

# **ORACLE12c**

**New Notes**

**By**

**MURALI SIR**

**Naresh technologies**

**SRI RAGHAVENDRA XEROX**

**Software Languages Material Available**

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

**Cell:9951596199**



# ORACLE - SQL

15/07/2015

Latest version 12C (Cloud Computing)

Version release in 2013 june 29

- 1) SQL
- 2) PL/SQL
- 3) dynamic SQL

Q) What is database?

Ans: Database is nothing but structured data



## XEROX ARIAN

- Oracle is a product from Oracle Corporation. This product is implemented based on ORDBMS (Object Relational database management system) concepts.
- Oracle is also called as relational database which is used to store data permanently in secondary storage devices. If you want to operate oracle database then we are using SQL, PL/SQL languages.
- All organisation store some type of data.

## Data:-

It is a collection of raw facts

- ex:-
- 1) Student marks
  - 2) Customer names

**SRI RAGHAVENDRA XEROX**  
 Software Languages Material Available  
 Beside Bangalore Ayyagar Bakery,  
 Opp. CDAC, Balkampet Road,  
 Ameerpet, Hyderabad.

## Information:-

- Meaningful data / Process data is also called as information.
- When ever we are processing data then we are achieving meaningful results this is called information.

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

## Data store:-

- It is a place where we can stored data or information in organization

- 1) books & papers
- 2) flat files
- 3) database

**SRI RAGHAVENDRA XEROX**  
 Software Languages Material Available  
 Beside Bangalore Ayyagar Bakery,  
 Opp. CDAC, Balkampet Road,  
 Ameerpet, Hyderabad.

## flat files :-

→ This is a traditional mechanism which is used to store data or information in individual unrelated files. These files are also called as flat files.

## drawbacks of flat files :-

- 1) data retrieval
- 2) data redundancy
- 3) data integrity
- 4) data security
- 5) data indexing

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

### 1) data retrieval :-

If you want to retrieve data from flat files then we must develop application program in high level languages. Whereas if you want to retrieve data from database then we are using SQL language.

### 2) data redundancy:-

16/07/2015

Sometimes we are maintaining multiple copies of the same data in different locations this data is also called as duplicate data or redundant data.

In flat files whenever we are modifying data in one location it is not reflected in another location this is called inconsistency.

17/07/2015

- In databases every transaction internally having 4 properties, these properties are also called as ACID properties. These properties only internally automatically maintain consistent data in databases.
- In databases we can also reduce duplicate data by using normalization process.

### 3) data integrity :-

Integrity means to maintain proper data, if you want to maintain proper data in databases then we are using constraints, triggers.

If you want to maintain valid data in flat files then we must develop application programs in high level languages

#### 4) Data Security :-

Data stored in flat files cannot be secured because files doesn't provide security mechanism whereas databases provides role based security.

#### 5) Data Indexing :-

If you want to retrieve data very fastly from databases then databases provides indexing mechanism whereas flat files doesn't provide indexing mechanism.

Organisation suffering from flatfile mechanism to store data are informations to overcome this problems. Organisation introduce a special software which is used to store data permanently in secondary storage devices. These softwares are also called as dbms softwares.

#### DBMS :- (Database Management System) :-

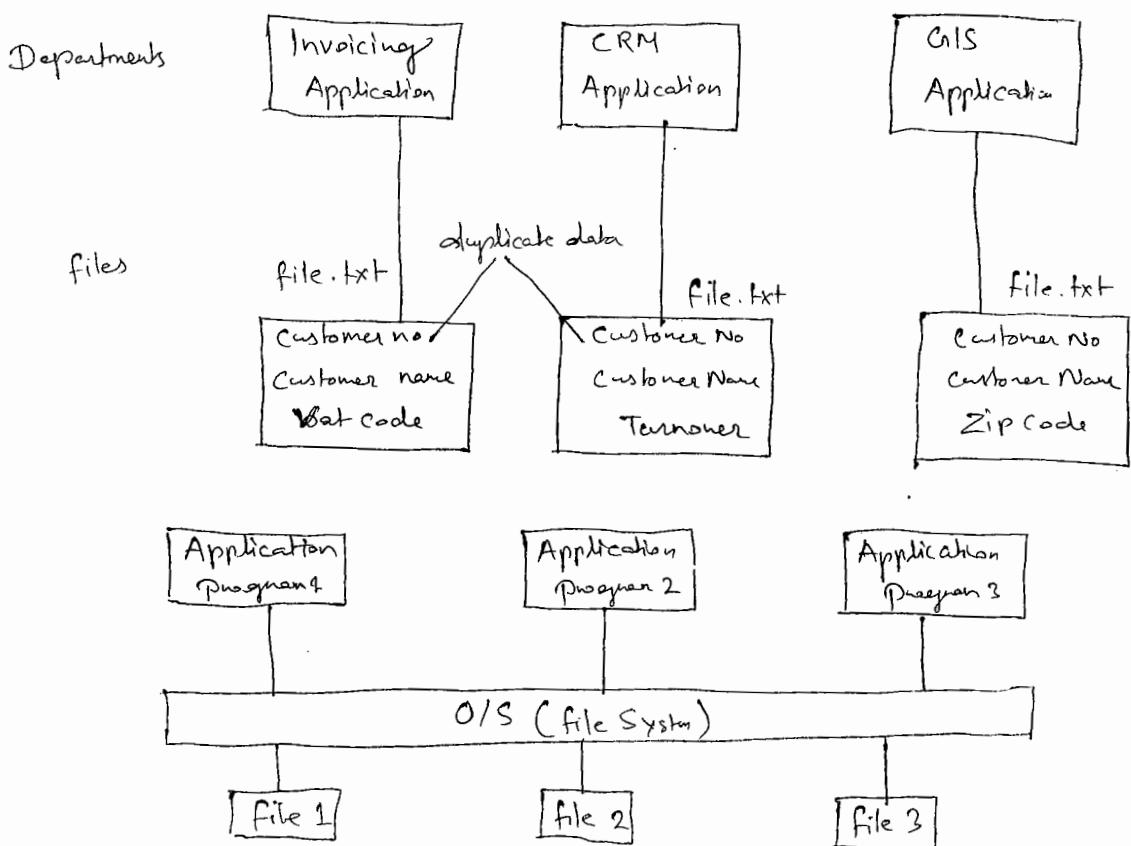
It is a collection of programs (S/W) written to manage database.

ex:- Oracle, teradata, sqlite, mysql, sybase, ingress, informix, db2, Sql Server, ...;

In flatfile mechanism every application program having its own file separate from other application program in the organization

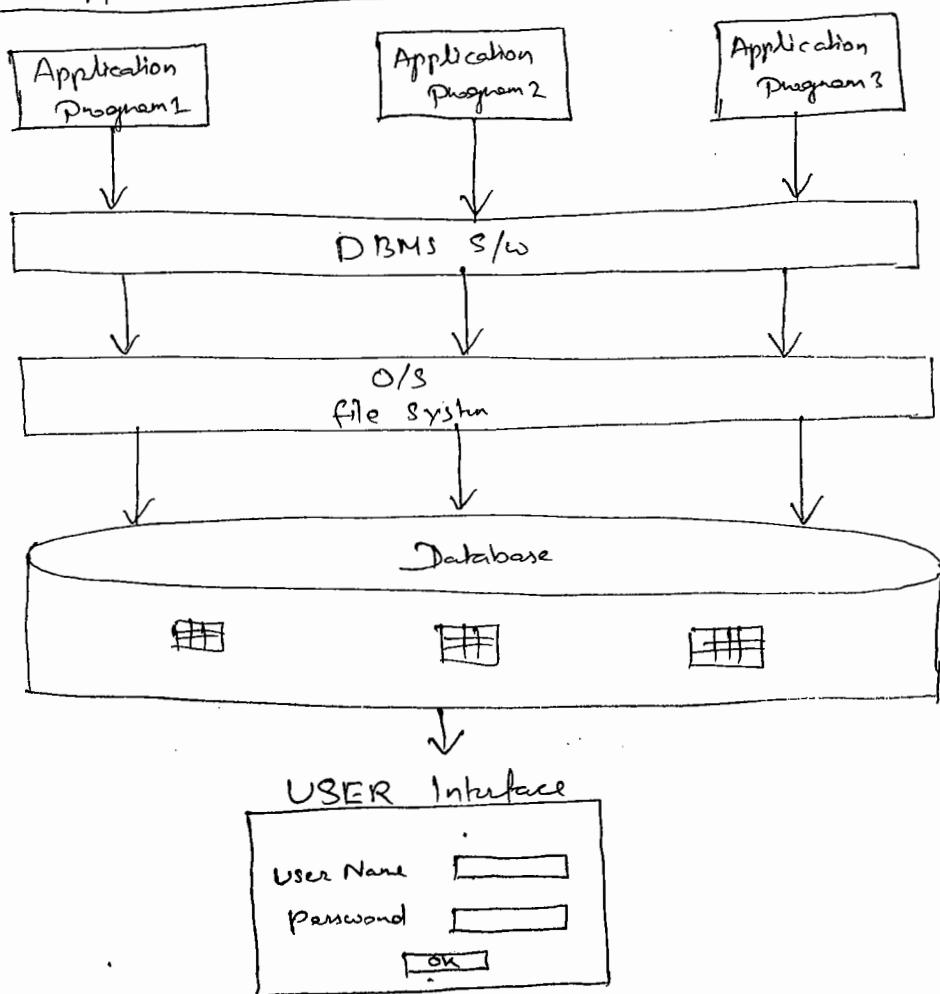
#### Flat file Approach for Data Management

e.X! -



Whenever we are installing dbms software into our system then automatically some place is created in hard disk this is called database and also an user interface is created automatically through this user interface we can directly interact with the database or through the application programs indirectly interacting with the database.

### DBMS Approach for data management



### Database :-

It is an organised collection of interrelated data i.e. database contains structured data.

Every database contains 2 types of structure

- 1) physical Structure
- 2) Logical Structure

A database structure which is visible in a operating system is called physical Structure. Physical structure is handled by database Administration.

### Logical Structure:-

A structure which is not visible in operating system is called logical structure. Logical structure is handled by database developer, database administrators.

20/7/2015

### DBMS Architecture (OR) ANSI/SPARC Architecture (OR) 3 Level Architecture

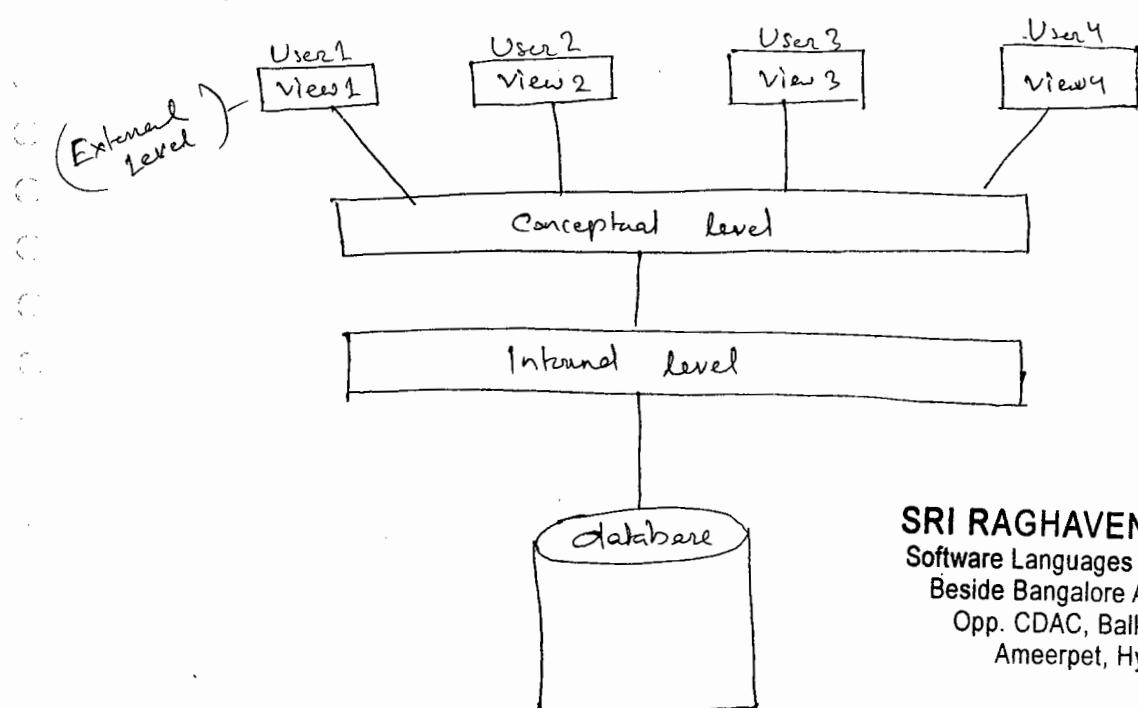
ANSI- (American National Standard Institute) has established 3 level architecture for dbms.

Object of dbms architecture is to separate users view of the database from the way physically it is stored.

3 level architecture contains

- 1) Conceptual level
- 2) External level
- 3) Internal level

This 3 level architecture is also called as ANSI / SPARC (Standard Planning And Requirements Committee)



(DBMS Architecture)

**SRI RAGHAVENDRA XEROX**  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

3 Level Architecture provides data independence.

### DATA Independence:-

Upper levels are unaffected by changes in the lower levels is called data independence. DBMS Architecture having two type of independence.

- 1) Logical <sup>data</sup> independence.
- 2) Physical Data Independence.

#### 1) Logical Data Independence:-

Changes in the conceptual level do not required changes to the external level is called Logical Data Independence.

e.g:- Adding a new entity in conceptual level then it is not effected in external level.

#### 2) Physical Data Independence:-

Changes in the internal level do not required changes to the conceptual level this is called physical data independence.

e.g:- Adding a new entity in internal level it is not effected in conceptual level

### Conceptual level:-

- Conceptual level describe logical structure of the database.
- Conceptual level does not describe how data is physically stored in data base.
- Conceptual level defines what type of data can be stored by specifying data type & sizes. It also defines what type of data cannot be stored in database by specifying constraints.
- Conceptual level ~~also~~ also defines relationship between tables by specifying foreign keys.

e.g create table bank (acc no number (10) primary key, name Varchar2 (10), balance number (10));

bank		
Primary key	acc no	name
	1001	mahi abc
	1002	30,000

## ORACLE DBMS

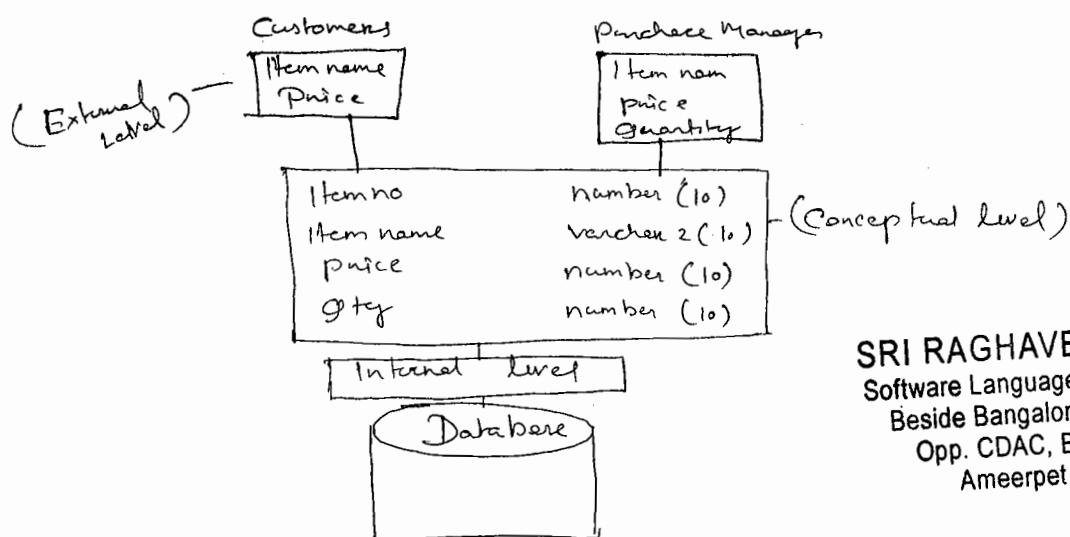
View, Synonym

Table

Index, Cluster

### External level :-

- External level describes user view of the database
- Database contains Large amount of data but some type of users wants to access portion of the data from the database. In this case only database administrator creating a views & then only those views given to no. of users.

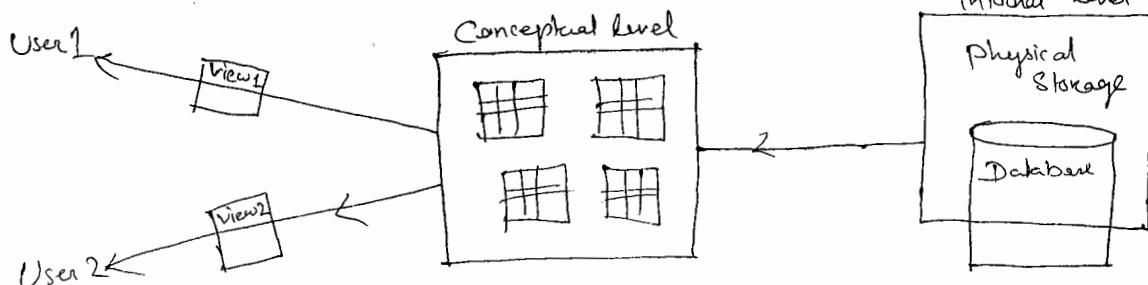


**SRI RAGHAVENDRA XEROX**  
 Software Languages Material Available  
 Beside Bangalore Ayyagar Bakery,  
 Opp. CDAC, Balkampet Road,  
 Ameerpet, Hyderabad.

## DBMS Architecture

21/07/2015

### External Level



### Data Model :-

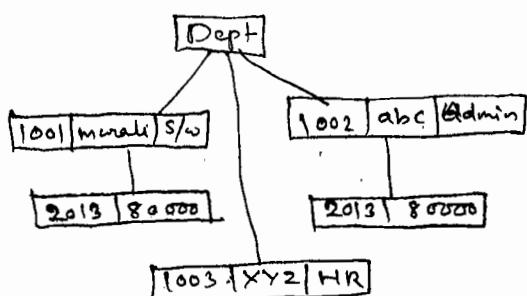
- How data is represented at the conceptual level defined by means of data model.
- In the history of database design three data models have been used there are
  - Hierarchical data model.
  - Network data model.
  - Relational data model.

#### a) Hierarchical data model :-

- In hierarchical data model organises data in tree like structure. this hierarchy is also called parent-child hierarchy. In hierarchical data model also data is represented in the format of records and also return type is also same as table in relational data model.
- Hierarchical data model is implement based on one to many relationship in one to many relationship no. of childs having single parent record only based on this restriction in this data model always child records are repeated that's why this data model having more duplicate data.
- In 1960 IBM introduced IMS (Information management System) product based on hierarchical data model. If you want to operate hierarchical data model product then we are using procedural languages.

22/07/2015

#### Hierarchical datamodel



#### Relational Data Model

Primary key		
empno	ename	Department
1001	murali	S/W
1002	abc	Admin
1003	xyz	HR

#### Foreign key

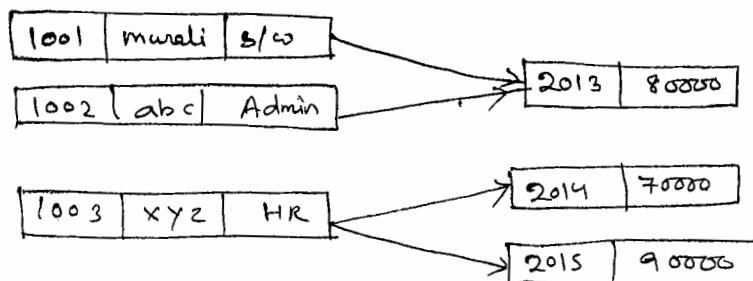
empno	year	Amount
1001	2013	50000
1002	2013	80000
1003	2014	70000
1003	2015	90000

## Network Data Model !-

→ In 1970 CODASYL (Conference on data System language) committee introduced network data model. This data model is implemented based on many to many relationships based on this restriction child segments are not repeated in this data model also data is stored in the format of records & also record type is also same as table in relational data model.

→ In 1970 IBM introduced IDMS (Information Data Management System) product if you want to operate network data model product also then we are using procedural language.

Network data Model (Diagram)



SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

→ In 1970 E. F. CODD introduced relational data model in this data model we are storing data in two dimensional table & also E. F. CODD introduced non procedural language SQL which is used to operate relational data model products.

