms - lob package we must initialise lob locator into non-empty locator by using empty - clob , empty - blob & nulls in insert statement and also using Returns into clause we can store lob locator value into appropriate lob variable. lob locator value into appropriate lob variable. This insert statement must be executed. Section of pl sql block.

Syntax: Insert into tablename values(empty\_lob ()) returning lobcolname into lobvariableename;

SRI SAI GHAVARI WIRE  
H.no. 7-1-213, Opp. KFC, Jyoti  
Narmada Chowk, Nizamuddin, New Delhi,  
Ameerpet, Hyderabad - 500 016

### Step 3:-

before we are using dbms\_lob package we must concatenate actual file with alias directory by using bfilename funt this function accepts 2 parameters and returns bfile datatype.

#### Syntax:-

bfilename := bfilename ('aliasdirectoryname', 'filename');

### Step 4:-

using in load from file procedure from dbms\_lob package int load data into lob column this procedure accept 3 parameters.

#### Syntax:-

dbms\_lob.loadfromfile (lobvariablename, bfilevariable name, length of bfile);

#### Note:-

If you want to ~~not~~ Returns length of the file then we must used getlength funt from dbms\_lob package

#### Syntax:-

dbms\_lob.getlength (bfilename);

### Step 5:-

Before we are using load from file we must open the file by using to procedure and also close the file after processing by using file close procedure from dbms\_lob package.

#### Syntax:-

dbms\_lob.fileclose (bfilename);

#### Syntax:-

dbms\_lob.fileopen (bfilename);

SQL> conn scott/tiger;  
connected.

SQL> declare  
v\_clob clob;  
v\_bfile bfile;  
begin  
insert into test values(1,empty\_clob())  
Returning col2 into v\_clob;  
v\_bfile := bfilename('xyz','file1.txt');  
dbms\_lob.loadFromFile(v\_clob,v\_bfile,dbms\_lob.getlength(v\_bfile));  
dbms\_lob.fileclose(v\_bfile);  
end;  
/

SQL> select \* from test;

Show

O/p: welcome.

Note: In oracle we also store large amount of data  
in image directly into database by using bfile  
datatype but we cannot display bfile in sql plus  
environment.

Ex: Create table test(

SQL> insert into test values(bfilename('xyz','file1.txt'))  
1 row created

SQL> select \* from test;

ERROR: columns an attitude type can't be  
displayed by SQL\*plus.

Date: 11-4-2016

## Member procedure, member Function:

oracle 8.0 introduces object technology i.e we can also use object features in pl/sql.

In oracle also we can create a user defined type by using object.

## Object:-

object is an user defined type which is use to store different datatype into single unit.

Syntax: Create or replace type typename as Object  
( attribute1 datatype (size), attribute2 datatype (size) )

Once we are creating an object , then if we want to store data into the object then we must instantiate that object by using following syntax:-

## Instantiating an object:-

In declare section of the plsql block , we are instantiating an object by using following syntax:-

Syntax: Objectvariablename objecttypename;

## Assign values into object :-

Syntax: Objectvariablename := objecttypename (val1, val2 ...);

Ex:- SQL> Create or Replace type employee as Object  
(empno number (10) , ename varchar2(10));

/

SQL> Set Serveroutput On;

SQL> Declare

e1 employee;

begin

e1 := employee (111, 'abc');

dbms\_output.put\_line ('my employee no is ' || e1.empno || '  
e1.ename)' );

```
dbms_output.put_line ('my employee name is '|| elename);
end;
```

Output:- my employee ~~name~~ is 11  
my employee name is abc

PLSQL also having object type as same as 'C class' in object oriented language. Object type having two parts. They are.

- 1) Type Specification
- 2) type body .

1) type Specification:-

In this we are declaring member data, member subprogram.

2) type body :-

In type body we are implementing member procedure member functions.

Syntax:-

Type Specification :-

SQL > Create or replace type type\_name as object (at + all,  
(at datatype size), ...) member Function declarations

Type body:-

SQL > Create or replace type body type\_name  
as

```
member procedure implementation;
member function implementation;
end;
```

Ex:- Create or replace type circle as object (is number (10),

member procedure p1,

member function f1 return number);  
1

SQL> Create or replace type body circle

as

member procedure p1

is

begin

dbms\_output.put\_line ('my procedure');

end p1;

member function f1 return number

is

begin

return 2\*3.14\*ia;

end if;

end;

1

Instantiating circle object:-

SQL> set serveroutput on;

SQL> declare

c circle;

begin

c := circle(8);

dbms\_output.put\_line (c.f1);

end;

1

O/p:- 50.24

## DYNAMIC SQL:-

Oracle 7.1 introduced 'dynamic sql'. It is comb of SQL, PLSQL.

In dynamic SQL, SQL statements are executed dynamically.  
These SQL statements are specified within PLSQL block  
by using 'execute immediate' clause. In this case  
SQL statement must be specified within single quotes.

Syntax:- execute immediate 'SQL Statement'.

This clause is used in executable section of the PLSQL block.