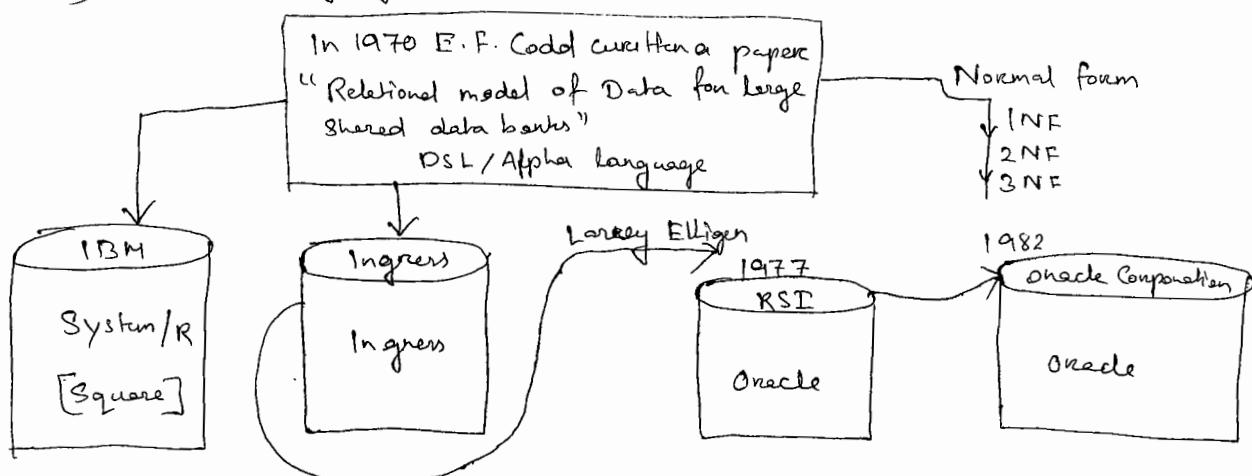
→ Relational data model mainly consists of 3 components. these are

1) Collection of database objects

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

2) Set of operators

3) Set of integrity rules.



## Oracle 12 c latest version

To operate oracle there three language required.

- 1) sql    2) pl/sql    3) Dynamic sql

→ In 1977 Larry Ellison, Bob Miner, Ed Oates founded SSDL (Software Development Laboratories).

→ In 1978 oracle first version was introduced but this version never released.

→ In 1979 sql company name changed into RSI (Relational Software Inc)

→ In 1982 RSI name changed into oracle corporation

## ORACLE Version

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

oracle 5.0 → 1985! → Client server architecture.

oracle 6.0 → 1988! → Introduced pl/sql language.

→ Row level locks

oracle 7.0 → 1992! → Data type varcharc changed into varchar2

→ Stored procedures

→ Stored function

→ Triggers

→ Foreign integrity constraints

→ Truncate command

→ Roles

→ View compilation.

oracle 7.1 → 1994! → Introduced dynamic sql

→ ANSI/ISO SQL92

oracle 7.2 → 1995! → inline views

→ ref cursors (are) cursor variables

→ dbms\_job package.

oracle 7.3 → 1996! → utl\_file package

→ bitmap indexes

oracle 8.0 → 1997! → object technologies

→ lobs (large object)

- instead of triggers
- columns increased per a table upto 1000
- partition tables, indexes

Oracle 8i (1-Internet) → 1999 ! → materialized views

- case conditional statements
- rollup, cube
- trim()
- bulk bind
- function based indexes
- analytical functions
- autonomous transactions

Oracle 9i → 2001 ! → merge statement

- ansi joins (or) 9i joins
- renaming a column
- multitable inserts
- flashback queries
- nvl2(), nullif()

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

Oracle 10i (grid technology) → 2003 ! → recursive

- flashback table
- indices of clause
- wm-concat()
- regular expressions

Oracle 11i → 2007 ! → Introduced continue statement in pl/sql loops

- read only tables
- virtual columns
- pivot() function
- introduced simple\_integer datatype in pl/sql
- compound triggers
- follows clause in pl/sql triggers
- enable, disable clause used in pl/sql trigger specification
- sequence once used in pl/sql without using dual table.
- named, mixed notations used in a subprogram executed using select statement.

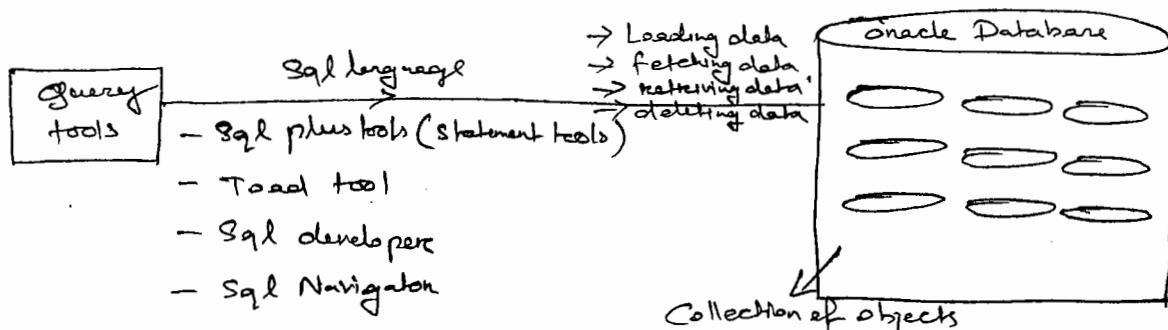
Oracle 12c → 2013 ! → invisible columns

- identity columns
- default values
- sequence session

- top-n query → fetch first clauses
- truncate table cascade
- sql with function

### Sql ( Structured Query Language ) :-

- 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 DDL Alpha Language which is used to operate relational databases
  - Later IBM company system/R team is introduced simple file version of DDL/Alpha called square. Again IBM changed square into sequel (Structured English Query Language). Again IBM changed sequel into sql
  - Sql is an non procedural language which is used to operate all relational databases
- In 1986 → ANSI SQL  
 In 1987 → ISO SQL  
 ANSI/ISO SQL 89 → SQL 89  
 ANSI/ISO SQL 92 → SQL 92  
 ANSI/ISO SQL 99 → SQL 99  
 ANSI/ISO SQL 2003 → SQL 03



### Oracle 10G, 11G, 12C edition (Enterprise edition)

User Name   
 password   
 error → Account Locked  
 Username  ↳  
 password   
 After user scott account unlock ↳

Sql> alter user scott account unlock; ↵

Sql> conn scott/tiger ↵

Enter password : tiger

Confirm password: tiger

To clear Screen :-

Command → cl scr; / (shift + delete) → as key board short cut.

To view all table :-

Command → select \* from tab;

DEPT → master table

EMP → child table

To view particular table :-

Syntax :-

e.g :- Select \* from table name;

Select \* from dept;

Select \* from emp;

## SQL

1) Data Definition Language (DDL) :-

- create
- alter
- drop
- truncate
- rename (concate a i)

2) Data Manipulation Language (DML) :-

- insert
- update
- delete
- merge (concate a i)

3) Data Retrieval Language / Data Query Language (DRL/ DQL) :-

- select

4) Transaction Control language (TCL) :-

- Commit
- Rollback
- save point

5) Data Control Language (DCL) :-

- grant
- revoke

## Data types

24/07/2015

→ Datatypes identifies type of data within a table column. Oracle having following datatypes.

- 1) number (P, S)
- 2) char / varchar (max-size)
- 3) Date

1) Number (P, S) :- P → Precision (total no. of digits)  
S → Scale

It is used to store fixed, floating point nos.

Syntax :-

Column name Datatype (P, S)

e.g:-

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:- When we are trying to store more than the (P-S) no. of digit before decimal point then oracle server returns an error.

e.x 1:- Hence number (7, 2)

P-S =  $7 - 2 = 5$  digits (maximum 5 digits before decimal point)

e.x 2:-

SQL > insert into test values (12345.6789);

SQL > select \* from test;

SNO
12345.68

Note:- Whenever we are using number (P, S) format then we are try to insert more no. of digits after decimal point than the specified scale the oracle server did not give error - In this case oracle server only automatically rounded that number based on the specified scale.

Execution:-

number (7, 2)

Step 1:- 12345.6789

(89) out of 100 above 50%

Step 2:- 12345.67

12345.68

### number (p) :-

→ It is used to store fixed numbers

#### Syntax :-

Columnname number (P)

e.g:-

SQL> create table test1 (sno number (7));

SQL> insert into test1 values (99.9);

SQL> select \* from test1;

-----  
SNO  
---  
100

SQL> insert into test1 values (99.3);

SQL> select \* from test1;

-----  
SNO  
---  
99

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

Note:- In oracle maximum limit of precision is upto 38 digits

### 2) char :-

It is used to stored fixed lengthed Alpha numeric data in bytes maximum limit is upto 2000 bytes by default character datatype having one byte

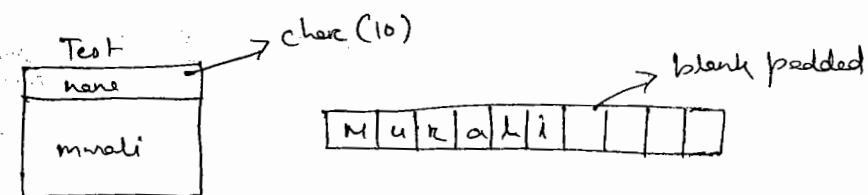
#### Syntax :-

Columnname char (size)

25/07/2015

Whenever we are trying to store less no. of bytes than the maximum size specified in character datatype then oracle server automatically adding blank spaces after end of the character. This mechanism is called blank padded mechanism.

e.g:-



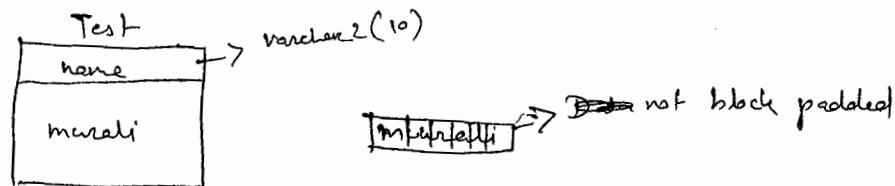
### Varchar2 (max size) :-

Oracle 7.0 introduced Varchar2 datatype it is used to store variable length alpha numeric data in bytes this datatype stores upto 4000 bytes

#### Syntax :-

Columnname Varchar2 (maxsize)

Whenever we are trying to store less no. of bytes than the datatype size specified in Varchar2 datatype then oracle server does not add blank spaces after end of the character this is called not blank padded mechanism.



### Varchar :-

Prior to oracle 7.0 if you want to store variable length alpha numeric data then we are using Varchar datatype. maximum limit of Varchar datatype is upto 2000 bytes. These datatype also store data same as a Varchar2 datatype.

### Syntax:-

Columnname Varchar (size)

### 3) Date !-

- It is used to store date in oracle date format.
- In oracle by default date format is DD-MON-YY

### Syntax:-

Columnname date

### SQL

#### 1) Data Definition Language (DDL)

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

**SRI RAGHAVENDRA XEROX**  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

All DDL commands defines structure of the table

### 1/ Create !-

- It is used to create database objects like tables, Views, sequences, synonyms, indexers.

### Creating a table :-

#### Syntax:-

Create table tablename (column 1 datatype (size), column 2 datatype (size), ...);  
e.g:-

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

### To view structure of the table !-

#### Syntax:-

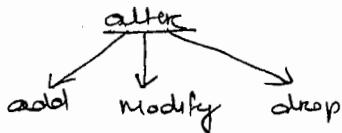
Sql> desc tablename;

e.x!:-

sql> desc first;

## 2) Alter :-

→ It is used to change structure of the existing table



### a) add :-

→ It is used to add no. of columns into existing table

Syntax:-

alter table tablename add (columnname1 datatype (size), ...);

e.x!:-

sql> alter table first add (sal number (10));

sql> desc first;

**SRI RAGHAVENDRA XEROX**  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

### b) Modify :-

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

Syntax:-

alter table tablename modify (columnname1 datatype (size), ...);

e.x!:-

sql> alter table first modify sno varchar2 (10);

sql> desc first;

### c) Drop :-

→ It is used to remove column from to the table.

method 1 → In oracle if you want to drop a single column at a time without using parenthesis then we are using following syntax.

Syntax:-

alter table tablename drop column columnname;

e.x!:-

sql> alter table first drop column sal;

sql> desc first;

method 2 → In oracle if you want to drop single or multiple columns with using parenthesis then we are using following syntax.

Syntax:-

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

e.x!:-

sql> alter table first drop (name);

SQL > desc first;

e.g:-

SQL > alter table first drop column sno;

error! cannot drop all columns in a table.

Note!- In all database system we cannot drop all columns in a table.

27/07/2015

### Drop

It is used to remove database objects from database. In all databases at a time only one object is allowed to drop from a database

Syntax!:-

drop objecttype objectname;

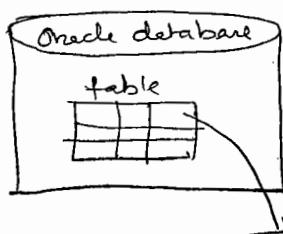
① drop table tablename;

② drop view viewname;

### dropping a table

#### before Oracle 10 g

SQL > drop table tablename;



Permanently removed / deleted

#### Oracle 10g, 11g, 12c [Enterprise Editions]

Syntax!:-

drop table tablename;



get it back from recyclebin.

Syntax!:-

flashback table tablename to before drop;

To drop permanently -

Syntax!:-

drop table tablename purge;

#### example! -

SQL > drop table first;

SQL > desc first;

error! object first does not exist

#### get it back the table:

SQL > flashback table first to before drop;

SQL > desc first;

#### to drop permanently! -

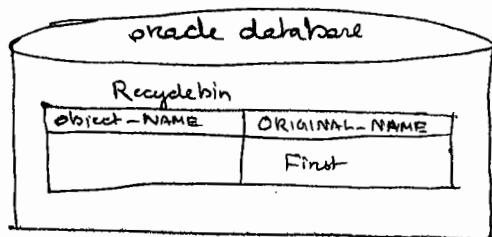
SQL > drop table first purge;

### TestNG !-

SQL> flashback table first to before drop;  
error: object not in recycle bin.

### Recyclebin :-

Recyclebin is a predefined readonly table which is used to stores drop tables oracle 10g only introduced recyclebin table generally whenever we are installing oracle server then automatically so many pre defined tables are created this tables are also called as data dictionary



SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

c. x! -

SQL> Create table first (sno number(10));  
SQL> drop table first;  
SQL> desc recyclebin;  
SQL> select ORIGINAL\_NAME from recyclebin;

ORIGINAL\_NAME  
FIRST

Note:- Same like a windows recyclebin we can also drop objects from recyclebin.

Syntax:-

drop a single table at a time from recyclebin

Syntax :- purge table tablename;

drop all table from recyclebin

Syntax :- purge recyclebin;

Recyclebin purged.

### 4/Truncate!-

Oracle 7.0 introduced truncate table command . when ever we are using truncate command total data is permanently deleted from table. ~~Syntax~~

Syntax:-

truncate table tablename;

E.g:-

```
SQL> Create table test as select * from emp;
SQL> Select * from test;
SQL> truncate table test;
```

Testing :-

```
SQL> Select * from test;
no rows selected
SQL> desc test;
```

S/ Rename:-

It is used to renaming a table using this command oracle 9i onwards we can also renaming a column.

Renaming a table

Syntax:-

```
rename oldtablename to newtablename;
```

Ex:-

```
SQL> rename test to second;
SQL> desc second;
```

Renaming a column (oracle 9i):-

Syntax:-

```
Alter table tablename rename column oldcolumnname to newcolumnname;
```

Ex:-

```
SQL> alter table emp rename column empno to sno;
```

```
SQL> Select * from emp;
```

Note!- In all database by default all ddl commands are automatically committed.

28/07/2015

Data Manipulation Language (DML) :-

→ These commands are used to manipulate data within a table  
These are

- 1) insert
- 2) update
- 3) delete
- 4) merge (oracle 9i)

1) Insert :-

It is used to insert data into a table

**SRI RAGHAVENDRA XEROX**  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

### method 1:-

#### Syntax:-

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

e.x!:- SQL> create table first (sno number (10), name varchar2 (10));

SQL> select \* from first;  
no rows selected

SQL> desc first;

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

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

SQL> select \* from first;

SNO	NAME
1	murali
2	sachin

### method 2!- (Using substitutional operator (&))

#### Syntax:-

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

**&** → Enter Value for msql

e.x!:- SQL> insert into first values (&sno, '&name');

Enter value for sno: 3

Enter value for name: abc

SQL> /

Enter value for sno: 4

Enter value for name: xyz

SQL> select \* from first;

SNO	NAME
1	murali
2	sachin
3	abc
4	xyz

### SRI RAGHAVENDRA XEROX

Software Languages Material Available

Beside Bangalore Ayyagar Bakery,

Opp. CDAC, Balkampet Road,

Ameerpet, Hyderabad.

### method 3!- (Skipping columns)

#### Syntax:-

insert into tablename (col1, col2, ...) values (value1, value2, ...);

```
SQL> insert into first(name) value ('zzz');
```

```
SQL> select * from first;
```

SNO	SNAME
1	murali
2	Sachin
3	abc
4	xyz
	zzz

## 2/ Update:-

It is used to change data within a table.

Syntax:-

```
update tablename set columnname = new value where columnname = oldvalue;
```

e.x:- SQL> update emp set sal=1000 where ename = 'SMITH';

I now updated

```
SQL> select * from emp;
```

Note:- If you want to insert particular cell value into a table then also we must use update command

e.x:- SQL> alter table first add address varchar2(10);

```
SQL> Select * from emp;
```

SNO	SNAME	ADDRESS
1	murali	
2	Sachin	
3	abc	
4	xyz	

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

```
SQL> Select * from first;
```

SNO	NAME	ADDRESS
1	murali	
2	Sachin	mumbai
3	abc	
4	xyz	

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

I now updated

SQL > Select \* from first;

SNO	NAME	ADDRESS
1	murali	
2	Sachin	
3	abc	
4	xyz	

### 3/ Delete:-

It is used to delete all rows or particular rows from a table

Syntax!:-

[delete from tablename;] → to delete all rows

Syntax!:-

[delete from tablename where condition;] → to delete particular rows

e.x!- SQL > delete from first;

7 rows deleted

option back data:-  
SQL > rollback;

SQL > select \* from first;

SRI RAGHAVENDRA XEROX

Software Languages Material Available

Beside Bangalore Ayyagar Bakery,

Opp. CDAC, Balkampet Road,

Ameerpet, Hyderabad.

29/07/2015

### Difference between delete, truncate!-

- When ever we are using delete from tablename or truncate table tablename then total records are deleted from a table.
- When ever we are using delete from tablename. The deleted records are temporary stored in a buffer we can get it back these deleted rows by using rollback (with out using commit)
- When ever we are using truncate command then total data is permanently deleted from a table because truncate is a ddl command & also all ddl transactions are automatically committed. That's why after truncating data we cannot opt to get back these data by using rollback also.

e.x!- SQL > delete from emp where ename = 'SMITH';

1 row deleted

SQL > Select \* from emp;

### Data retrieval Language (or) Data Query Language:-

In all database if we want to retrieve data from a table then we are using Select statement

Syntax!:-

Select col1, col2, ....

from tablename

where condition  
group by columnname  
having condition  
order by columnname [asc/desc];

- ① select all cols & all rows
- ② select all cols & particular rows → where condition
- ③ select particular cols & all rows  
→ cols, cols2, ...
- ④ select particular cols & particular rows.

Creating a new table from existing table / Copying a table from another table :-

Syntax:-

Create table newtablename as Select \* from existing tablename;  
or

Select \* from existingtablename;

e.g. -

SQL> Create table sleep as select \* from emp;

SQL> Select \* from sleep;

Creating a new table from existing table without copying data! -

Syntax:-

Create table newtablename as select \* from existing tablename where false condition;

e.g. -

SQL> Create table test as select \* from emp where 1=2;

SQL> Select \* from test;

no rows selected

SQL> Select desc test;

Operations used in select statement

- ① Arithmetic operators (\*, /, +, -)
- ② Relational operations / Comparison operations (=, <, <=, >, >=, <> or !=)
- ③ Logical operations (AND, OR, NOT)
- ④ Special operators :

where condition

It's also used in where condition.

Select - col1, col2.

Arithmetic operators are used in number, ~~text~~ datatype column.

Q) Write a query to display ename, sal, annsal from emp table?

SQL> Select ename, sal, sal\*12 annsal from emp;

ename      sal      annsal  
-----    -----  
smith      1000      12000

Column aliasname

Q) Write a query to display the employees except job as clerk from emp table?

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

Syntax:-

Select \* from tablename where columnname operator value;

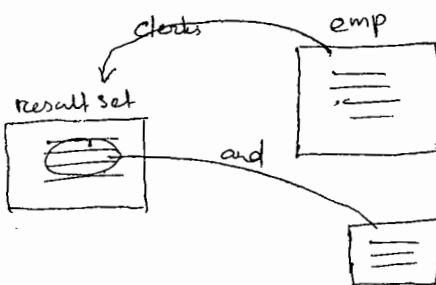
Q) Write a query to display the employees who are getting more than 2000 salary from emp table?

SQL> Select \* from emp where sal > 2000;

30/07/2015

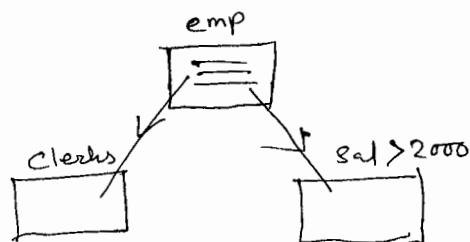
→ If you want to specify multiple condition on where clause then we are using logical operators. When ever we are using logical operator AND database server filter the data from resultset.

SQL> Select \* from emp where job = 'CLERK' and sal > 1500



**SRI RAGHAVENDRA XEROX**  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

SQL> Select \* from emp where job = 'CLERK' or sal > 1500



→ If you want to filter data from table based on number of condition then we are using 'OR' operator.

Q) Write a query to display the employees who are working job as clerk, salesman?

SQL> Select \* from emp where job = 'CLERK' or job = 'SALESMAN'.

Q) write a query to display the employees who are belongs to the department nos 20, 30, 50, 70, 90 from emp table

Ans) `SQL> select * from emp where deptno = 20 or deptno = 30 or deptno = 50 or deptno = 70 or deptno = 90;`

### Special Operators:-

① In	not in
② between	not between
③ is null	is not null
④ Like	not like

i) In :- It is used to point pick the value one by one from list of values.

We can also use in operator in place of are operator when we are trying to retrieve multiple values from a single column in this case 'in' operator performance is very high compare to 'OR' operator

Syntax:-

`Select * from tablename where columnname in (val1, val2, ...)` list of values.  
↳ any datatype!

e.g! -

`SQL> select * from emp where in(20, 30, 50, 70, 90);`

Q) write a query to display Smith, King employee details from emp table?

Ans) `SQL> Select * from emp where ename in ('SMITH', 'KING');`

Note:- In all databases 'not in' operator does not work with 'null' values

e.g! -

`SQL> select * from emp where deptno not in (10, 20, null);`  
null rows selected

Null! - Null is a undefined, unavailable, unknown value, it is not same as zero.

In all database many arithmetic operations performed on NULL values again it <sup>will</sup> becomes

Null

e.g! -  $\text{null} + 50 \Rightarrow \text{null}$

Q) write a query to display ename, sal, commission, sal+commission of the employee SMITH from emp table?

ANS: SQL> Select ename, sal, comm, sal+comm from emp where ename = 'SMITH'

ENAME	SAL	COMM	SAL+COMM
SMITH	1400	---	---

→ In arithmetic operations performed on null values again it will becomes null to over come this problem oracle introduced.(NVL()) function.

NVL() :- NVL is a predefined function which is used to substitute / replace user defined value in place of 'null'.

Syntax:-

`nvl(expr1, expr2)`

→ Here expr1, expr2 must belongs to same datatype.

If expression1 is null then it will returns expression 2 otherwise it will return expression1

- e.x:-  
 1) `nvl(null, 30) => 30`  
 2) `nvl(20, 30) => 20`

SRI RAGHAVENDRA XEROX  
 Software Languages Material Available  
 Beside Bangalore Ayyagar Bakery,  
 Opp. CDAC, Balkampet Road,  
 Ameerpet, Hyderabad.

Solution:-

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

ENAME	SAL	COMM	SAL+COMM
SMITH	1400	1400	---

$$\begin{aligned}
 & \text{sal} + \text{nvl}(\text{comm}, 0) \\
 \Rightarrow & 1400 + \text{nvl}(\text{null}, 0) \\
 \Rightarrow & 1400 + 0 \\
 \Rightarrow & 1400
 \end{aligned}$$

31/07/2015

NVL2() :- Oracle also introduced nvl2(). NVL2(2) accepts 3 parameter

Syntax:-

`nvl2(expr1, expr2, expr3)`

→ If exp1 is null then it will returns expression3 otherwise it returns exp2

- e.x:-  
 1) `nvl2(null, 20, 30) => 30`  
 2) `nvl2(10, 20, 30) => 20`

Q) Update the emp commission as following condition in emp table.

ANS: → If commission → null then update commission → 500.

2) If comm → not null then update comm → 500  
(Using nvl-2(function))

nvl2(comm, comm+500, 500)

SQL> update emp set comm=nvl2(comm, comm+500, 500);  
SQL> select \* from emp;

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

Syntax!-

Select \* from tablename where columnname between lowvalue AND highvalue;

e.g!-

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

SQL> Select \* from emp where sal not between 2000 and 5000;

is null, is not null;-

In oracle is null, is not null operators used in where clause only. These operators are used to test whether a column having null values or not.

Note!- In all databases are are not allowed to use comparison operator = alongwith null value in where condition to overcome this problem oracle introduced is null special operator

Syntax!-

Select \* from tablename where columnname is null;

Syntax!-

Select \* from tablename where columnname is not null;

Q) write a query to display the employees whose display commission is null from emp table.

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

Q) write a query to display the employees whose commission is not null from emp table.

SQL> Select \* from emp where comm is not null;

Like!-

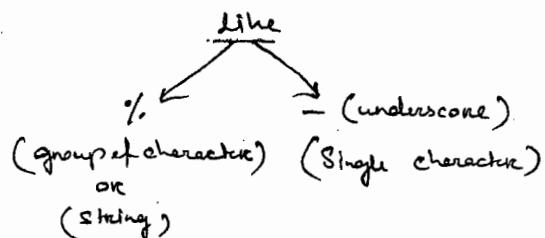
01/08/2015

→ Like operator is used to search data & also retrieves data based on character pattern.

→ Like operator performance is very high compare to searching functions.

→ Along with the like operator we are using two special characters. These are %, \_ (underscore) → wild card characters.

→ These two special characters are also called as wild card characters.



Syntax:-

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

Q) write a query to display the employees whose ename start with M from emp table by using like operator?

SQL> Select \* from emp where ename like 'M%';

MARTIN  
MILLER

In this place we are using %.

Q) write a query to display the employees whose ename contains M from emp table by using like operator? (Any letter M)

SQL> Select \* from emp where ename like '%M%';

SMITH  
MARTIN  
ADAMS  
JAMES  
MILLER

Here M is in different places in first, second, middle last any where.

Q) write a query to display the employee whose ename second letter would be L from emp table by using like operator?

SQL> Select \* from emp where ename like '\_L%';

ALLEN  
CLERK  
BLAKE

Here L is in second place

Q) write a query to display the employee whose ename 4th letter is L from emp table by using like operator?

SQL> Select \* from emp where ename like '\_\_\_L%';

MILLER

Q) write a query to display the employees whose ename start with S\_ from emp table by using like operator?

SQL> Insert into emp (eno,ename) values (1,'S\_SMITH');

SQL> Select \* from emp;

SQL> Select \* from emp where ename like 'S\_%';

ENAME  
---  
S\_SMITH  
SMITH  
SCOTT

→ When ever we are using wild card character within column data then if you want to

return data based on those wild card character then database server returns wrong result because in this case wild card character meaning does not change to overcome this problem if you want return current accurate data and then

→ ANSI/ISO SQL introduced escape function along with like operator whenever we are using escape function then database server automatically escapes special meaning of the escape character wild card character in this case along with escape function we must use any escape character these escape character must be within single quote after escape function. By default length of escape character is 1 byte.

Syntax:-

Select \* from tablename where columnname like 'character pattern' escape 'character escape character';

→ These escape character must be used within character pattern before wild card character. whenever we are using this one then database server changes special meaning of the wild card character.

SQL> Select \* from emp where ename like 'S<sub>1</sub>-%' escape '\_';  
S-MITH } - output will display.

? → underscore treated as underscore, not treated as wild card character.

Q) write a query to display the employees whose ename starts in second letter contains -- underscore underscore those employee names from emp table by using like operator ?

SQL> insert into emp(empno, ename) values (2, 'S--MITH');  
SQL> select \* from emp;  
SQL> select \* from emp; where ename like 'S<sub>1</sub>@\_@-%' Escape '@';  
S--MITH

Q) → underscore treated as underscore, not treated as wild card character

Q) write a query to display the employees who are joining in the month December from emp table by using like operator ?

SQL> Select \* from emp where hirable like '%DEC%';

Q) write a query to display the employees who are joining in the year 81 from emp table by using like operator ?

SQL> Select \* from emp where hirable like '%81';

Concatenation operators :- (ii) → double pipes

03/08/2015

→ If we want to display column data along with string then we are using concatenation operator.

e.x:- select 'my employee name are' || ename from emp;

Output

my employee name are SMITH

my employee name are ALLEN

---

→ Using this operator we can also display our own spaces in both the column & also allowed to display string in both columns.

e.x:- SQL> Select ename || ' ' || sal from emp;

Functions :-

→ Functions are used to solve particular task and also function must return a value

→ Oracle having two types of functions

1) Pre-defined Functions

2) User-defined Functions

→ In oracle userdefined functions are created in PL/SQL only

1) Predefined functions :-

- |   |  |
|---|--|
| 1 | → Number functions                       |
| 2 | → character functions                    |
| 3 | → Data functions                         |
| 4 | → Group functions or Aggregate functions |

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

1) Number functions:- These functions are operate over number data.

abs():- It is used to convert negative sign into positive sign.

e.x!- SQL> Select abs (-50) from dual;

Output → 50 (positive no.)

dual!- → dual is an predefined virtual table which contains one row & one column. dual table is used to test predefined, user defined function functionality.

e.x!- SQL> Select nvl (null, 70) from dual; → output - 70

SQL> Select nvl (30, 70) from dual; → output - 30

→ Through the dual table we can also perform mathematical expression.

e.x!- SQL> Select 20+40 from dual; → output - 60

→ By default dual table column datatype is VARCHAR2

e.x! SQL> Select \* from dual; → output - D  
SQL> desc dual;

SQL) Select ename, sal, comm, abs(comm-sal) from emp where comm is not null;  
mod (m,n)!- It will gives remainder after m is divided by n.

e.x!- SQL) Select mod (10, 5) from dual; → output - 0.

\* round (m,n)!-

It rounds even floated value numbers m based on n.

e.x!- SQL) Select round (1.8) from dual; → output - 2.

SQL) Select round (1.23456, 3) from dual

Output  
1.235

Execution!-

Step 1:- 1.234

out of 100  
56 above 50%.

Step 2:- 1.234  
1  
1.235

Note!- Always round check remaining no. if remaining no. is above 50% then one added to the rounded no.

e.x SQL) Select round (1285.456, -1) from dual; → output - 1290

Execution!-

Step 1:- 1280 .5 is replaced with 0 because 5 is above 50%.

Step 2:- 1280

1  
1290

SQL) Select round (1295, -2) from dual; → output 1300

Execution!-

Step 1:- 1200 95 is replaced with 0 as 95 above 50%.

Step 2:- 1200

1  
1300

e.x!- SQL) Select ename, round (sal, -3) from emp;

trunc (m,n)!-

07/08/2015

In truncate given floated value no. m based on n.

e.x!- SQL) Select trunc (1.9) from dual;

output - 1

SQL) Select trunc (1.23456, 3) from dual; → output - 1.234

Ceil (), floor ()!-

This functions always return integer ~~and~~ ceil returns nearest greatest integer where as ~~floor~~ floor return nearest lowest integer.

e.x:- SQL> Select ceil (1.3) from dual; Output : - 2

SQL> Select floor (1.9) from dual; Output → 1

Greatest (exp1, exp2, ... expn), least (exp1, exp2, ... expn) :-

greatest returns maximum value among given expression where as least returns minimum value among given expression.

ex!:- SQL> Select greatest (2, 4, 6, 8, 9) from dual; output → 9

\*x!:- SQL> select ename, sal, comm, greatest (sal, comm) from emp where comm is not null;

Character functions!:-

Upper ()!:- It is used to convert or string or column value into upper case.

ex!:- SQL> Select upper ('abc') from dual; → output - ABC

SQL> Select upper (ename) from emp; → output - all ename will display.

Lower ()!:-

e.x!:- SQL> Select lower (ename) from emp;

② Create a query to update all employee name into lower case within emp table ?

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

14 rows updated

SQL> Select \* from emp;

Initcap ()!:- It returns first letter is capital & all remaining letters are small.

ex!:- SQL> Select initcap (ename) from emp; → output will first letter capital & all small.

SQL> Select initcap ('ab cd ef') from dual; → output Ab Cd Ef

Length ()!:- This function always returns no. datatype i.e it counts no. of characters including spaces.

ex!:- SQL> Select length ('AB<sub>space</sub>CD') from dual; → output 5 characters including space.

Substr ()!:- It will extract portion of the string within given string based on last two parameter.

SQL> Select substr ('ABCDEF', 2, 3) from dual; → output → BCD

SQL> Select substr ('ABCDEF', -2, 3) from dual; → Output - FG there is no

SQL> Select substr ('ABCDEF', -5) from dual; → Output - CDE FG<sub>Character after FG</sub>.

Syntax!:-

substr (columnname, Starting position, number of character);  
numbers  
+ve      -ve  
from position →

Q) write a query to display the employees whose ename second letter would be capital LA from emp table by using substr().

↳ SQL Select substr(ename, 2, 2);

SQL> Select \* from emp where substr(ename, 2, 2) = 'LA';  
output → BLAKE  
CLARK

where clause

Q) write a query to display the employees whose ename second letter would be capital AR from emp table by using substr().

↳ SQL> Select \* from emp where substr(ename, 2, 2) = 'AR'; output → WARD  
MARTIN

Q) write a query to display the employees whose length of ename is 5 from emp table.

↳ SQL> Select \* from emp where length(ename) = 5; output → employee length will be displayed.

Note:- In all database system we are not allowed to use group functions in where clause but we are allowed to use number(), character(), date() in where clause.

e.g:- SQL> Select \* from emp where sal = max(sal);  
error: group function is not allowed here.

instr() :-

05/08/2015

In string always returns no. datatype that is it returns position of the the delimiter, position of the character, position of the string cut off given string.

e.g:- SQL> Select instr('ABCDEF', '\*') from dual; → output - 0

SQL> Select instr('ABCDEF', 'C') from dual; → output - 3

SQL> Select instr('ABCDEFGHIJKLMNOPM', 'CD') from dual;  
output → 3

SQL> Select instr('ABCDEF<sup>2</sup>GHIJKLMNOPM', 'CD', -6, 2) from dual;  
output → 3

SQL> Select instr('ABCDEF<sup>2</sup>GHIJKLMNOPM', 'CD', -5, 2) from dual;  
output → 9

Syntax:-

instr {  
Column name or String name, 'String', Searching Position, number of occurrences from position  
+ve -ve  
}

In string function always return position based on the last two parameter but oracle server always counts no. of characters left side first position onwards

### Lpad:-

It will fills remaining spaces with specified character on the left side of the given string here always second parameter returns total length of the string

#### Syntax:-

`Lpad ( columnname on , total length , 'filled character' );`

e.x:- SQL> select lpad ('ABCD', 10, '#') from dual;

output → #####ABCD

SQL> select lpad ('ABCD', 2, '\*') from dual;

output → AB

when ever we are submitting Lpad, Rpad functions then oracle server returns first parameter left side first character onwards upto maximum length specified in second parameter that's why whenever we are specifying less no. of bytes than the no. of character in first parameter then Lpad, Rpad function returns same results.

### Rpad:-

e.x:- SQL> Select Rpad ('ABC', 5, '\*') from dual;

output ABC\*\*

SQL> Select Rpad ('ABC', 2, '\*') from dual;

output AB

### Ltrim():-

It removes specified characters on the leftside of the given string.

#### Syntax:-

`Ltrim ( columnname on , {set of character} );`

e.x:- SQL> Select Ltrim ('SSMISSTHSS', 'S') from dual;

output → MISSTHSS

SQL> Select job, Ltrim (job, 'SM') from emp;

output → Job      Ltrim (Job)

CLERK            LERK

SALESMAN        ALESMAN

MANAGER         ANAGER

### Rtrim():-

e.x:- SQL> Select rtrim('SSMISS THSS', 'S') from dual;

output → SMISS TH

### Trim():-

oracle 8i introduced trim function it is used to remove left & right side of

the specified character.

#### Syntax:-

```
trim('character' from 'string');
```

e.x:- SQL> select trim ('s' from 'SSMISSThSS') from dual;  
output → MISSTh

Note:- we can also convert trim function into Ltrim, Rtrim by using leading, trailing process class.

e.x:- SQL> Select trim (Leading 's' from 'SSMISSThSS') from dual;  
output → MISSThSS  
SQL> select trim (trailing 's' from 'SSMISSThSS') from dual;  
output → SSMISSTh

Note:- Using trim function we can also removes leading, trailing spaces

e.x:- SQL> select length(trim (' smith ')) from dual;  
output → length = 5 trim removed free space.

#### \*translate(), replace () :-

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

e.x:- Select translate ('india', 'in', 'xy'), replace ('india', 'in', 'xy') from dual;

<u>Translate</u>	<u>Replace</u>
xydxa	xydia

Syntax:- translate (string1, string2, string3)

06/08/2015

e.x:- SQL> select translate ('ABCDEF', 'FEDCBA', 123456) from dual;  
output - 654321

SQL> Select replace ('A B C', ' ', 'xyz') from dual;  
output - AxyzBxyzC

e.x:- SQL> Select job, replace (job, 'SALESMAN', 'MARKETING') from emp;  
SQL> select replace ('SSMISSThSS', 'S') from dual;  
output → MITH

Note:- whenever we are not using 3rd parameter in replace function then automatically 2nd parameter specified character permanently removed from the given string.

\* Q) write a query which counts number of occurrences of the character E within the given string 'sleep' by using replace function?  
Ans:- SQL> Select length ('sleep') - length (replace ('sleep', 'e')) from dual;  
output - 2

### Concat (Str1, Str2) :-

It is used to concat given two strings.

e.x:- SQL> Select concat ('ABC', 'XYZ') from dual;  
output - ABCXYZ

### Date functions:-

In oracle by default date format is DD-MON-YY . Oracle server having following date function there are

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

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

### 1) Sysdate:-

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

e.x:- SQL> Select sysdate from dual;  
output - 06-AUG-15

### 2) add\_months() :-

It is used to add or subtract no. of months to the specified date based on second parameter.

#### Syntax:-

add\_months ('date', number)  
                  +ve      -ve

e.x:- SQL> select add\_months (sysdate, 1) from dual;  
output - 06-SEP-15

SQL> select add\_months (sysdate, -1) from dual;  
output - 06-JUL-15

### 3) Last\_day() :-

It will returns last date of the specified month.

#### Syntax:- Last\_day ('date')

e.x:- SQL> Select last\_day ('06-AUG-15') from dual;  
output 31-AUG-15

#### 4) next\_day() :-

It returns next occurrence day from the specified date based on second parameter.

Syntax:- `next_day ('date', 'day');`

e.x!- SQL> Select next\_day (sysdate, 'Fri') from dual;  
output - 07 - AUG-15

#### 5) months\_between (date1, date2) :-

This function always return number datatype i.e It returns ~~no~~ number of months between two specified date. Here date1 must be greater than(>) date2 otherwise this function returns negative value.

e.x!- SQL> Select ename,round (months\_between (sysdate,hiredate)) from emp;

#### Date Arithmetic

- 1) Date + number ✓
- 2) Date - number ✓
- 3) Date1 + Date2 ✗
- 3) Date1 - Date2 ✓

e.x!- SQL> Select sysdate + 1 from dual;

SQL> Select sysdate - 1 from dual;

SQL> Select sysdate - sysdate from dual;

Q) write a query to display first date of the current month by using sysdate, add\_month, last\_day functions?

↳ SQL> Select last\_day (add\_month (sysdate, -1)) + 1 from dual;

07/08/2015

#### Date Conversion functions

- 1) to\_char()
- 2) to\_date()

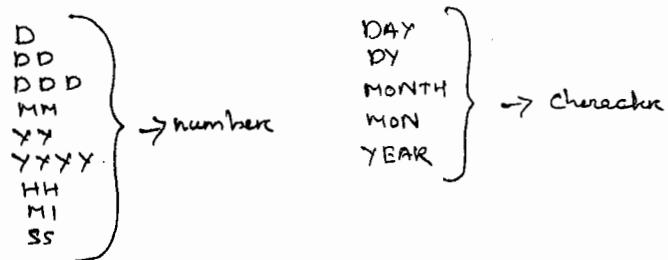
1) to\_char () :- It is used to convert date type into character type i.e It converts date type into date strings.

e.x!- SQL> Select to\_char (sysdate, 'DD/MM/YY') from dual;  
output - 07/08/15

Note:- Basically to\_char character formats ~~of~~ are case sensitive format.

e.x!- SQL> Select to\_char (sysdate, 'DAY') from dual;  
output → FRIDAY ~~07~~

SQL> select to\_char(sysdate, 'day') from dual;  
output - Friday.



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

SQL> select to\_char(system, 'D') from dual;  
output - 6 (→ day of the week (Sunday → 1, Monday → 2, ...))