Generally we are not allowed to use DDL, DCL statement in PLSQL so overcome this problem if want to use these statements in PLSQL blocks then we must use dynamic SQL construct;

```
SQL> begin
 create table test (sho number (10));
 end;
 /
```

Error:

```
SQL> begin
 execute immediate 'create table test
 (sho number(20))';
 end;
 /
```

PL SQL procedure successfully completed.

```
SQL> desc test;
```

write a pl/sql program to create a role

```
SQL> begin
 create role u ;
end ;
/
```

ERROR..

```
SQL> connect sys as sysdba
password: sys
connected .
```

```
SQL> begin
 create role u ;
end
/
ERROR..
```

```
SQL> begin
 execute immediate 'create role u ' ;
end
/
```

PL/SQL procedure successfully completed

Retrieving data from dynamic SQL statements:

Through 'into' clause we can retrieve data from dynamic SQL statements.

Write a dynamic SQL program which is used to display max sal from emp table.

```
SOLN: SQL> declare
 v_sal number(10);
 begin
 execute immediate 'select max(sal)
 from emp'
 into v_sal;
 dbms_output.put_line(v_sal);
 end;
 /

```

OP: 6900

12-4-2016.

Q: Write a dynamic SQL program to retrieve all employee names from emp table and store it into index by table.

By using Bulk collect clause & also display contain from index by table.

```
SQL> declare
 type t1 is table of varchar2(10)
 index by binary_integer;
 v_t t1;
 begin
 execute immediate 'select ename from emp' bulk
 collect into v_t;
 for i in v_t.first..v_t.last
 loop
 dbms_output.put_line(v_t(i));
 end loop;
 end;
```

```
 end loop;
 end;
```

```
/
```

## • Passing values into dynamic SQL statements:-

In dynamic SQL we are not allowed to pass actual values directly into SQL statements within execute immediate clause in place of actual values we must used place holder within SQL statement. In dynamic SQL place holders are represented by using ':' operator. Whenever we are submitting these type of dynamic SQL statements Oracle Server will replace specified value through using clause in place of place holder that's why through the using clause we return specify specify actual values within insert, update, select statements.

Q:- Write a dynamic SQL programme to insert a Record into dept table.

```
SQL> declare
 v_deptno number(10) := &deptno';
 v_dname varchar2(10) := '&dname';
 v_loc varchar2(10) := '&loc';
begin
 execute immediate 'insert into dept values (:1,:2,:3)'
 using v_deptno, v_dname, v_loc';
end;
```

```
/
```

Enter value for deptno: 1

dname: a

loc : b

```
SQL> Select * from dept;
```

SRI SAI BHARATHI AERO  
M.NO. 7-1-200/1, GOWDAPALE, MYSORE  
Ram Mila Towers, Near Laxmi Narayan Temple  
Ambedkarpet, Mysore - 570 002

Note: we can also used using , into clauses , at a time in single dynamic sql statement in these case always into clause @ precedes than the using clause.

Q:- write a dynamic sql programme for user Enter department no.  
= then display dname , loc , from dept table .

```
SQL> declare
 v_deptno number(10) := <>deptno;
 v_dname varchar2(10);
begin
execute immediate 'select dname, loc from dept
where deptno = :1' into v_dname, v_loc
using v_deptno;
dbms_output.put_line(v_dname || '||' || v_loc);
end;
```

/

O/p: Enter value for dept = 10  
NewYork .

## Oracle 11g Features:-

Oracle 11g introduces Read only table through Alter Command this table we can not perform DML Operations

Syntax: alter tablename read only;

Syntax:

alter table tablename read write;

Oracle 11g introduces simple-integer datatype in PL/SQL.

simple-integer datatype internally having not null clause that why we can not accept null values into this datatype whenever we are using this variable we must assign the value at time of variable declaration in declare section of the PL/SQL Block.

\* Syntax:-

VariableName simple-integer := value;

Ex:- SQL> declare

```
a simple-integer := 50;
begin
 dbms_output.put_line(a);
end;
/
```

Note:- simple\_integer datatype performance is very high compare to pls\_integer datatype.

Ex:- SQL> declare

```
a pls-integer;
begin
 a := 70;
 dbms_output.put_line(a);
end;
/
```

3) Oracle 11g introduces continue statement in PL/SQL.  
It is also same as C language continue statements.  
In PL/SQL also continue statement sends control beginning  
of the loop.

Syntax:

continue

Ex: SQL> begin  
for i in 1...10  
loop  
if i=5 then  
continue;  
end if;  
dbms\_output.put\_line(i);  
end loop;  
end;

O/P: 1 2 3 4 6 7 8 9 10 (5 is skip bcz continue)

4) Oracle 11g introduces virtual columns to store stored expressions directly into Oracle database by using generated always as clause.

Syntax:

(columnname datatype(size) generated always as (stored expression) [virtual])

5) Oracle 11g introduces follows clause in triggers.  
It is used to control execute of triggers when we are using same level of triggers.

6) Oracle 11g introduces variable assignment concept  
where we are using sequence in PL/SQL Block in these introduces dual table.