SQL> select to\_char(system, 'DD') from dual;  
output -

SQL> select to\_char(system, 'DDDD') from dual;

SQL> select to\_char(system, 'DDSPTH') from dual;  
output - SEVENTH SP - SPELL OUT

SQL> Select to\_char(system, 'HH24:MI:SS') from dual;  
output - 11:54:04

SQL> Select to\_char('15-June-05', 'DD-MONTH-YY') from dual;  
output - error

Note!- whenever we are using to\_char() always first parameter must be date type otherwise oracle server returns an error

2) to\_date()!- It is used to convert char type into date type

i.e it converts date string into date type (Oracle date format displayed only)

e.x!- SQL> Select to\_date('23/JUN/05') from dual;  
Output - 23-JUN-05

SQL> select to\_date('23/06/05') from dual;  
Output - error not a valid month.

Note!- whenever we are using to\_date always passed parameter → return value match with the default date format return value otherwise oracle server return an error. To overcome this problem, we can use a second parameter as same as first parameter format then only oracle server automatically convert date string into date type.

sql> select to\_date ('15/06/05') from dual;

Output error - not a valid month → should returns in the default format

e.x:- sql> Select to\_date ('23/06/15', 'DD-MM-YY') from dual;  
Output - 23-JUN-05

e.x:- sql> select to '09-FEB-05' + 5 from dual;  
Output - error: invalid number.

• Solution

sql> select to\_date ('09-FEB-05') + 5 from dual;  
Output - 14-FEB-05  
(or)

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

Q) write a query to display given date string into client requirement format by using to\_char function. Given date is '15-JUN-05' & display format is '15/JUNE/05'

↳ sql> Select to\_char ('15-JUN-05', 'DD/MONTH/YY') from dual;  
Output - error: invalid number

Solution

sql> Select to\_char (to\_date ('15-JUN-05'), 'DD/MONTH/YY') from dual;  
Output - 15/JUNE/05

08/08/2015

fill mode (fm) :-

Whenever we are using to\_char() month, day formats then oracle server returns space when server returns less bytes than the maximum size specified in the date format to overcome this problem if you want to suppress spaces, leading 0 (zero) then oracle introduced fill mode (fm) within to\_char()

e.x:- sql> Select to\_char (sysdate, 'DD/MM/YY') from dual;  
Output → 08/AUGUST/15 Crash here

sql> Select to\_char (sysdate, 'DD/FMMONTH/YY') from dual;  
Output → 08/AUGUST/15

Q) write a query to display the employees who are joining in the month December from emp table by using to\_char().

↳ sql> Select \* from emp where to\_char(hiredate, 'MON') = 'DEC';  
or  
sql> Select \* from emp where to\_char(hiredate, 'MM') = '12';

Ex12!- (using MONTH format)

SQL> Select \* from emp where to\_char(hiredate, 'MONTH') = 'DECEMBER';  
Output → no row selected.

Solution!-

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

Q) Create a query to display the employees who are joining in the year 81 from emp table by using to\_char()

SQL> Select \* from emp where to\_char(hiredate, 'yy') = '81';  
Ex:- showing four digit yyyy

SQL> Select \* from emp where to\_char(hiredate, 'yyyy') from emp;

Q) Note:- In oracle whenever we are passing date string into predefined oracle date functions (ADD\_MONTHS(), Last\_day(), ...) then oracle never internally automatically converts date string into date type that's why here not required to use to\_date() explicitly but here passed parameter must be in oracle date format

e.x!- (Automatic conversion)

SQL> Select last\_day('23-JUN-05') from dual;  
Output → 30-JUN-05

Ex2!- SQL> Select last\_day('23-06-05') from dual;  
Output → not a valid month. (error)

Solution!-

SQL> Select last\_day(to\_date('23-06-05', 'DD-MM-YY')) from dual;  
Output → 30-JUN-05.

Inserting dates into table!-

SQL> Create table test (col1 date);

SQL> Insert into test values (sysdate);

SQL> Select \* from test;

Output →  
Col1  
---  
15-AUG-07

SQL> Select col1 insert into test values ('15-08-07');  
Error! not a valid month

Solution!-

SQL> Insert into test values (to\_date('15-08-07', 'DD-MM-YY'));  
1 row inserted.

SQL> Select \* from test;

Output → Col1  
---  
15-AUG-07

**SRI RAGHAVENDRA XEROX**  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

## round(), trunc() functions used in date!-

10/08/15

→ In oracle date data type contains both date & time portion whenever we are using round, trunc() functions then date part can be changed based on the time portion & also time portion automatically set to zero(0).

e.g:- SQL> Select to\_char(sysdate, 'DD-MM-YY hh24:mi:ss') from dual;  
output → 10-AUG-15 12:31:32

→ whenever we are using round function then oracle server automatically changes one date to the given date if time portion is  $\geq 12\text{ noon}$ , and also time portion is automatically set to zero(0).

e.g:- SQL> Select to\_char(round(sysdate), 'DD-MM-YY hh24:mi:ss') from dual;  
output → 11-AUG-15 00:00:00

→ whenever we are using trunc() function then oracle server automatically returns same date if time portion is  $\geq 12\text{ noon}$  also and here also internally time portion is automatically set to zero(0).

e.g:- SQL> Select to\_char(trunc(sysdate), 'DD-MM-YY hh24:mi:ss') from dual;  
output → 10-AUG-15 00:00:00

Q) write a query to display the employees who are joining today from emp table?

↪ SQL> insert into emp (empno, ename, hiredate) Values(1, 'murali', sysdate);

SQL> select \* from emp where hiredate = sysdate;

no rows selected

Solution:

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

Note!:- In oracle using round, trunc functions we can also return first date of the year, first date of the month.

Syntax!:-

round(date, 'year')

Syntax!:-

round(date, 'month')

Syntax!:-

trunc(date, 'year')

Syntax!:-

trunc(date, 'month')

e.x! - august → 10th → 2015!

SQL> select round(sysdate, 'year') from dual;  
output → 1 - JAN - 16

execution! - Server checks given date is in (Jan-Jun) or in (Jul-Dec)  
if Jul-Dec then it will returns next years first date.

SQL> select round(sysdate, 'month') from dual;  
output → 1 - AUG - 15

execution! - 10th AUGUST is in (1-15) then it returns this month first date.

SQL> select round(sysdate, 'day') from dual;  
output → 9th - August - 15

execution! - 10th August is Monday is in (mon-wed), then it returns this week  
first date (Sunday)

e.x! -

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

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

SQL> select trunc(sysdate, 'day') from dual;  
output → 09 - AUG - 15

### Group functions (or) aggregate functions:-

Oracle having following group functions there are.

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

In all databases group functions are operate over no. of values in a column & returns a single value.

1) max()! - It returns maximum value from a column

e.x! - SQL> select max(sal) from emp;  
output → 5400

SQL> select max(hiredate) from emp;  
output → 23 - MAY - 87

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

SQL> Select max(ename) from emp;

2) min():-

SQL> Select min(sal) from emp;

output → 1400

SQL> Select min(hiredate) from emp;

output → 17-DEC-80

Note!:- In all databases we are not allowed to use group function in where clause.

e.x!:- SQL> Select \* from emp where sal = min(sal);

error: group function is not allowed here

3) avg():- It returns average from no. datatype column.

e.x!:- SQL> Select avg(sal) from emp;

output → 2266.07143

SQL> Select avg(comm) from emp;

output → 550

In oracle by default all group functions ignores null values except count(\*)

If you want to count null values then we must use nvl function.

e.x!:- SQL> Select avg(nvl(comm,0)) from dual;

output → 157.142857

4) Count(\*)!:- It counts number of rows in a table including null value.

e.x!:- Select count(\*) from emp;

output → 14

5) count(columnname)!:- It counts no. of not null values in a column.

e.x!:- SQL> Select count(comm) from emp;

output → 4 (values is there in comm column)

SQL> Select count(deptno) from emp;

output → 14

SQL> Select count(distinct(deptno)) from emp;

group by!:- This clause is used to divide similar data items into set of logical groups. 11/08/2015

Syntax!:-

Select columnname from tablename group by columnname;

Q) write a query to display no. of employee in each department from emp table by using groupby?

Ans:- SQL> select deptno, count(\*) from emp group by deptno;

Output → 

DEPTNO.	COUNT(*)
10	3
20	5
30	6

Q) write a query to display no. of employees in each job from emp table by using groupby?

Ans:- SQL> select job, count(\*) from emp group by job;

Output → 

JOB	COUNT(*)
CLERK	4
SALESMAN	4
PRESIDENT	1
MANAGER	3
ANALYST	2

e.x:- SQL> Select deptno, min(sal), max(sal) from emp group by deptno;

DEPTNO	MIN(SAL)	MAX(SAL)
10	2050	5400
20	1400	3200
30	1450	2250

Note:- In all databases we can also use group by clauses without using group functions.

e.x:- SQL> select deptno from emp group by deptno;

Output → 

deptno
10
20
30

\*Note:- Other than group function column specified after select those all columns must be used after groupby otherwise oracle server returns an error: not a group by expression.

e.x:- SQL> select deptno, job, sum(sal) from emp group by deptno;  
error! not a group by expression

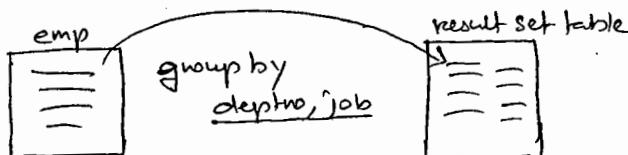
Solution

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

Note:- whenever we are submitting group by class then database servers arrange data in some proof based on specified columns after group by class. This result will be stored internally in result set table from this table only we are selecting displayed columns after select list. that's why after group by class

we are specifying no. of columns also. These all columns not required to display after select.

e.x:- SQL> select deptno from emp group by deptno, job;



Note:- In all database when we are trying to display group function with another column then database servers return errors. To overcome this problem we must use group by class.

e.x:- Step 1

SQL> select sum(sal) from emp;

31725

Step - 2

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

error: not a single-group group function

Solution:-

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

Output →

DEPTNO	SUM(SAL)
10	9550
20	11875
30	10300

Note:- Generally whenever we are submitting group function then database server executes all values at a time for the table column whenever we are using this group function within group by class then these group function are execute for each & every sub group within group by.

Q) Write a query to display those department having more than 3 employees from emp table by using group by ?

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

Solution:-

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

Output →

DEPTNO	COUNT(X)
30	6
40	5

Having clause! - After group by clause we are not allowed use where clause in place of this one ansi/iso sql provided another clause having generally we want if you want to restrict rows in a table then we are using where clause whereas if you want to restrict groups after group by then we must use having clause.

12/08/2015

Generally in where clause we are not allowed to use group function but whereas in having clause we can also use group functions.

e.x! - SQL > Select deptno, sum(sal) from emp group by deptno having sum(sal)>10000

DEPTNO	SUM(SAL)
30	10300
20	11875

Q) Write a query to display year, no. of employees for year in which more than one employee was hired from the year from "year" table by using group by?

= SQL > Select to\_char(hiredate, 'YYYY'), count(\*) from emp group by

TO_C	COUNT(*)
1987	2
1980	1
1982	1
1981	10

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

Year	COUNT(*)
1987	2
1981	10

Note! - In all databases we can also use having clauses in invisible functions because in all databases whenever we are using group by clause based on group functions all group functions are internally available.

e.x! - SQL > Select deptno, sum(sal) from emp group by deptno having count(\*)>3;

DEPTNO	SUM(SAL)
30	10300
20	11875

Order by! - This clause is used to arrange data in either ascending order or in descending order along with order by clause we are using keywords there are asc, desc by default order by clause having ascending order

Syntax! -

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

e.x!- SQL> select \* from emp order by sal desc;

Note!- we can also use multiple columns in order by clause but in this case database servers sorting data based on first column specified with in order by clause.

e.x!- SQL> select deptno, sal from emp order by deptno, sal desc;

output →	DEPT NO	SAL
	10	5400
	10	2100
	10	2000
	20	3200
	20	

### Syntax!-

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

e.x!- SQL> select deptno, count(\*) from emp where sal > 1000 group by deptno  
having count(\*) > 3 order by deptno desc;

output →	DEPTNO	COUNT (*)
	30	6
	20	5

### rollup, cube!-

Oracle 8i introduced rollup, cube clause these clause are used along with group by clause only. If you want to display subtotal, general total automatically. Then we must use rollup, cube clause along with group by.

→ rollup is used to calculate sub total values automatically based on a single column if you want to calculate subtotal, general total based on no. of columns then we must use cube.

Syntax!- SQL> select col1, col2.. from tablename group by rollup (col1, col2....);

SQL> select col1, col2 from tablename group by cube (col1, col2....);

e.x!- SQL> select deptno, job, sum(sal) from emp group by rollup (deptno, job);

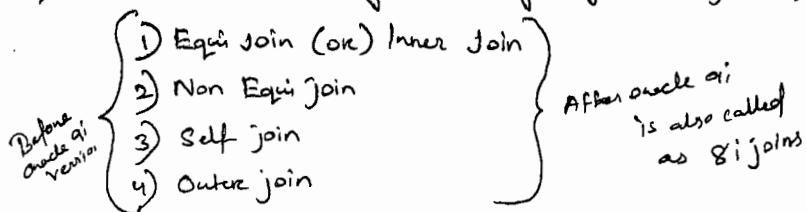
SQL> select deptno, job, sum(sal), ~~from~~ count(\*) from emp group by  
cube (deptno, job);

13/08/2015

e.x! - SQL> Select ename, sum(sal) from emp group by rollup(ename);

## JOINS

- Joins are used to retrieve data from multiple tables
- In all databases when we are joining 'n' tables then we must use ' $n-1$ ' joinings conditions.
- Oracle server having following types of joins



→ These 4 joins are also called as 8ijoin

## 8ijoin (or)ansi joins

- 1) Inner join
- 2) Left outer join
- 3) Right outer join
- 4) Full outer join
- 5) natural join

→ In oracle we can also retrieve data from multiple tables without using joins in this case oracle server internally uses default join when we are specifying no. of tables within from clause in oracle default join is cross join but cross join is internally implemented based on cartesian product. that's why this join returns more duplicated data.

e.x! - SQL> Select ename, sal, dname, loc from emp, dept;

### 1) Equi Join :-

- Based on equality operator we are retrieving data from multiple tables here joining conditional columns must belongs to same data type.
- whenever tables having common columns then only we are using equi join & also these common columns must belongs to same data type.

### Syntax:-

Select col1, col2 ..... from tablename1, tablename2 .

where tablename1. Common columnname = tablename2. Common columnname,

→ Join Condition.

e.x! - Select ename, sal, deptno, dname, loc  
from emp, dept;  
where emp.deptno = dept.deptno;

} throws error: column ambiguously defined.

throws

Solution :-

SQL > select ename, sal, dept.deptno, dname, loc  
 from emp, dept  
 where emp.deptno = dept.deptno;

Note :- In all databases for avoiding feature ambiguity then we are specifying tablename along with every column using a (.) dot operator with select list

Using aliasname :-

- we can also create alias names for the tables in from clause of the joins. these alias names are used in either in select list or in joining condition instead of table name.
- These alias names are also called as reference name.
- These alias names must be different names

Syntax :- from tablename1 aliasname1, tablename2 aliasname2;

e.g! - SQL > select ename, sal, d, deptno, dname, loc  
 from emp e, dept d  
 where e.deptno = d.deptno;

emp changes to e  
 dept changes to d

Note :- In all databases by default equijoins returns matching rows only

[here deptno 40 does not display when we are using d.deptno also]

- Q) write a query to display the employees who are working on the location chicago  
 from emp, dept tables by using equijoins ?

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

Output	
ENAME	LOC
WARD	CHICAGO
TURNER	CHICAGO
ALLEN	CHICAGO
JAMES	CHICAGO

Note :- In oracle if you want to filter the data after joining condition then we must use logical operator ~~AND~~ AND with in big joins where as in joins we can also use either logical operator AND or where conditions.

- Q) write a query to display dname, sum(sal) from emp, dept tables by using equijoins?

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

Output	
DNAME	SUM(SAL)
ACCOUNTING	25500
RESEARCH	11875
SALES	10300

Ex:- SQL) Select loc, min(sal), max(sal), avg(sal), sum(sal)  
 from emp e, dept d  
 where e.deptno = d.deptno group by loc;

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

<u>Output</u>		
DEPTNO	DNAME	SUM(SAL)
10	ACCOUNTING	9550
20	RESEARCH	11875
30	SALES	10300

Ex:- SQL) Select dname, sum(sal)  
 from emp e, dept d  
 where e.deptno = d.deptno  
 group by dname  
 having sum(sal) > 10000

<u>Output</u>	
DNAME	SUM(SAL)
RESEARCH	11875
SALES	10300

Ex:- SQL) Select dname, sum(sal)  
 from emp e, dept d  
 where e.deptno = d.deptno  
 group by rollup(dname)

<u>Output</u>	
DNAME	SUM(SAL)
ACCOUNTING	9550
RESEARCH	11875
SALES	10300
	31725

### Outer Join :-

→ These join is used to retrieve all rows from one table & matching rows from another table.

→ Generally using equijoin we are retrieving matching rows only. If you want to retrieve non matching rows also then we are using join operator (+) within joining condition of the equijoin these join is also called as oracle 8i outer join.

Note:- These join operators can be used only one side at a time within joining condition

Condition

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

↙  
matching rows      ↘  
all rows

Output  
non matching rows  
40 OPERATIONS BOSTON

### Non Equi Join :-

Based on other than equality condition (<>, <, <=, >, >=, between, ....) we are retrieving data from multiple tables

Ex:- SQL) Create table t1 (deptno number(10), ename varchar2(10));

SQL> Insert into t1 values (....);

SQL> Select \* from t1;

DEPTNO	ENAME
10	a
20	b

SQL> Create table t2 (deptno number (10), dname varchar (10));

SQL> Insert into t2 values (....);

SQL> Select \* from t2;

DEPTNO	DNAME
10	x
20	y
30	z

SQL> Select \* from t1, t2 where t1.deptno > t2.deptno;

DEPTNO	ENAME	DEPTNO	DNAME
10	a	10	x
20	b	20	y

SQL> Select \* from t1, t2 where t1.deptno > t2.deptno;

DEPTNO	ENAME	DEPTNO	DNAME
20	b	10	x

17/08/2015

Note:- In oracle when table doesn't have common column then also we are using non equi join but in this case <sup>one</sup> table column value lies between another table two columns.

For e.g:- SQL> select \* from emp;

SQL> select \* from salgrade;

SQL> select ename, sal, lsal, hisal from emp, salgrade  
where sal between lsal and hisal;

or

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

Self Join!-

Joining a table to itself is called self join here also joining conditional column must belongs to same datatype.

Generally if you want to compare 2 different column value with a single table then we are using self join but those two columns must belongs to

Same data types.

In self join we must create alias names for the table within from clause. These alias names are also called as reference names. These alias names must be different names. These alias names behaves like a ~~new~~ exact table during query execution time.

Syntax:- from tablename aliasname1, tablename aliasname2;

Q) write a query to display employee names & their manager names from emp table by using self join?

↳ SQL> Select e1.ename "employee", e2.ename "managers"  
from emp e1, emp e2 where e1.mgr = e2.empno;

Q) write a query to display the employees who are getting more salary than their managers from emp table by using self join?

↳ SQL> Select e1.ename "employee", ~~e2.ename~~ e1.sal, e2.sal,  
e2.ename "managers"  
from emp e1, emp e2 where e1.mgr = e2.empno  
and e1.sal > e2.sal;

employees	SAL	SAL	managers
FORD	3200	2375	JONES
SCOTT	3200	2375	JONES
MILLER	2100	2050	CLARK

Q) write a query 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.hiredate, e2.ename  
"managers" from emp e1, emp e2 where e1.mgr = e2.empno  
and e1.hiredate < e2.hiredate

9i joins (or) ANSI joins :-

- 1) Inner join
- 2) Left outer join
- 3) Right outer join
- 4) Full outer join
- 5) natural join

**SRI RAGHAVENDRA XEROX**  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

1) Inner Join! → These joins also return matching rows only here also joining conditional column must belong to same datatype.

→ Inner join performance is very high compare to oracle 8i equijoin where tables having common column then only we are using this join.

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

18/08/2015

Q) write a query to display the employees who are working on the location chicago from emp, dept tables by using 9i inner join?

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

Using clause! - In 9i joins we can also use using clauses in place of 'ON' clause Using clause performances is very high compare to 'ON' clause & also using clause return common column one time only.

Syntax! -

SQL> Select \*  
from tablename1 join tablename2  
using (common columnname1, common column2: ...);

e.x:- SQL> Select \* from t1;

A	B	C
X	Y	Z
P	Q	R

SQL> Select \* from t2;

A	B
X	Y
S	T

SQL> Select \*

from t1 join t2

on t1.a = t2.a and t1.b = t2.b;

A	B	C	A	B
X	Y	Z	X	Y

SQL> Select \*

from t1 join t2  
using (a, b);

A	B	C
X	Y	Z

Note:- whenever we are using Using clause then we are not allowed to use aliasname on joining conditional column within select list.

e.x:- SQL> Select ename, sal, d.deptno, dname, loc  
from emp e join dept d using (deptno);

error: column part of Using clause cannot have qualifier.

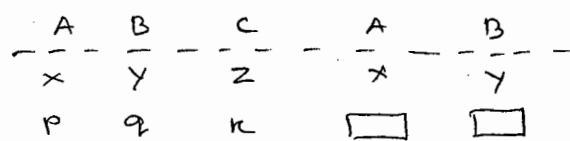
Solution:-

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

## 2) Left Outer Join:-

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

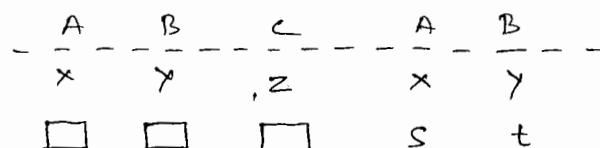
e.x:- SQL> Select \* from t1 left outer join t2  
on t1.a=t2.a and t1.b=t2.b;



## 3) Right Outer Join:-

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

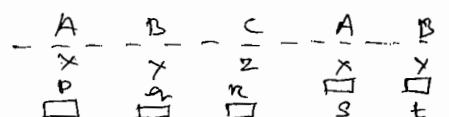
e.x:- SQL> Select \* from t1 right outer join t2  
on t1.a=t2.a and t1.b=t2.b;



## 4) Full Outer Join:-

This join returns all rows from all the table because it is the combination of Left, Right outer joins. This join also returns null value in place of non matching rows in another table.

e.x:- SQL> Select \* from t1 full outer join t2  
on t1.a=t2.a and t2.b=t2.b;



### 5) Natural Join:-

This join also returns matching rows only in this join we are not allowed to use joining condition explicitly. because internally database server only automatically establishes join condition. but here both tables must have common column name this join performance is very high compare to inner join.

#### Syntax:-

SQL> select \* from tablename1 natural join tablename2;

Note:- This join internally uses using clause that's why this join also returns common column one time only.

e.x:- SQL> select \* from t1 natural join t2;

A	B	C
X	Y	Z

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

### Cross Join:-

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

#### 8<sup>i</sup> joins

#### Syntax:-

```
Select col1, col2...
from table1, table2, table3
where table1 common col = table2
      common col
      AND
      table2 common col = table3 common col
```

#### 9<sup>i</sup> joins

#### Syntax:-

```
Select col1, col2
from table1 join table2
on table1 common col = table2 common col
      join table3
on table2 common col = table3 common col;
```

### CONSTRAINTS

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

Oracle server having following types of constraints.

- 1) Not null
- 2) Unique

- 3) Primary key
- 4) Foreign key
- 5) Check

Generally constraints are created on table columns. All above constraints are defined in two ways 19/08/2015

- 1) Column Level
- 2) Table Level

1) Column level:- In this method we are defining constraints on individual columns i.e whenever we are defining column then only specify we are specifying constraint type.

Syntax:- SQL> create table tablename (column1 datatype(size),  
constraint type, column2 datatype(size) constraint type ...);

2) Table level:- In this method we are defining constraints on group of columns i.e here first we are creating all columns then last only we are specifying constraint type along with groups of columns.

Syntax:- SQL> create table tablename (column1 datatype(size), column2 datatype(size)  
----- constraint type (col1, col2, ...));

### 1) not null:-

→ In all databases not null constraint doesn't support table level.

→ not null constraint doesn't accept null value but it will accept duplicate value.

#### Column level:-

SQL> create table z1(sno number(10) not null, name varchar2(10));

SQL> insert into z1 values(null, 'a');

ORA-1400: cannot insert NULL into sno

### 2) Unique:-

→ Unique constraint defined in column level, table level

→ Unique constraint doesn't accept duplicate values but it will accept null values.

Note:- Whenever we are creating unique constraint then oracle server internally automatically creates 'btree' index of those columns

#### Column level:-

SQL> create table z2(sno number(10) unique, name varchar2(10));

#### Table level:-

SQL> create table z3(number(10), name varchar2(10), unique(sno, name));

SQL> Select \* from Z3;

SNO	NAME
1	Merkali
1	abc

SQL> ~~set~~ insert into Z3 values (1, 'abc');  
error! Unique constraints violated

### 3) Primary key:-

→ primary key uniquely identifying a record in a table there can be only one primary key in a table.

→ primary key doesn't accept null values & also accepts doesn't accept duplicate value.

Note:- whenever we are creating primary key oracle server inherently automatically creates 'btree' indexes\* of those columns.

#### Column level:-

SQL> Create table Z4 (sno number(10) primary key, name varchar2(10));

#### 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. is the combination of column as a single primary key.

### 4) Foreign key:-

→ If you want to establishes relationship between tables then we are using referential integrity constraints foreign key. One table foreign key must belong another table primary key and also these two columns must belong to same datatype.

→ Always foreign key values are based on primary key values only.

→ Generally primary key doesn't accept duplicate, null values where as foreign key accepts duplicate, null values.

#### Column level:- (References) :-

##### Syntax:-

SQL> Create table tablename (col1 datatype (size) references mastertablename  
(primarykeycolumnname), col2 datatype (size), ...);

SQL> Create table h4 (sno number(10) references Z4);

OR

SQL> Create table g4 (x number(10) references Z4 (sno));

## Table level:- (Foreign key , references) !-

20/08/2015

### Syntax:-

```
SQL> Create table tablename (col1 datatype (size), col2 datatype (size),
..... foreign key (col1, col2, ....) references
mastertablename (primarykey colnames));
```

e.x!- SQL> Create table hs (sno number (10), name varchar2 (10), sal number (10),
foreign key (sno, name) references z5);

whenever we are establishing relationships between tables using foreign key  
then oracle server internally automatically violates following two rules  
there are

- 1) Deletion in master table
- 2) Insertion in child table

### 1) Deletion in master table:-

when we are trying to delete a master table record within master table & also  
that record exist in child table then oracle server returns an error ora - 2292  
to overcome this problem if you want to master table record in master table then  
first we must delete child table records within child table then only we are  
allowed to delete those records in master table otherwise use ~~on delete cascade~~ clause.

### On delete cascade :-

On delete cascade clause can be used along with foreign key only whenever we  
are using on delete cascade clause in child table then we are trying to delete  
master table record in master table then automatically corresponding master & child  
table records are automatically deleted in both the table.

### Syntax:-

```
SQL> Create table tablename (col1 datatype (size) references mastertablename
(primaryKey colname) on delete cascade, .....);
```

e.x!- SQL> Create table mes (sno number (10) primary key);

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

SQL> Select \* from mes;

SNO -

1

2

3

4

SQL> Create table child (sno number (10) references mes on delete cascade);

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

SQL> Select \* from child;

SNO -  
|-----|  
| 1 |  
| 2 |  
| 3 |  
| 4 |

**SRI RAGHAVENDRA XEROX**  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

## Testing :- (Deletion In Master Table) :-

SQL> delete from mas where sno = 1;  
1 row deleted

SQL> select \* from mas;

Sno
2
3
4

SQL> select \* from child;

Sno
2
2
3
4

## On Delete Set null;

- Oracle also supports another clause on delete set null along with foreign key
- whenever we are using these clause when we are trying to delete primary key value in master table then automatically that value is deleted from master table & also those values are set to null in foreign key column in child table

### Syntax:-

SQL> create table tablename (col1 datatype (size) references master tablename  
(primary key col~~name~~ name) on delete set null;

## 2) Insertion in child table!:-

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

e.g:- SQL> insert into child values (5);

error: ORA- 2291 : Integrity constraint violated - parent key not found.

### Solution:-

SQL> insert into mas values (5);  
1 row created.

SQL> select \* from mas;

SQL> insert into child values (5);

SQL> select \* from child;

## Check!:-

- It is used to define logical conditions according to our business rules.
- In oracle we are not allowed to define check constraint on sysdate

### Column Level!:-

Syntax:- SQL> create table tablename (col1 datatype (size) check (logical condition),  
col2 datatype (size), ...);

```

SQL> create table test (name varchar2(10) check (name = upper(name)));
SQL> insert into test values ('abc');
error: check constraint
(SCOTT.SYS-005529) violated.
SQL> insert into test values ('ABC');
SQL> Select * from test;

```

21/08/2015

e.x:- SQL> create table test1 (sal number(10) check (sal > 5000));
SQL> insert into test1 values (2000);
error: check constraint violated
SQL> insert into test1 values (9000);
SQL> select \* from test1;

### Table level:-

e.x:- SQL> create table test2 (name varchar2(10), sal number(10), check name=upper(name) and sal > 5000);

### Assigns user defined names to constraints

- In all database whenever we are creating constraints database servers internally automatically creates unique identification number for identifying constraints uniquely.
- Oracle also generates an unique identification no. in the format of `SYS_Cn` for identifying constraints uniquely. This is called predefined constraint name.
- In place of this one we can also create our own name by using constraint keyword. This is called user defined constraint name.

Syntax:-

```

constraint userdefinedname constrainttype
    ↗ Constraint name
    ↗ May be notnull, unique, primarykey, foreignkey, check

```

e.x:- SQL> create table test3 (sno number(10) primary key);

### Testing:-

```

SQL> insert into test3 values (1);
1 row created
SQL> insert into test3 values (1);
error: unique constraint
(SCOTT.SYS-005544) violated

```

### User defined constraint name

SQL> create table test4 (sno number(10) constraint P\_ab primary key);

### Testing:-

```

SQL> insert into test4 values (1);
1 row created.

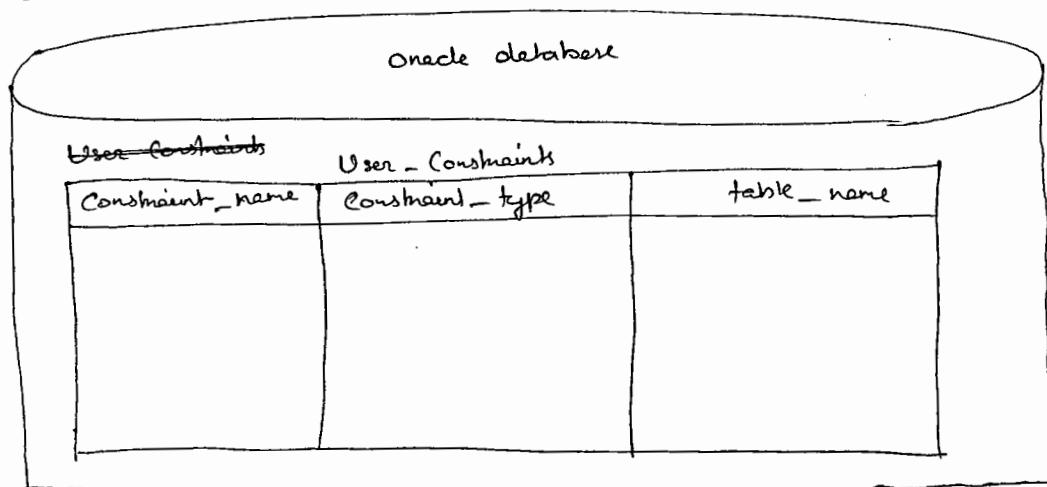
```

**SRI RAGHAVENDRA XEROX**  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

`SQL> insert into testy values (1);`  
error: unique constraint  
`(SCOTT.P_AB) violated`

## Data Dictionary :-

In oracle all objects information stored in read only table automatically. These read only table are also called as data dictionaries that is whenever we are installing oracle server then automatically so many read only tables are created. These read only table stores particular ~~read only table store~~ object related information. These are also called data dictionaries.



Note! - In oracle all constraints information stored under user\_constraint data dictionary.

e.g.:  $\text{sg}_1$  else were-constraints;

SQl> select constraint\_name, constraint\_type from user\_constraints where table\_name='EMP';

<u>CONSTRAINT_NAME</u>	<u>CONSTRAINT_TYPE</u>
PK_EMP	P
FK_DEPTNO	R

```
SQL> Create table emp (empno number (4) constraint pk_emp primary key,...  
deptno number(2) constraint fk_deptno references dept (deptno));
```

Note:- In oracle if you want to view column names along with constraints name then we are using user\_cons\_columns data dictionary.

e.g.: `SHOW desc user_cons_columns;`

`SQL> select constraint_name, column_name from user_cons_columns  
 where table_name = 'EMP';`

<u>CONSTRAINT_NAME</u>	<u>COLUMN_NAME</u>
PK_EMP	EMPNO
FK_DEPTNO	DEPTNO

Note:- In oracle if you want to view logical condition of the check constraints then we are using SEARCH\_CONDITION property from USER\_CONSTRAINT data dictionary.

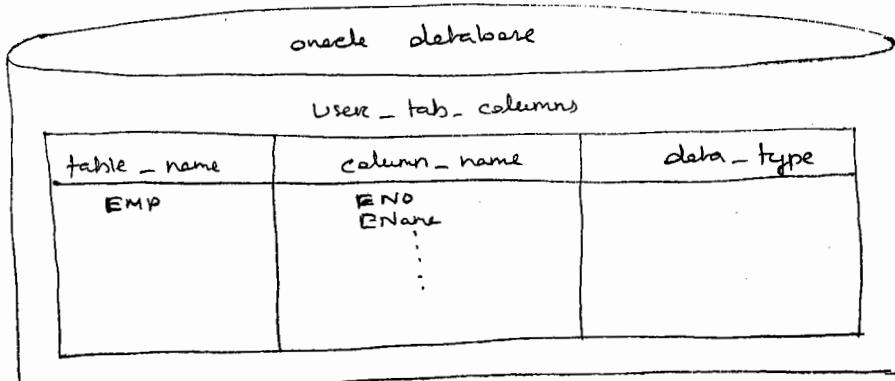
e.x:- SQL> desc user\_constraints;

SQL> Select search\_condition from user\_constraints where table\_name = 'TEST';

SEARCH\_CONDITION  
name = upper(name);

22/08/2015

→ In Oracle all columns information stored in "user\_tab\_columns" data dictionary.



e.x:- SQL> desc user\_tab\_columns;

SQL> select column\_name from user\_tab\_columns where table\_name = 'EMP';

\*Q) Create a query which counts no. of columns in emp table?

SQL> select count(\*) from user\_tab\_columns where table\_name = 'EMP';

Count(\*)  
-----  
8

default clause:-

In Oracle we can also provide default values for a column using default clause

Syntax:-

columnname datatype(size) default value

e.x:- SQL> create table test (name varchar2(10), sal number(10) default 3000);

SQL> insert into test(name) values('xyz')

SQL> select \* from test;

NAME SAL  
--- ---  
XYZ 3000

→ 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.

e.x:- SQL> desc user\_tab\_columns;

SQL> select column\_name, data\_default from user\_tab\_columns where table\_name = 'TEST';

Column-name	DATA-DEFAULT
Sal	3000

Adding (or) dropping constraints on existing table:-

→ In oracle using alter command we can also add or drop constraints on existing table.

→ In oracle if you want to add constraints on existing table existing columns then we are using table level syntax method.

Primary key	sno		Name	Unique

SQL> Alter table b1 add primary key (sno);

SQL> Alter

Note:- If you want to add a new column along with constraint then we are using column level syntax method.

SQL> Alter table b1 add name varchar2(10) unique;

Adding a foreign key

e.x:- SQL> Create table b2 (sno number (10));

SQL> Alter table b2 add foreign key (sno) references b1(sno);

Note:- If you want to add not-null constraints on existing table existing column then we are using alter with modify.

Syntax:-

alter table tablename modify columnname not null;

e.x:- SQL> Create table b3 (sno number (10));

SQL> Alter table b3 modify sno not null;

SQL> desc b3;

Note:- In all database whenever we are creating a table from existing table by using select statement then except not null constraint all constraint are never copied.

e.x:-

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);
```

### disapplying constraints :-

24/08/2015

#### methods:-

```
Alter table tablename drop constraint constraint_name
```

#### methods:-

Syntax ① Alter table tablename drop primary key;

Syntax ② Alter table tablename drop unique (col1, col2, ...)

e.x!:-

```
sql> create table test (sno number(10) primary key;
      sql> Alter table test drop primary key;
```

Note!:- If you want to drop primary key along with referenced foreign key then use cascading cascade clause along with alter drop

Syntax!:-

```
alter table tablename drops primary key cascade;
```

e.x!:- sql> alter table b1 drop primary key;

error: this unique/primary key is referenced by some foreign keys.

```
sql> alter table b1 drop primary key cascade;
```

e.x!:- sql> desc user\_cons\_columns;

```
sql> select column_name, constraint_name from user_cons_columns where table_name='B1';
```

COLUMN_NAME	CONSTRAINT_NAME
SNO	SYS_C005575
SNAME	SYS_C005576