Syntax:

begin  
variableName := sequenceName.nextval;  
end;

- 7) Oracle 11g introduces compound triggers. Oracle
- 8) Oracle 11g introduces name, mixed notations when a sub program is executed by using select statements.
- 9) Oracle 11g introduces PIVOT() which is used to display rows into columns and also allow to display aggregate in tabular forms.

SRI SAI BHAVANI XEROX  
I.I.No. 7-1-203/1, Ground Floor,  
Rani Mira Towers, Near Satyanarayana Temple,  
Ameerpet, Hyderabad-500 016  
ph: 9908195958

SRI SAI BHAVANI XEROX  
I.I.No. 7-1-203/1, Ground Floor,  
Rani Mira Towers, Near Satyanarayana Temple,  
Ameerpet, Hyderabad-500 016  
ph: 9908195958

## Oracle 12c Features:-

1) Oracle 12c introduced invisible columns.

13-04-2016

### Syntax:-

(columnname datatype(size)  
invisible)

### Syntax:-

alter table tablename modify (columnname invisible);

1) Ex:-

sql> create table test (sno number(10), name varchar2(10));

sql> insert into test values(1);

Error:- not enough values

sql> alter table test modify (name invisible);

sql> desc test;

sno number(10)

sql> insert into values (1);

1 row inserted

sql> select \* from test;

## ② Oracle 12c introduced identity columns:-

prior to Oracle 12c if you want to generate primary key value automatically then we are using auto increment concept, in autoincrement concept

we are using sequence & rowlevel triggers. To overcome this problem Oracle 12c introduced new auto increment concept without using rowlevel trigger in PL/SQL.

In this case we are specifying sequence value within a table creation. By using default clause,

Sel + P = u

### Syntax:

columnname datatype (size) default sequence name • next val  
primary key .

Ex:-

create table test (sho number (10) default s1.next val  
Primary key ,ename varchar2 (10));

SQL> insert into test values (ename) values ('& name');

SQL> Enter value for ename : abc

SQL> Enter value for ename : xyz

SQL> select \* from test;

| SNO | Name |
|-----|------|
| 1   | abc  |
| 2   | xyz  |

Note:- Oracle 12c provided another method for autoincrement without using sequence also in this case oracle server only internally automatically create a sequence but these sequence generated sequence values first get stored . If you want to generate primary key value automatically without creating sequence then we must use generated as identity clause that why these column also called as identity column.

Syntax: SQL> column datatype (size) generated identity  
primary key ,

Ex:- SQL> create table test1 (sho number (10) generated  
as identity primary key ,ename varchar2 (10));

SQL> insert into test (name) values ('abc');

SQL> Enter values for name : abc  
Enter values for : xyz

SQL> Select \* from test;

| SNO | NAME |
|-----|------|
| 1   | abc  |
| 2   | xyz  |

③ Oracle 12c introduced truncate table + cascade clause  
For truncating another table record before we are using this clause we must use on delete cascade clause within child table.

Syntax:-

truncate table tablename cascade;

SQL> create table mas(sno number(10) primary key);  
SQL> insert into mas values (..);

SQL> Select \* From mas;

| SNO |
|-----|
| 1   |
| 2   |
| 3   |

SQL> create table child (sno number(10) References mas on delete cascade);

SQL> insert into child values ('-');  
select \* from child

| SNO |
|-----|
| 1   |
| 1   |
| 2   |
| 2   |

SQL> truncate table emp;

Error: unique / primary key in table referenced by enabled Foreign keys.

### Oracle 12c

SQL> truncate table console;

- 4) Oracle 12c introduced new top-n analysis by ~~select~~  
... First) next clauses.

→ prior to Oracle 12c if you want to write top-n-analysis queries then we must use inline views , rownum . to overcome this problem Oracle 12c introduced new top Analysis without using inline views or rownum .

### Syntax:

Select from tablename order by columnname [ asc / desc ]  
Fetch First / next Rows only ;

- 2) Select \* from emp order by  
SQL> desc fetch first  
5 row only ;

- 3) What query to skip 1st - three and then display next five rows in top-n-analysis from Emp table ?

Select \* from emp order by sal desc  
offset 3 rows fetch next 5 rows only ;

5) Oracle 12c introduced session specific Sequence

SQL> conn scott/tiger

a) SQL> Create sequence S1 start with 1 increment by 1 session

SQL> select S1.nextval from dual;

SQL> conn scott/tiger

9)  
select S1.nextval from dual

(6) Oracle 12c introduced with in PL/SQL local functions  
=> Oracle 9i introduced with clause for improve performance  
loop-n-analysis somelike SQL with clause 12c  
introduced with clause in local function and  
also execute the function immediatly by using  
select statements.

SQL> with function F1 (number) returns number  
is begin

return :a;

end ;

Select F1(4) from dual ; ph: 9908195958

1

16

=> Oracle 12c introduced Accessible by clause in  
in stored Procedures .

SRI SAI BHAVANI XEROX

H.No. 7-1-209/1, Ground Floor,  
Ram Mira Towers, Near Satyan Theatre,  
Ameerpet, Hyderabad-500 016

ph: 99081 95958