```
sql> alter table b3 column drop constraint SYS_C005575;
```

```
sql> desc b3;
```

### SUBQUERYS

→ 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 databases having two types of queries

- 1) Non correlated subqueries
- 2) Correlated subqueries

→ In non correlated subqueries child query executed first then only parent query is executed.

→ Where as in correlated subquery parent query executed first then only child query is executed.

## 1) Non-correlated Subquery :-

→ Non correlated subquery having two parts

a) child query (or) inner query

b) parent query (or) outer query

### a) child query :-

→ A query which provides values to the another query is called child query

→ In oracle maximum limit of child queries are upto 255

### b) parent query :-

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

## Non correlated Subqueries

1) Single row subqueries

2) multiple row subqueries

3) multiple columns subqueries

4) inline views or subqueries are used in from clause

Q) write a query to display the employees who are getting more salary than the average salary from emp table?

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

→ This is a single row subquery because here child query returns single value.

→ In single row subqueries we are using =, <, <=, >, >= operators

### Execution:-

#### Step 1:-

SQL> Select avg(sal) from emp;

2308.92857

#### Step 2:-

SQL> Select \* from emp where sal > 2308.92857;

Q) write a query to display the employees who are working in Sales department from emp, dept tables?

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

Q) write a query to display senior most employee from emp table.

SQL> Select \* from emp where hiredate = (Select min(hiredate) from emp);

25/01/2015

Note:- Generally in all databases whenever we are using subquery data with servers only returns parent query table columns as output to overcome this problem if you want to display child query tables columns as output then we are using joins with in 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');

ENAME	DNAME
ALLEN	SALES
WARD	SALES
MARTIN	SALES
BLAKE	SALES
TURNER	SALES
JAMES	SALES

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

(Q) write a query to display the employees who are working at same as SMITH department no. from emp table?

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

(Q) write a query to display second highest salary from emp table?

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

(Q) write a query to display second highest salary details from emp & dept table?

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

(Q) write a query to display department name 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));

DNAME  
ACCOUNTING

(Q) write a query to display the employees who are working under BLAKE from emp table by using empno, mgr columns?

SQL> select \* from emp where mgr = (select empno from emp where ename = 'BLAKE');

(Q) write a query 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

Note:- In all database system whenever child query contains nested group function then we must use group by in child query.

Solution:-

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

JOB	Avg(SAL)
SALESMAN	1525

Note:- whenever child query contains nested groups function then only we are using group by clause . we can also use where clause in child query when child query contains single group function.

Q) write a query to display whose job avg sal is more than the CLERK job avg salary from emp table by using group by ?

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

JOB	Avg(SAL)
PRESIDENT	5600
MANAGER	2291.66667
ANALYST	3200

**SRI RAGHAVENDRA XEROX**  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

26/08/2015

SQL> select deptno, min(sal) from emp group by deptno having min(sal) > (select min(sal) from emp where deptno = 20)

Q) write a query to display the employee details from emp table who are getting maximum salary in each department

SQL> select \* from emp where sal = (select max(sal) from emp group by deptno);  
error: single-row subquery returns more than one row.

→ This is a multiple row subquery because here child query returns multiple values  
In multiple row subqueries we are using in, all, any operators

Note:- we can also use in operator in single row subquery

Solution:-

SQL> select \* from emp where sal in (select max(sal) from emp group by deptno);

OR

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

DEPTNO	SAL	ENAME
30	2250	BLAKE
20	3200	FORD
20	3200	SCOTT
10	5600	KING

Q) write a query to display the employees who are working in either sales or research department from emp, dept tables?

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

### TOP-N Analysis

→ In oracle if you want to implement 'TOP-N Analysis' queries then we are using following two concepts these are

- 1) inline view
- 2) Rownum

#### 1) Inline View :-

→ Oracle 7.2 introduced inline view

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

→ Generally in all databases we are not allowed to use order by clause in child queries to overcome this problem oracle 7.2 introduced subquery in from clause of the parent query these type of queries are also called as inline view.

#### Syntax:-

Select \* from (select statement);

→ In oracle we are not allowed to use alias names in where condition to overcome this problem if you want to use alias names in condition then we must use inline views, because whenever we are using inline views internally alias names behaves like an exact table column within database.

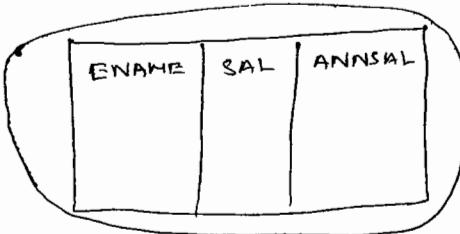
e.g:- SQL> Select ename, sal, sal \* 12 annsal from emp where annsal > 30000  
error "ANNSAL": invalid identifier

#### Solution:-

SQL> Select \* from (select ename, sal, sal \* 12 annsal from emp)  
where annsal > 30000;

ENAME	SAL	ANNSAL
SCOTT	3200	38400
KING	5600	67200
FORD	3200	38400

SQL> Select \* from



where Annsal > 30000;

## 2) Rownum:-

- Rownum is a pseudo column it behaves like a table column if you want to restrict rows in a table then we are using generalised column 'Rownum'.
- Generally whenever we are installing oracle server then automatically some columns are created in oracle database there are Rownum, Rowid these columns belongs to all tables in oracle database
- Rownum is pseudo column which automatically assigns no.s to each rows in a table at the time of selection

e.x:-

SQL> Select rownum, ename from emp;

ROWNUM	ENAME
1	SMITH
2	ALLEN
3	WARD

- Generally rounum having temporary values.

e.x:-

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

ROWNUM	ENAME
1	CLARK
2	KING
3	MILLER

- (a) write a query to display first row from emp table by using rounum?

SQL> Select \* from emp where rownum = 1;

27/08/2015

- (b) write a query to display record row from emp table by using rownum?

SQL> Select \* from emp where rownum = 2;  
no rows selected.

- Generally rounum doesn't count with more than one positive integer i.e it counts with less than, <, <= operators.

- (c) write a query to display first 5 row from emp table by using rounum

SQL> Select \* from emp where rownum <= 5;

Note:- whenever we are requesting data from the table by using rownum oracle server retrieve data from the table based on following algorithm.

SQL> Select \* from emp where rownum = 2;

no row selected.

Algorithm

rownum = 1

for i in (select \* from emp)

loop

if (rownum = 2) then

  output record;

  rownum = rownum + 1;

end if;  
end loop;

- Q) write a query to display first 5 highest salary employees from emp table?  
SQL> select \* from (select \* from emp order by sal desc) where rownum <= 5;
- Q) write a query to display 5th highest salary employee 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) write a query to display second record from emp table by using rownum?  
SQL> select \* from emp where rownum = 2 minus select \* from emp where rownum <= 1;
- Q) write a query to display rows between 1 & 5 from emp table by using rownum?  
SQL> select \* from emp where rownum between 1 and 5;
- Q) write a query to display rows between 4 to 7 from emp table by using rownum?  
SQL> select \* from emp where rownum <= 7 minus select \* from emp where rownum <= 4;
- Q) write a query to display last two rows from emp table by using rownum?  
SQL> select \* from emp where rownum minus select \* from emp where rownum <= (select count(\*) - 2 from emp);

SQL> select \* from emp where ~~rownum~~ rownum >= 1;  
rows are selected  
SQL> select \* from emp where rownum > 1;  
no rows selected

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

Note:- whenever we are using aliasname for rownum in inline views then that aliasname works with all SQL operations

- Q) write a query to display second row from emp table by using rownum alias name?  
e.g:-

SQL> select \* from (select rownum r, ename, sal from emp) where r = 2;

R	ENAME	SAL
2	ALLEN	2000

or → to display all column purpose

SQL> select \* from (select rownum r, emp.\* from emp) where r = 2

- Q) write a query to display first row, Last row from emp table by using rownum alias name?

SQL> select \* from (select rownum r, ename, sal from emp) where r = 1 or  
r = (select count(\*) from emp);

R	ENAME	SAL
1	SMITH	1300
14	MILLER	2200

Q) write a query to display even no. of records from emp table by using rownum alias name?

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

Q) write a query to display 2nd, 3rd, 4th records from emp table by using rownum alias name? 28/08/2015

SQL> select \* from (select rownum r, ename, sal from emp) where r in (2, 3, 4, 5);

Q) write a query to display rows between 4 to 7 from emp table by using rownum alias name?

SQL> select \* from (select rownum r, ename, sal from emp) where r between 4 and 7;

Q) write a query to display 5th 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;

R	NAME	SAL
5	JONES	3375

Analytical functions used in inline views:-

**SRI RAGHAVENDRA XEROX**  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

Analytical functions:-

→ Oracle 8i introduced analytical functions

→ Analytical functions are similar to group functions but group function reduces no. of rows in each group whereas analytical functions does not reduce no. of rows in each group because analytical functions internally executes for each row in a table

e.g. group function:-

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

DEPTNO	AVG(SAL)
10	3618.66667
20	2595
30	1450

Analytical functions:-

DEPTNO	AVG(SAL)
10	3618.66667
10	3618.66667
10	3618.66667
20	2595
20	2595
20	2595
...	...
...	...

→ Oracle having following analytical function there are

- 1) row\_number()
- 2) rank()
- 3) dense\_rank()

These three analytical functions automatically assigns rank either group wise or row wise in a table

Syntax:-

Analytical function () over (partition by columnname order by columnname [asc/desc])

row\_number()

→ row\_number() analytical function automatically assigns different rank numbers when values are same where as rank(), dense\_rank() functions automatically assign same rank no. when values are same & also rank skips next consecutive rank no. where as dense\_rank does not skip next consecutive rank numbers

e.x:-

row\_number()

SQL> select deptno, ename, sal, row\_number() over(partition by deptno order by sal desc) r  
from emp;

deptno	ename	sal	r
20	SCOTT	3400	1
20	FORD	3400	2
20	JAMES	3375	3

rank()

deptno	ename	sal	r
20	SCOTT	3400	1
20	FORD	3400	1
20	JAMES	3375	3

**SRI RAGHAVENDRA XEROX**  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

dense\_rank()

deptno	ename	sal	r
20	SCOTT	3400	1
20	FORD	3400	1
20	JAMES	3375	2

Q) write a query to display the employees highest salary to lowest salary in each department from emp table & also automatically assigns ranks by using analytical functions?

e.x:- SQL> select \* from (select deptno, ename, sal, ~~row\_number()~~ <sup>now\_number</sup> over(partition by deptno order by sal desc) r from emp) where r <= 10;

Q) write a query to display 2nd highest salary employee in each department from emp table by using analytical function?

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

DEPTNO	ENAME	SAL	R
10	CLARK	3350	2
20	JONES	3375	2
30	ALLEN	2000	2

Q) write a query to display 5th highest salary employee 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=5;

DEPTNO	ENAME	SAL	R
30	BLAKE	3250	5

Note:- In analytical functions partition by clause is an optional clause

Q) write a query to display nth highest salary employee from emp table by using rownum alias name?

SQL> 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
10	KING

31/08/2015

rowid:-

rowid is a pseudo column, it behaves like a table column.

→ In oracle if you want to retrieve data very fastly then we must use rowid

→ In oracle whenever we are inserting data into table then automatically oracle server generates an unique identification number in hexa decimal format for identifying a record uniquely this address is stored in within block this address is also called as rowid.

→ Generally rowid having temporary values whereas rowid having fixed values.

Ex:-

SQL> select rownum, rowid, ename from emp;

SQL> select rownum, rowid, ename from emp where deptno=10;

→ In oracle by default rowid having ascending order we can also use min, max functions in rowids

Ex:- SQL> select min(rowid) from emp;

SQL> select max(rowid) from emp;

Q) write a query to display first record from emp table by using rowid ?

SQL> select \* from emp where rowid = (select min(rowid) from emp);

Q) write a query to display last record from emp table by using rowid ?

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

Note:- In oracle we can also use rowid in analytical functions

Q) write a query to display second record from emp table by using row-number analytical function, rowid ?

SQL> select \* from (select deptno, ename, sal, row\_number() over (order by rowid))

rc from emp) where rc = 2;

Q) write a query to display second record from each department from emp table by using analytical function, rowid ?

SQL> select \* from (select deptno, ename, sal, row\_number() over (partition by deptno order by rowid))

rc from emp) where rc = 2;

SRI RAGHAVENDRA XEROX

Software Languages Material Available

Beside Bangalore Ayyagar Bakery,

Opp. CDAC, Balkampet Road,

Ameerpet, Hyderabad.

DEPTNO	ENAME	SAL	R
10	KING	5000	2
20	JONES	2975	2
30	WARD	1250	2

Q) write a query to display last two records from emp table by using analytical function, rowid ?

SQL> select \* from (select deptno, ename, sal, row\_number() over (order by rowid desc))

rc from emp) where rc <= 2;

DEPTNO	ENAME	SAL	R
10	MILLER	1300	1
20	FORD	3000	2

Deleting Duplicate rows:-

→ In oracle if you want to delete duplicate rows in a table then we must use rowid

Q) write a query to display duplicate rows from the following table ?

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

SQL> insert into test values (8 sno);

SQL> select \* from test;

SNO
10
90
10
20
20
30
40

SQL> select sno, count (\*) from test group by sno having count (\*) > 1;

SNO	count (*)
20	2
30	3

Q) write a query to delete duplicate rows except one row in each group from the following table are

write a query to delete duplicate rows in a table

SQL> delete from test where rowid not in (select min (rowid) from test group by sno);

SQL> select \* from test;

SNO
10
20
30
40

#### Mutliple Columns SubQuery !-

→ In all database we can also compare multiple <sup>column</sup> values of the child query with the multiple column values of the parent query these type of query are also called as multiple columns subquery.

→ In multiple column subquery we must specified parent query where condition column with in parenthesis. () .

#### Syntax:-

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

Q) write a query to display the employees whose job, MGR match with the JOB, MGR of the employee SCOTT from emp table by using multiple column subquery ?

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

01/09/2015

Q) write a query to display the employees who are getting maximum salary in each department by using multiple rowsub query?

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

DEPTNO	SAL	ENAME
30	2850	BLAKE
20	3000	SCOTT
10	5000	KING
20	2850	FORD

Q) write a query to display the employees who are getting maximum salary in each department by ~~row~~ column from emp table by using multiple column subquery?

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

DEPTNO	SAL	ENAME
30	2850	BLAKE
20	3000	SCOTT
10	5000	KING

Q) write a query to display senior most employee in each job from emp table by using multiple column subquery?

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

JOB	HIREDATE	ENAME
CLERK	17-DEC-80	SMITH
SALESMAN	20-FEB-81	ALLEN
MANAGER	02-APR-81	JONES
PRESIDENT	17-NOV-81	KING
ANALYST	03-DEC-81	FORD

Q) write a query to display ename, dname, salary of the employees whose salary, commission match with the salary, commission of the employees working in the location 'DALLAS'?

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

### Correlated subqueries:-

→ Generally in non correlated subqueries child query is executed first then only parent query executed where as in correlated subqueries parent query is executed first then only child query is executed.

- Generally in non correlated subqueries child query is executed only once per parent query table whereas in correlated subqueries child query is executed for each row of parent query table
- whenever we are submitting co-related subquery then database servers get a candidate row from the parent query table and then control passes into child query where condition and then based evaluation value it compares value with where condition of the parent query.
- In correlated subqueries we must create an alias name for the parent query table & pass this alias name into child query where condition

Syntax:-

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

- Generally correlated subquery are used in denormalization process. In this process we use co-related updates which is used to modify one table column data based on another table column if those tables are related for comparing no. of tables into single table

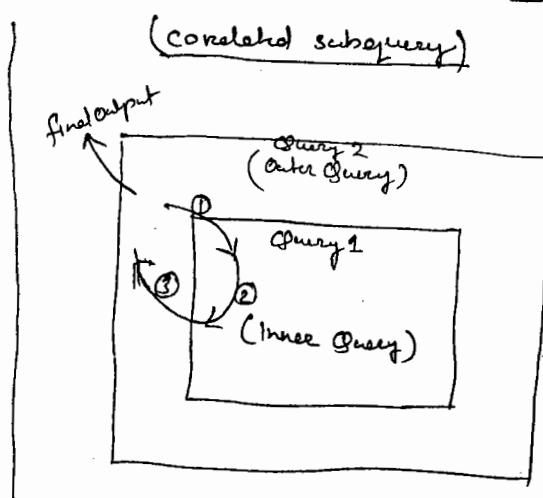
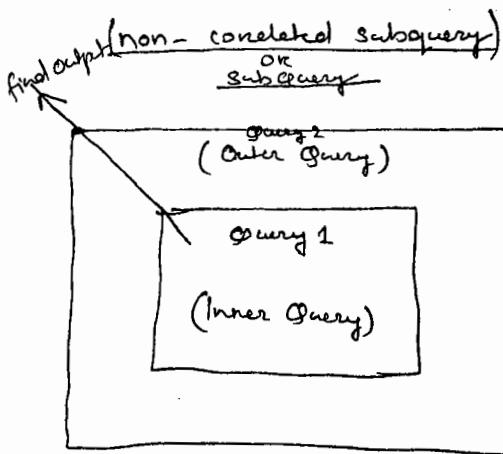
Correlated Update Syntax:-

```
Update tablename1 aliasname1 set
      columnname = (Select columnname from tablename2 aliasname2 where aliasname1.
                     commoncol = aliasname2.commoncolname);
```

e.x:-

```
sql> alter table emp add dname varchar2(10);
sql> update emp e set dname=(select dname from dept d where e.deptno=d.deptno),
      select * from emp;
```

02/09/2015



Q) Create a query to display first highest salary employee from emp table by using co-related subquery?

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

KING 5000

Q) Create a query to display second highest salary employee from the following table by using co-related subquery?

e. ex:-  
SQL> create table test (ename varchar2(10), sal number(10));  
SQL> insert into test values ('....');

ENAME	SAL
abc	100
xyz	150
pqr	200
zzz	300

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

ENAME	SAL
pqr	200

Execution:-

phase 1:-

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

Step 2:- select count(\*) from test e2. where e2.sal >= 100;

4

Step 3:- select \* from test e1 where 2 = 4; (false)

phase 2:-

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

Step 2:- select count(\*) from test e2. where e2.sal >= 150;

3

Step 3:- select \* from test e1 where 2 = 3; (false)

phase 3:-

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

Step 2:- select count(\*) from test e2. where e2.sal >= 200;

2

Step 3:- select \* from test e1 where 2 = 2; (True)

pqr 200

**SRI RAGHAVENDRA XEROX**  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

Note! - whenever resource table having duplicate data within competing column then above query doesn't return any value to overcome this problem we must use distinct clause within count() function

e.x:-

SQL> insert into test values ('murali', 200);

SQL> select \* from test;

SQL> select \* from test e1 where 2 = (select count(\*) from test e2 where e2.sal >= e1.sal);

no rows selected

Solution! -

SQL> select \* from test e1 where 2 = (select count(distinct(sal)) from test e2 where e2.sal >= e1.sal);

ENAME	SAL
pqr	200
murali	200

Q) write a query to display 2nd highest salary employee from the above table by using correlated subquery n-1 method,

SQL> select \* from test e1 where (2-1) = (select count(distinct(sal)) from test e2 where e2.sal >= e1.sal);

ENAME	SAL
pqr	200
Murali	200

Q) write a query to display nth highest salary employee from emp table by using co-related subquery?

SQL> select \* from emp e1 where &no = (select count(distinct(sal)) from emp e2 where e2.sal >= e1.sal);

Enter value for no: 1

KINCH 5000

Q) create a query to display the employee who are getting more salary than the average salary of their jobs from emp table by using co-related subquery,

SQL> select \* from emp e where sal > (select avg(sal) from emp where job = e.job);

### Exists Operator:-

- In oracle we can also use exists operator in co-related subqueries exist operator performance is very high compare to in operator.
- Exist operator always returns boolean value either true or false.
- Exist operator is used in where condition of the parent query.
- whenever we are using exist operator we are not allowed to use column name along with exist operator in where condition of the parent query.

### Syntax:-

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

- Generally if you want to test one table column values available or not available in another table If those tables are related then only we are using exists operator.
- Exists operator is used to test whether the given set is empty or not empty
- Exist operator <sup>non</sup>empty set returns true. where as exists operator empty set returns false.

ex1:- exists {1, 2, 3} = true

ex2:- exists {} = false

- Q) write a query to display those departments from dept table having employees in emp table by using co-related subquery exist operator?

SQL> Select \* from dept d where exists (select \* from emp where deptno = d.deptno);

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO

- Q) write a query to display the employees who are getting same salary as 'scott' salary from emp table by using co-related subquery by exists operator?

SQL> Select \* from emp e1 where exists (select \* from emp e2 where e2.ename = 'SCOTT' and e1.sal = e2.sal);

- Q) write a query to display the employees who are getting more salary than the scott salary from emp table by using co-related subquery exist operator?

SQL> Select \* from emp e1 where exists (select \* from emp e2 where e2.ename = 'SCOTT' and e1.sal > e2.sal);

**SRI RAGHAVENDRA XEROX**  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

Not exists! -

- Q) write a query to display those department doesn't have employees in emp table by using co-related subqueries?

SQL> select \* from dept d where not exists (select \* from emp where deptno = d.deptno);

DEPTNO	DNAME	LOC
40	OPERATIONS	BOSTON

- Q) write a query to display those department does not have employees in emp table by using non co-related subquery?

SQL> select \* from dept d where deptno not in (select deptno from emp);

DEPTNO	DNAME	LOC
40	OPERATIONS	BOSTON

Note! - Generally not in operator does not work with null values instead of this one we can also use co-related subqueries along with not exists operator.

e.g.-

SQL> ~~insert~~ into emp(empno, deptno) values (1, null);

SQL> select \* from dept d where deptno not in (select deptno from emp);  
no rows selected

SQL> select \* from dept d where not exists (select \* from emp where deptno = d.deptno);

DEPTNO	DNAME	LOC
40	OPERATIONS	BOSTON

Subquery special operators used in non-correlated subqueries! -(all, any)

- Q) write a query to display the employees who are getting more salary than the highest paid employee in 20th dept from emp table?

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

- Q) write a query to display the employees who are getting more salary than the lowest paid employee in 10th department from emp table?

SQL> select \* from emp where sal > (select min(sal) from emp where deptno = 10);

→ In all databases whenever resource table having more no. of records & also child query contains max or min function & also when we are comparing more no. of value by using relational operator then those type of query degrades performance of the Appln to overcome this problem for improve performance of the query then ansi/iso sql provided subquery special operators these are all, any.

These operators are used along with relational operators in parent query where condition.

e.x:- SQL> select \* from emp where sal > all (select sal from emp where deptno > 20);  
          select \* from emp

all, any operators used in multiple rows subquery :-

04/09/2015

In  $\Rightarrow$  It returns same values in the list? Child query.

> All  $\Rightarrow$  It satisfies all values in the list

> Any  $\Rightarrow$  It satisfies any value in the list

in  $\Rightarrow$  = Any.

Q) write a query to display the employees who are getting more salary than the all salaries of the clerk from emp table by using subquery special operator?

SQL> select \* from emp where sal > all (select sal from emp where job = 'CLERK');

Note:- whenever we are using all operator database server internally uses logical operator and whenever we are using any operator database server internally uses logical operator or.

e.x:- SQL> select \* from emp where deptno > all (10, 20);

      30

SQL> select \* from emp where deptno > any (10, 20);

      20

      30

Note:- In operator is also same as =any but not in operator not same as <> any

e.x:- SQL> select \* from emp where deptno not in (10, 20);

      30

SQL> select \* from emp where deptno <> any (10, 20);

      10

      20

      30

Note:- Not in is also same as <> all

e.x:- SQL> select \* from emp where deptno not in (10, 20);

      30

SQL> select \* from emp where deptno <> all (10, 20);

      30

SRI RAGHAVENDRA XERO  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

## VIEWS

- It is a database object which is used to provide authority level of security.
- Generally views are created from tables those tables are also called as base tables.
- Generally view doesn't store data that's why view also called as virtual table or windows of a table.
- Generally if you want to restrict table columns from one user into another user then only we are creating views with required column & then only that view given to the no. of users.
- Based on the base tables views are categorised into two types.

- 1) Simple views
- 2) Complex views (or) join views.

- Simple views is a view which is created from only one base table whereas complex views is view which is created from no: of base table.

### 1) Simple View :-

#### Syntax:-

create or replace view Viewname  
as  
Select \* from tablename where condition;

#### DML operations on simple view :-

- In oracle we can also perform dml operations through simple view to base table based on following restriction
- 1) If a simple view contains group(<sup>eg</sup> function, group by, rownum, distinct, set operators, joins), then we cannot perform dml operations through simple view to base tables.
- 2) we must include base table non null column into the view then only we can perform insertion operation through simple view to base table.

e.g. 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, 'marali', 30);

1 row created

sql> select \* from emp;

e.x! - SQL> create or replace view v2

a)  
select ename, sal, deptno from emp where deptno = 10;

SQL> select \* from v2;

SQL> insert into v2 (ename, sal, deptno) values ('xyz', 2000, 30);

ERROR! cannot insert NULL into EMPNO

05/09/2015

→ Generally in all databases throw the views are simplifying queries.

→ In oracle when a view contains functions or expression then we must create alias name for those functions or expressions.

e.x! - SQL> create or replace view v1

a)  
select deptno, max(sal)

from emp

group by deptno;

error! must name this expression with a column alias

Solution! -

SQL> create or replace view v1

a)  
select deptno, max(sal) a

from emp

group by deptno;

SQL> select \* from v1;

DEPTNO	A
30	2850
20	3000
10	5000

Note! - In oracle we can also specify alias name through view parameter in this case we must specify alias name for every column with in parenthesis after view name

e.x! - SQL> create or replace view v2 (deptno, maximumsal)

a)

select deptno, max(sal)

from emp

group by deptno;

SQL> select \* from v2;

DEPTNO	MAXIMUMSAL
30	2850
20	3000
10	5000

Note:- In oracle when a view contains rownum also then we must use alias name `*`

e.x!- `sql> create or replace view v3`  
as  
`select rownum sno, ename from emp;`

`sql> select * from v3;`

SNO	SNAME
1	SMITH
2	ALLEN
3	WARD
...	...

→ In all database system whenever we are creating a view then automatically view definition are permanently stored in database.

→ In oracle if you want to view these definition then we are using `user_views` data dictionary.

e.x!- `sql> desc user_views`

`sql> Select TEXT from user_views where view_name = 'V1';`

TEXT

`Select deptno, max(sal) a  
from emp  
group by deptno`

### Complex View (or) join view:-

→ Complex view is a view which is created from multiple base tables.

e.x!- `sql> create or replace view v5 as select ename, sal, dname, loc from emp,dept  
where emp.deptno = dept.deptno;`  
`sql> select * from v5;`

→ In all databases we are unable to perform dml operations through complex view to base table. In oracle when we are trying to perform dml operations then some table columns are effected & also some other table columns are not effected. If you want to view effected, unaffected column then we are using `user_updatable_columns` data dictionary.

e.x!- `sql> update v5 set ename = 'abc' where ename = 'SMITH';`

1 row updated

`sql> update v5 set DNAME = 'xyz' where dname = 'SALES';`

Error! Cannot modify a column which maps to a non key-preserved table.

SQL> desc empupdateable\_columns;

SQL> Select these column\_name, updatable from empupdateable\_columns  
where table\_name = 'VS';

COLUMN_NAME	UPDATABLE
ENAME	YES
SAL	YES
DNAME	NO
LOC	NO

Generally we cannot perform dml operation through complex views to base tables to over come this problem oracle introduced instead of triggers in pl/sql.

By default instead of triggers are row level triggers. and also instead of triggers are created on views.

### Trigger!-

→ Trigger is also same as stored procedure & also it will automatically invoked whenever dml operations performed on table or view

→ All database system having two types of triggers

- 1) Statement level triggers
- 2) Row level triggers

In statement level trigger code is executed only once per dml statements whereas in row level trigger trigger code is executed for each row after dml statements

### Syntax!-

create or replace trigger triggername  
before/after insert/update/delete on tablename

[for each row] → for row level triggers



### Difference between statement level, row level triggers

SQL> create table test (col1 date);

#### Statement level triggers

SQL> Create or replace trigger t1 after update on emp begin insert into test  
values (sysdate);  
end;

**SRI RAGHAVENDRA XEROX**  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

### Testing:-

SQL> update emp set sal = sal + 100

where deptno = 10;

3 rows updated

SQL> select \* from test;

col1

---  
05-SEP-15

SQL> drop trigger tr1;

SQL> delete from test;

### Row level trigger:-

SQL> create or replace trigger tr2

after update on emp for each row

begin

insert into test

values (sysdate);

end;

/

### Testing:-

SQL> update emp set sal = sal + 100

where deptno = 10;

3 rows updated

SQL> select \* from test;

col1

---  
05-SEP-15

05-SEP-15

05-SEP-15

07/09/2015

### row level trigger:-

In row level trigger, trigger body is executed for each row for dml statements, that's why we are using for each row clause in trigger specification and also dml transaction values are internally automatically stored.

In two rollback by qualifiers

- these are
  - 1) : old
  - 2) : new

These qualifiers are used in either in trigger specification or in trigger body.

Syntax:-

: old. columnname

Syntax:-

: new. columnname

	insert	update	delete
:new	✓	✓	✗
:old	✗	✓	✓

(a) Create a pl/sql row level trigger on emp table, whenever we're 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> create or replace trigger tr1 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;

/

Testing!-

SQL> delete from emp where sal > 2000;

SQL> select \* from backup;

Instead of triggers!-

In oracle we are not allowed to perform dml operations, through view to base tables. 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 this

Instead of triggers are created on views.

Syntax!-

Create or replace trigger triggername

instead of insert/update/delete on viewname

for each row

begin

trigger  
body {  
  --  
  end;

SQL> create or replace trigger tr1

instead of update on vs

for each row

begin

```

update dept set
dname = :new_dname where
dname = :old_dname
update dept set      = :new_loc where loc = :old_loc
end;
trigger created

```

### Testing:

```

SQL> update VS set dname = 'XYZ'
      where dname = 'SALES';
row updated.

```

```

SQL> desc were_updatable_columns;

```

```

SQL> select column_name, updatable from were_updatable_columns
      where table_name = 'VS';
      column_name    updatable

```

column_name	updatable
ENAME	YES
SAL	YES
DNAME	YES
LOC	YES

### Materialized View:-

- Oracle 8i introduced many materialized views. Materialized views also created from base tables. Materialized views are used in data warehousing application.
- Generally views does not store data, whereas materialized views stores data.
- Materialized views are used to improve performance of the joined or aggregatable queries. Materialized view stores result of the query.
- Materialized views store replication of the remote database into local node.
- Materialized views also stores data same like a table but whenever we are refreshing materialized view its synchronized data based on base table.

### Syntax:-

```

Create materialized view Viewname as
      Select statement;

```

### Difference between view, materialized view:-

<u>View</u>	<u>Materialized View</u>
1→ View does not store data	1→ Materialized view stores data
2→ View purpose is security	2→ Materialized view purpose is Improve performance
3→ When we are dropping base table then view cannot be accessible.	3→ When we are dropping base table also materialized view can be accessible.
4→ We can perform DML operation on views.	4→ We cannot perform DML operation on materialized views.

08/09/2015

→ In Oracle before we are creating materialized view then database Administrator must give create any materialized view privilege to user otherwise oracle server returns error.

Syntax:-

grant create any materialized view to username;

e.x!- 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 scott;

sql> conn scott/tiger;

sql> create materialized view mz1

as

select \* from emp;

Materialized view created.

Note!- In oracle when views returns insufficient privileges error then we are using create any view system privilege to user.

Syntax:-

grant create any view to username;

e.x!- 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;

view created.

→ In Oracle one of the materialized view based table must contain primary key otherwise oracle server returns an error.

**SRI RAGHAVENDRA XEROX**  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

e.x!- SQL> create table test (sno number (10));

SQL> create materialized view mw1

as

select \* from test;

error: table 'TEST' does not contain a primary key constraint.

e.x!- 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 mw1

as

select \* from base;

→ In this case materialized view also behaves like a view because whenever we are creating materialized view then automatically materialized view definitions are permanently stored in database. Same as a view definition.

→ In oracle if you want to view materialized view definitions then we are using user\_mvviews data dictionary.

e.x!- SQL> desc user\_mvviews;

SQL> select query from user\_mvviews where mvviews\_name = 'MW1';

#### Execution:-

→ Generally views are not used for improve performance of the query because whenever we are creating a view. View definition are automatically stored in database whenever we are requesting views by using select statement then oracle server each & everytime executes view definitions in this case each & every time base tables are effected.

→ Materialized view performance is very high compare to view performance i.e whenever we are creating materialized view then automatically materialized view definitions are stored in database and also oracle server stores query result within materialized view. whenever user request materialized view by using select statement then oracle server retrieved data directly from materialized

View in this case base tables are not effected,

→ whenever we are refreshing materialized view, then only materialized view definitions are executed in this case only base tables are effected.

SQL> Select rowid, sno, name from base;

SQL> Select rowid, sno, name from v1;

Here view rowid are same as base table rowid that's why views doesn't stores data i.e. ~~base~~ views are also called as 'Virtual tables' or view is also called 'Windows of a table' i.e. Through the view we can Access base table data.

SQL> Select rowid, sno, name, mwo1;

In this case materialized view rowids are different from base table rowid's that's why materialized view stores data.

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 mwo1;

SNO	NAME
1	a
2	b
3	c
4	d

→ When we are refreshing materialized view it synchronizes data based on base table.

→ In oracle if you want refresh materialized view then we are using refresh procedure from dbms\_mview package

Syntax:-

dbms\_mview.refresh ('materialized viewname');  
↳ predefined  
↳ package name      procedure name

e.g!- SQL> exec dbms\_mviews.refresh ('mw1');

PL/SQL

SQL> select \* from mw1;

SNO	NAME
1	A
2	B
3	C
4	D

9/9/2015

Oracle having two types of materialized views

1. Complete refresh materialized views
2. Fast refresh materialized view.

#### Complete refresh materialized view :-

- In oracle by default materialized views are complete refresh materialized views.
- In complete refresh materialized view, whenever we are refreshing materialized views internally rows are recreated, when we are not modifying data within base table also.

#### Syntax:-

create materialized view Viewname refresh complete

as

select statement

e.g!- SQL> select rowid, sno, name from mv1;

SQL>

dbms\_mview.refresh ('mw1');

SQL> select rowid, sno, name, from mw1;

Here rowids are changed.

#### Fast refresh materialized views:-

- Fast refresh materialized view are also called as incremental refresh materialized views.
- Generally complete refresh materialized views performance is very low compare to fast refresh materialized views, because in complete refresh materialized views rows are recreated, where, as in fast refresh materialized view rows are not changed even if we are refreshing materialized view no of time also.

#### Syntax:-

create materialized view Viewname refresh fast

as

select statement;

→ Before we can perform fast refresh methods, it needs a mechanism to capture changes made to its base table. This mechanism is also called as materialized view log.

→ That's why, before we are creating fast refresh method, we must update create materialized view log on base table by following Syntax.

#### Syntax:

create materialized view log on base table name.

eg:-

```
SQL> create materialized view log on base;
SQL> create materialized view mw2 refresh fast as select * from base;
SQL> select rowid, sno, ename from mw2;
SQL> update base set name = 'zz' where sno = 2;
SQL> select * from base;
```

SNO	NAME
1	A
2	zz
3	C
4	D

```
SQL> dbms_mview.refresh ('mw2');
SQL> select rowid, sno, name from mw2;
```

(Here rowids are not change, only data is effected)

#### On demand / on commit:-

In oracle we are refreshing materialized view on two ways.

1. Manually

2. Automatically

#### 1. Manually :-

We are refreshing materialized view by using refresh procedure from dbms\_mview package. This method is called on demand method.

By default method is on demand.

#### 2. Automatically :-

We can also refresh materialized view without using dbms\_mview package. This method is called on commit method.

## Materialized views:-

Syntax:-

Materialized view viewname

refresh complete / refresh fast on demand / on commit

or

Select statement

SQL> create materialized view mw3 refresh fast on commit as

select \* from base;

SQL> update base set ename = 'murali' where sno = 3;

SQL> select \* from base;

SNO	NAME
1	XY
2	ZZ
3	murali
4	D

SQL> select \* from mw3;

SNO	NAME
1	XY
2	ZZ
3	C
4	D

SQL> commit;

SQL> select \* from mw3;

SNO	NAME
1	XY
2	ZZ
3	murali
4	D

## Data Control Language:- (DCL)

→ grant

→

SQL> conn sys as sysdba ↴

SQL> conn system / tiger ↴

SQL> Enter password: sys.

### Creating user:-

#### Syntax:

SQL> Create user username identified by password;

#### Syntax:

SQL> Grant, connect , resource to username  
      or  
      DBA

#### Syntax:-

SQL> conn username / password

### Creating a user:-

SQL> conn sys as sysdba;

Enter password: sys

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 exists.

SQL> conn scott / tiger;

SQL> grant all on emp to murali;

SQL> Select \* from emp;

error: table or view does not exists

SQL> Select \* from scott.emp;

SQL> create synonym ab for scott.emp;

SQL> Select \* from ab;

10/9/2015

privilege  
→ privilege is right given to the other user whenever we are giving privileges (permission) then only those users allow to perform some operations of the database.

→ Data Security point of view all database system having two types of user privileges

- 1) system privileges
- 2) object privileges.

### System Privileges :-

- These privileges enables user to perform actions in database. These privileges are given by database administrators only. Whenever administrator giving these privileges to no. of users then only no. of users allow to perform particular actions in the database. Otherwise Oracle Server returns an error in sufficient privileges.
- Oracle having more than 80 system privileges these are create session, create table, create procedure, create any index, create any materialized view, create any view, create trigger, ....

### Syntax:-

Grant system-privileges to username1, username2, ...;

e.g!-  
SQL> conn sys as sysdba;

SQL> Enter password : sys

SQL> grant create procedure, create any index, create any materialized view to scott, merali;

Note!- In oracle any user wants to connects to the database server then database administrator must give create session system privilege.

e.g!-  
SQL> conn sys as sysdba;

Enter password : sys

SQL> create user one identified by one;

SQL> conn one/one;

error! user ONE lacks CREATE SESSION privilege;

### Solution! -

SQL> grant create session to one;

SQL> conn one/one;

### Role! -

→ roles are created by database administrator only. role is nothing but either collection of system privileges or collection of object privileges.

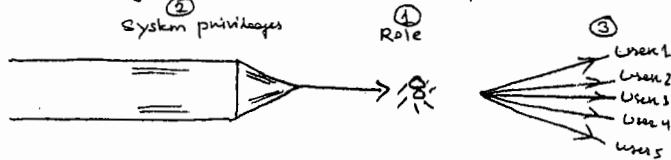
→ Oracle having two types of roles

- 1) User defined roles
- 2) Pre defined roles

### 1) User defined roles! -

→ In multiuser environment number of user works on same project & also some users requires common set of privileges. In this case only roles are

administration creates a role & assigns common set of privileges role & then only that role given to the number of users.



Step 1:- Creating a role!-

Syntax!-

```
create role rolename;
```

Step 2:- Assigns system privileges to role!-

Syntax!-

```
grant systemprivileges to rolename;
```

Step 3:- Assigns role to number of users!-

Syntax!-

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

```
SQL> conn sys as sysdba;
```

Enter password: sys

```
SQL> create role r1;
```

```
SQL> grant create procedure, create trigger, create any index to r1;
```

```
SQL> grant r1 to scott, mureli;
```

→ In oracle all system privileges related to role stored under role-sys-privs' data dictionary.

e.g!- 

```
SQL> conn sys as sysdba;
```

Enter password: sys

```
SQL> desc role-sys-privs;
```

```
SQL> select role, privilege from role-sys-privs  
where role = 'R1';
```

ROLE	PRIVILEGE
R1	CREATE TRIGGER
R1	CREATE ANY INDEX
R1	CREATE PROCEDURE

Preddefined roles!-

Whenever we are installing oracle server then automatically three predefined roles are created these are

- 1) connect → used by end users
- 2) Resource → used by developers
- 3) dba → used by database administrators

→ Connect role internally having create session privilege which is used to allows users connect to the server

e.x! -  
SQL> desc role\_sys\_privs;  
SQL> select role, privilege from role\_sys\_privs  
where role  
in ('CONNECT', 'RESOURCE');  
SQL> select role, privilege from role\_sys\_privs  
where role in ('DBA');

e.x! -  
SQL> conn sys as sysdba;  
Enter password: sys  
SQL> create user kkk identified by kkk;  
SQL> grant connect to kkk;  
SQL> create table test (sno number (10));  
error! insufficient privileges.

11/9/2015

### Object Privileges :-

→ This privilege are given by either by database developer (or) database administrator. This privilege enables users to perform some operation on the object.

→ Oracle having select, update, delete, insert <sup>all</sup> execute, read, write object privileges.

Syntax:-  
grant object privileges on objectname to username / rollname / public

e.g:-  
SQL> conn scott/tiger;

SQL> grant update (ename) on emp to murali;  
SQL> grant all on emp to murali;  
SQL> grant all on emp to rcl;  
SQL> grant all on emp to public;  
SQL> conn murali/murali;  
SQL> select \* from scott.emp;

→ In oracle if you want to view all object privileges related to user then use one way user\_tab\_privs data dictionary

e.x! - SQL> desc user\_tab\_privs;

### Revoke!-

→ This command is used to cancel system or object privileges from user.

### Syntax!-

revoke systemprivileges from username1, username2;

### Syntax!-

revoke objectprivileges on objectname from username1, username2;

e.x!:-

sql> revoke all on emp from murali;

→ In all databases if you want to restrict table columns from one user to another user then we must create a view with required columns & then only that view given to the no. of users. group objectprivileges that's why ~~base~~ views only used for security.

### Syntax!-

grant all on viewname to username1, username2, ...;

e.x!:- sql> Conn scott/ tiger;

sql> create or replace view v1

as

Select empno, ename, sal from emp;

sql> grant all on v1 to murali;

sql> Conn murali/murali;

sql> Select \* from scott.v1;

### With check option!-

→ If you want to provide constraint type mechanism on views then we are using with check option clause when a view contains ~~with~~ check option clause then we are not allowed to use other than where cond' value from view to base table.

### Syntax!-

Create or replace view viewname

as

Select \* from tablename where condition with check option;

e.x!:-

sql> Create or replace view v1

as

Select \* from emp where deptno=10 with check option;

sql> Select \* from v1.

sql> ~~sql~~ Insert into v1(empno, ename, deptno) values (1, 'abc', 30);

Error: view WITH CHECK OPTION: where as clause violation.

`SQL> Insert into v1(empno, ename, deptno) values (1, 'abc', 10);`  
1 row created.

### Read only views! -

- when a view contains with read only clause then those type of views are also called as read only views.
- In these views we cannot performed dml operation.

e.x!- `SQL> create or replace view v2`

as  
`select * from emp where deptno=10 with read only;`  
view created

`SQL> delete from v2 where deptno=10;`  
error! .cannot delete from view!

### Force view (or) forced view! -

- We can also create a view without having base table these type of views are also called as force force view

#### Syntax! -

`create or replace force view Viewname`  
as  
`select * from anyname;`

e.x!- `SQL> create or replace force view v3`

as  
`select * from hyd;`  
warning! view created with compilation errors.

`SQL> create table hyd(sno number(10));`

`SQL> alter view v3 compile;`

`SQL> desc v3;`

Note! - In all databases through the views we can achieve logical data independence i.e whenever we are adding a column into the table it is not effected in view within external level.

#### e.x!- -

`SQL> conn scott/tiger;`

`SQL> grant all on v1 to manek;`

`SQL> alter table emp add address varchar(10);`

`SQL> select * from emp;`

```
SQL> Conn mureli / mureli;  
SQL> Select * from scott.v1;
```

→ we can also drop view by using drop view viewname

e.x:- SQL> drop view v1;  
view dropped.

### Merge Statement:-

- Oracle 9i introduced merge statement. This is a dml statement which is used to transfer data from source table into target table when table structure are same. In merge statement we use using update, insert statement  
→ Merge statement is also called as upsert statement. Merge statement is used in data warehousing application.

### Syntax:-

```
Merge into targettablename  
using source tablename  
on ('joining conditions')  
when matched then  
update set targettablecol = source-tablecol, ...  
where not matched then  
insert (targettablecolumns) values (source table columns);
```

e.x:-

```
SQL> create table dept$ as select * from dept;  
SQL> insert into dept$ (1,'a','b');  
SQL> merge into dept d  
using dept$ 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 merge statement we cannot update all columns that is we can not update on clause columns.

15/11/15

## SYNONYM

- Synonym is a database object which provide security because synonym hides another schema user name, object name.
- Synonym is an alias name (or) reference name of original object.
- Synonym also created by database administrator.
- All database systems having 2 types of synonym
  - 1) Private Synonym
  - 2) Public Synonym

→ In all databases by default synonym are private synonym

### Syntax:-

```
Create Synonym Synonymname forc
username. objectname @ databaseLink;
```

→ In oracle before we are creating public Synonym then we are using create public synonym system privilege to user

### Syntax:-

```
grant create public synonym to username;
```

### Syntax:-

```
create public Synonym Synonymname
forc username. objectname @ database Linkname;
```

SQl> scott/Hgerc	SQl> conn/murali/murali	SQl> conn al/al
SQL> insert all on emp to murali,al;	<pre>SQL&gt; Select * from scott.emp SQL&gt; Create synonym xy forc scott.emp; SQL&gt; Select * from xy; SQL&gt; Create public synonym xy for scott.emp;</pre> <p>Error! insufficient privileges</p>	<pre>SQL&gt; Select * from scott.emp SQL&gt; Select * from xy;</pre> <p>Error! Table (or) View does not</p>
	<pre>SQL&gt; Conn sys as sysdba Pass:- sys SQL&gt; Grant create public synonym to murali;</pre>	
	<pre>SQL&gt; Conn murali/murali; SQL&gt; Create public synonym xy for scott.emp; SQL&gt; Select * from xy;</pre>	
<pre>SQL&gt; drop synonym xy; Synonym dropped.</pre>		

- we can also drop synonym by using `drop synonym synonymname`
- All synonym information stored under `user_synonym` data dictionary
- SQL: `desc user_synonym;`

## SEQUENCE

- Sequence is a database object which is use to generate sequence numbers automatically.
- Generally sequences are used to generate primary key value automatically. Basically sequence is an independent database object once a sequence has been created then number of users simultaneously access that sequence.
- Generally sequences are created by database administrator.

### Syntax

```
create sequence sequenceName
  start with n
  increment by n
  minvalue n.
  maxvalue n
  cycle / no cycle.
  cache / no cache;
```

- If we want to generate sequence value then we are using following two pseudo column there are

- 1) `currval`
- 2) `nextval`

`currval` returns current sequence number where `nextval` return nextval sequence number.

### Syntax:-

- 1) `sequenceName.currval`
- 2) `sequenceName.nextval`

- These pseudo columns are used in `insert, update, delete, select` statements.
- In order if you want to access sequence value by using `select` statement then we must use `dual` table

### Syntax:-

- 1) Select sequence name. currval from dual;
- 2) Select sequence name. nextval from dual;

Ex:-

```
sql> create sequence s1  
      start with 5  
      increment by 2;
```

```
sql> select s1. currval from dual;  
error! Sequence s1. CURRVAL is not  
yet defined in this session.
```

```
sql> select s1. nextval from dual;  
      5
```

```
sql> select s1. nextval from dual;  
      7
```

```
sql> select s1. currval from dual;
```

→ If we want to generate 1st sequence number then we must use nextval pseudo column because currval pseudo column only returns current value of the sequence session if sequence session already having a value.

16/9/2015

```
sql> conn scott/tiger  
sql> create sequence s1;  
sql> select s1.nextval from dual;  
      1  
sql> select s1.nextval from dual;  
      2  
sql> select s1.nextval from dual;  
      3  
sql> select s1.currval from dual;  
      3
```

```
sql> conn scott/tiger  
sql> select s1.nextval from dual;  
      4  
sql> select s1.nextval from dual;  
      5  
sql> select s1.nextval from dual;  
      6  
sql> select s1.currval from dual;  
      6
```

Note!- In oracle we can also change sequence parameter value by using alter command.

Syntax!-

```
alter sequence sequencename  
parametername newvalue;
```

Note!- In oracle we cannot change starting sequence no. by using alter command

e.x!-

```
sql> create sequence s1  
      start with 5  
      increment by 1  
      minvalue 3  
      maxvalue 100;
```

```
sql> select s1.nextval from dual;
```

5

```
sql> /
```

6

```
sql> alter sequence s1  
      increment by -1;
```

```
sql> select s1.nextval from dual;
```

5

```
sql> select s1.nextval from dual;
```

4

```
sql> select s1.nextval from dual;
```

3

```
sql> select s1.nextval from dual;
```

error! Sequence s1.NEXTVAL goes below .MINVALUE and cannot be instantiated

```
sql> alter sequence s1
```

start with 4;

error! cannot alter starting sequence number.

Note!- Generally start with cannot be less than min value.

e.x!- sql> create sequence s1  
 start with 3  
 increment by 1  
 minvalue 5;

Note!- Generally sequences are used to generate primary key value automatically

e.x!- sql> create table test (orderno number(10) primary key, itemname varchar2(10));

```
sql> create sequence s1 start with 1001;
```

```
sql> insert into test(orderno, itemname) values (s1.nextval, &'itemname');
```

Enter value for itemname : abc  
 SQL> /  
 Enter value for itemname : xyz  
 SQL> /  
 Enter value for itemname : pqr  
 SQL> Select \* from test;

<u>ORDERNO</u>	<u>ITEMNAME</u>
1001	abc
1002	xyz
1003	pqr

SQL> create sequence s2;  
 SQL> alter table test add sno number(10);  
 SQL> Select \* from test;  
 SQL> update test set  
 sno = s2.nextval;

<u>ORDERNO</u>	<u>ITEMNAME</u>	<u>SNO</u>
1001	abc	1
1002	xyz	2
1003	pqr	3

### Cache!

→ Cache is a memory area which is used to free allocate set of sequence numbers. If you want to access sequence values very fastly then only we are using cache optional clause within sequence database.

→ Generally sequences are created in harddisk whenever user request they sequences then client process checks request and sequence no. is available in cache memory area.

→ If it is not available or the oracle server searches in harddisk if it is available then sequence values are transfer to cache memory area from that memory area only client process access sequence no. This process automatically decreases performance of the application. When they are accessing sequence value more no. of time. To overcome this problem oracle provided cache optional clause. In sequence database object which is used to access sequence value very quickly but whenever system crashes automatically

Cached values are lost.

Note!- In oracle by default cache value is 20 and also cache minimum value is 2

e.x!- create sequence s1

start with 1

cache 1;

error! the number of values to cache must be greater than 1.

→ All sequence information stored under `table user_sequences` data dictionary  
SQL> desc user\_sequences;

→ we can also drop sequence by using `drop sequence sequence name;`

## Index

→ Index is a database object which is used to improves performance of the applications.

→ Indexes are created on table columns whenever we are requesting data by using index columns then database server very fast retrieving data database.

→ Generally indexes are created by database administrator.

→ In all database we are creating indexes in two ways:

1) Automatically

2) Manually

1) Automatically!-

Whenever we are creating primary key or unique constraint then oracle server internally automatically creates btree indexes on those columns.

2) Manually!-

We can also create indexes exclusively explicitly by using following syntax.

Syntax!-

create index indexname on tablename (col1, col2, ...);

18/9/2015

→ Whenever select query contains where clause or order by clause then only oracle server searches for indexes within database. whenever where or order by clause columns having indexes then oracle server uses index scan mechanism for retrieving data very fastly from the database. if those columns does not have indexes then oracle server internally uses full table scan for retrieving data from the database

Note!- In oracle whenever where clause contain `<> (or) is null (or) is not null` operator then oracle server doesnot search for indexes if those columns

already having indexes also.

→ Oracle having two types of indexes

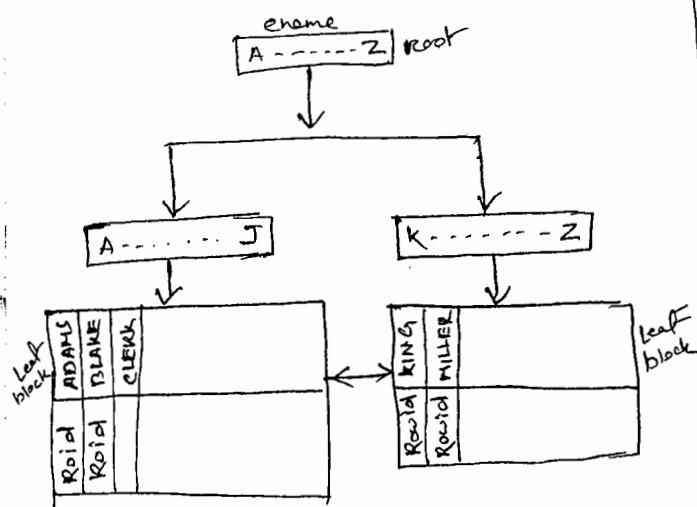
- 1) btree indexes
- 2) bitmap indexes

1) btree indexes :-

In oracle by default indexes are btree indexes. whenever we are creating btree indexes then oracle server automatically creates tree structure on indexed columns. In this tree structure allways leaf blocks stores actual data along with rowids. whenever we're requesting data by using where clause like under by clause then oracle server searches those columns having btree indexes. when btree indexes are available then oracle server internally uses index scan on btree structure to retrieve data very fastly from the leaf blocks.

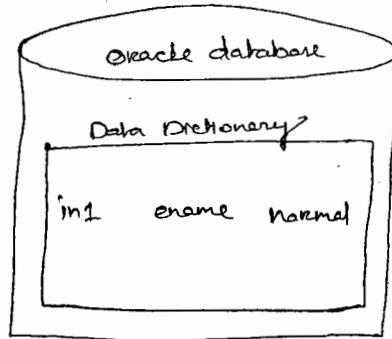
#### Developer

```
SQL> select * from emp  
      where ename = 'KING';
```



#### DBA

```
SQL> create index in1 on emp(ename);
```



Note:- In oracle all indexes information stored under user\_indexes data dictionary.

e.g:-  
SQL> create index in1 on emp(name);

SQL> desc user\_indexes;

SQL> select index\_name, index\_type from user\_indexes where table\_name = 'EMP';

INDEX_NAME	INDEX_TYPE
IN1	NORMAL
PK_EMP	NORMAL

Note:- In oracle if you want to view column names along with index name then we are using user\_index\_columns data dictionary.

Ex:-

```
sql> desc user_idx_columns;
sql> select index_name, column_name from user_idx_columns where
table_name = 'EMP';
INDEX_NAME          COLUMN_NAME
PK_EMP              EMPNO
IN1                 ENAME
```

### Calculating performance of a query in oracle through plan table:-

Step1:- use an explain plan for clause before select query by using following syntax

Syntax:-

```
sql> explain plan for select statement;
```

whenever we are submitting this query oracle server internally creates plan table based on query execution. If you want to view plan table then we are using display function from dbms\_xplan package by using following syntax.

Syntax:-

```
sql> select * from table(dbms_xplan.display());  
displayed from .
```

Step2:- (display plan table):-

Syntax:-

```
sql> select * from
table(dbms_xplan.display());
```

Example:- (without using indexes)

```
sql> select * from emp where ename = 'SCOTT';
```

Testing:-

```
sql> explain plan for select * from emp where ename = 'SCOTT';
sql> select * from table(dbms_xplan.display());
```

Example:- (with using index)

```
sql> create index in1 on emp(ename);
```

```
sql> select * from emp where ename = 'SCOTT';
```

Testing:-

```
sql> explain plan for select * from emp where ename = 'SCOTT';
```

```
sql> select * from table(dbms_xplan.display());
```

19/9/2015

### Function based indexes:-

- Oracle 8.0 introduced function based indexes by default function based indexes are btree indexes.
- In oracle whenever we are requesting data by using functions or expression in where clause then oracle server doesn't search for indexes if those columns having already indexes also to overcome this problem oracle 8.0 introduced extension of the btree indexes called function based indexes which create indexes on columns along with functions or expressions. then only oracle server searching for indexes.

#### Syntax:-

Create index indexname on tablename (functionname(columnname))  
or  
expression

→ stored expression.

#### e.x!- (without using function based indexes):-

SQL> select \* from emp where upper(ename) = 'SCOTT';

#### Testing!-

SQL> Explain plan for select \* from emp where upper(ename) = 'SCOTT';

SQL> select \* from table (dbms\_xplan.display());

#### e.x!- (with using function based indexes)

SQL> Create index in2 on emp (upper(ename));

SQL> Select \* from emp where upper(ename) = 'SCOTT';

#### Testing!-

SQL> Explain plan for select \* from emp where upper(ename) = 'SCOTT';

SQL> select \* from table (dbms\_xplan.display());

SQL> desc user\_indexes;

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

INDEX_NAME	INDEX_TYPE
IN2	FUNCTION-BASED INDEXES

### Virtual columns :-

- Oracle 11g introduced virtual columns in a table which store stored expressions directly into database. Prior to oracle 11g we can also stored stored expressions into oracle database indirectly by using views, function based indexes.
- In oracle 11g we can also stored stored expression directly in oracle database by using generated always as clause in virtual column

### Syntax:-

column name datatype [size] generated always as (stored expression) [virtual]

e.x!-

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 (80, 20);

sql> select \* from test;

a	b	c
80	20	100

Note:- In oracle if you want to view virtual column expressions then we can  
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';

COLUMN_NAME	DATA_DEFAULT
c	"A+B"

→ Oracle having 2 types of btree indexes

1) non unique btree indexes.

2) Unique btree indexes.

→ In oracle by default automatically created indexes are unique btree indexes

unique btree indexes performance is very high compare to non unique btree indexes

→ We can also create unique btree indexes explicitly by using following syntax

### Syntax:-

create unique index indexname on tablename (columnname);

Note:- we cannot create unique indexes on duplicate values columns

sql> create unique index ln3 on emp (ename);

Index created

sql> create unique index ln4 on emp (job);

error: cannot CREATE UNIQUE INDEX;

duplicate keys found.

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

### Bitmap index:-

→ Oracle 7.3 introduced. Bitmap indexes bitmap indexes are used in data warehousing

Applications

### Syntax:-

create bitmap index indexname on tablename (columnname);

→ Bitmap indexes are created on low cardinality columns

$$\text{cardinality of a column} = \frac{\text{number of Distinct values}}{\text{Total number of records}}$$

ex:-

1) cardinality of a empno col =  $10/14 \Rightarrow$  high cardinality  
 ↳ btree indexed.

ex:- 2) cardinality of a Job col =  $5/14 \Rightarrow 0.357 \dots \Rightarrow$  Low cardinality  
 ↳ bitmaps indexed.

21/9/2015

→ whenever we are creating bitmaps indexes internally oracle server automatically creates bitmap table by using no. of bits based on the indexed column values

→ whenever user requesting data by using logical operators or equality operators then oracle server directly operates bits within bitmap table then resultant bitmap automatically convert into rowid by using an internal bitmaps function

SQL> create tablebitmaps index. in1 on emp(job); | Developer  
 SQL> select \* from emp where job= 'CLERK';

Job	Bitmap table													
	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

→ we can also drop index by using drop index indexname;

SQL> drop index in1;

Note:- In all databases through the indexes we are achieving physical data independence.  
 i.e whenever we are adding indexes on table columns table structure does not change only performances is effected.

### Set Operators:-

→ set operators are used to retrieve data from single or multiple tables, Set operators are also called as vertical joins.

→ Oracle having following set operators.

① Union → returns unique value and also data is sorted.

② Union all → unique + duplicate

③ Intersect → it returns common values

④ Minus → values are in first query those values are not in second query.

e.x!-

```
SQL> select job from emp where deptno=10  
union  
select job from emp where deptno = 20;
```

JOB  
ANALYST  
CLERK  
MANAGER  
PRESIDENT

→ whenever we are using set operation always corresponding expression must belongs to same datatype & also set operations always returns first query column or alias names as column headings

e.x!-

```
SQL> select dname from dept  
union  
select ename from emp;
```

Note:- using set operations we can also retrieve data from multiple queries when corresponding expression not belongs to same datatype also. In this case we must use appropriate type conversion functions.

e.x!- SQL> select deptno from emp

```
union  
select dname from dept;
```

↑  
error! expression must have same datatype as corresponding expression

Solution!-

```
SQL> select deptno "deptno", to_char(null) "deptnames" from emp  
union  
select to_number(null), dname from dept;
```

deptno	deptnames
10	ACCOUNTING
20	OPERATIONS
30	RESEARCH
	SALES

Conversions !-

→ Converting one datatype into another datatype is called conversions

→ Oracle having two types of conversions

1) implicit conversion

2) Explicit conversion

1) Implicit Conversions!-

→ Here oracle server only automatically converts into one datatype to another datatype.

→ In oracle, if an expression contains string representing pure numeric value then oracle server automatically converts string type into number type.

e.x!-  
SQL> select sal + '100' from emp;

→ In oracle whenever we are passing numbers into character functions then oracle server automatically converts number type into character type.

e.x!-  
SQL> select length ('2456') from dual;  
Output  
4

→ In oracle whenever we are passing date string into pre-defined date functions then oracle server automatically converts date string into date type but here Parsed parameter must be default date format.

e.x!-  
SQL> select last\_day ('12-aug-05') from dual

31-AUG-05

### Implicit Conversion table

22/09/2015

From	To	Assignment	Expression Evaluation
Varchar or char	numbers	yes	yes
Varchar or char	Date	yes	yes
numbers	Varchar (or) char	yes	no
Date	Varchar (or) char	yes	no

### Explicit Conversion

→ Converting one datatype into another datatype explicitly by using explicit conversion function.

→ Oracle having following explicit conversion function.

#### decode () :-

→ It is a conversion function which is used to decoding the values.

→ decode function internally uses equality operator (=)

→ Decode function is also same as if---then--else construct within pl/sql

#### Syntax:-

decode (columnname, value, stmt1, value2, stmt2, .... stmts);

e.x!- SQL> select decode (1, 2, 3, 1, 5, 6, 7) from dual;

Output → 5

e.x:-

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

deptno	decode(deptno)
10	ten
20	twenty
30	others.

Q) write a query to update ~~sal~~ commission of the employee within emp table by using following condition ) if job = 'CLERK' then update comm  $\rightarrow$  10% of sal  
 2) if job = 'SALESMAN' then update comm  $\rightarrow$  20% of sal  
 3) if job = 'ANALYST' then update comm  $\rightarrow$  30% of sal  
 else update comm  $\rightarrow$  40% of sal.

`update emp set comm = decode(job, 'CLERK', sal * 0.1, 'SALESMAN', sal * 0.2,  
'ANALYST', sal * 0.3, sal * 0.4);`

`select * from emp;`

→ In oracle if you want to display aggregate values in tabular form & also rows converted into columns then we are using decode conversion function within groupby this type of reports are also called as pivot reports

e.x:-

`select job, sum(decode(deptno, 10, sal)) "deptno 10",  
sum(decode(deptno, 20, sal)) "deptno 20",  
sum(decode(deptno, 30, sal)) "deptno 30" from emp group by job;`

JOB	deptno 10	deptno 20	deptno 30
CLERK	2000	3000	1450
SALESMAN			2800
PRESIDENT	5300		
MANAGER	2750	3075	2950
ANALYST		6200	

e.x!- `select dname, sum(decode(job, 'CLERK', 1, 0)) "Clerks",  
sum(decode(job, 'SALESMAN', 1, 0)) "Salesman",  
sum(decode(job, 'ANALYST', 1, 0)) "Analyst",  
from emp e, dept d where e.deptno = d.deptno group by  
dname`

DNAME	Clerks	Salesman	Analyst
ACCOUNTING	1	0	0
RESEARCH	2	0	2
SALES	1	7	0

### Case Statement :-

- case statement is also used to decoding the value.
- case statement performance is very high compare to decode conversion function
- Oracle 8.0 introduced case statement & also oracle 8i introduced case conditional statement. this is also called as searched case.

Note:- Decode internally uses equality operation where as in case statement we can also use all SQL operators explicitly.

#### Method 1

```
case columnname when value1 then stmt1  
when value2 then stmt2  
-----  
else stmts end.
```

e.x:- `SQL> select ename, sal, deptno, case deptno  
when 10 then 'ten'  
when 20 then 'twenty'  
else 'others' end  
from emp;`

deptno	case (deptno)
10	ten
20	twenty
30	others

23/01/2015

#### Method 2:-

`SQL> select ename, sal, case  
when sal < 1000 then 'low salary'  
when sal between 1000 and 2000  
then 'medium salary'  
when sal in ( 2300, 2500, 2900, 3000)  
then 'special salary'  
else 'other salary' end  
from emp;`

## pivot () :-

→ Oracle 11g introduced pivot ()

→ pivot () is very high compare to decode () through this function we are converting number of rows into columns & also display aggregate function value in tabular form.

## Syntax :-

select \* from (select col1, col2, .... from tablename)

    pivot (aggregate function ()) for columnname in (value1, value2));

e.x:-

SQL> select \* from (select job, sal, deptno from emp)

    pivot (sum(sal) for deptno in (10 as deptno10, 20 as deptno20, 30 as deptno30));

JOB	deptno10	deptno20	deptno30
CLERK	2080	3080	1450
SALESMAN			2800
PRESIDENT	5300		
MANAGER	2750	3075 6200	2950
ANALYST			

SQL> select \* from (select deptno, job from emp)

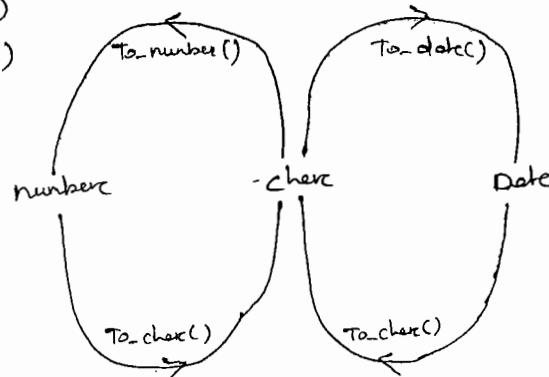
    pivot (count(\*) for deptno in (10 as deptno10, 20 as deptno20, 30 as deptno30));

JOB	deptno10	deptno20	deptno30
CLERK	1	2	1
SALESMAN	0	0	4
PRESIDENT	1	0	0
MANAGER	1	1	1
ANALYST	0	2	0

1) to\_number()

2) to\_char()

3) to\_date()



## 1) to\_number :-

→ It is used to convert a string representing numeric value with format into numeric value without format

e.x! - SQL> select '\$35.6' + 3 from dual  
error

SQL> select to\_number ('\$35.6') + 3 from dual  
error

Solution:-

SQL> select to\_number ('\$35.6', '\$99.9') + 3 from dual;  
Output  
38.6

e.x! - SQL> select to\_number ('a35.6') + 3 from dual;  
error

SQL> select to\_number ('a35.6', 'a99.9') + 3 from dual  
error: Invalid number format model.

to\_char(), to\_date() :-

Converting numbers into words

→ to\_char() having jsp format which takes Julian date & then spell out

e.x! - SQL> select to\_char(sysdate, 'jsp') from dual;

two million four hundred fifty-seven thousand two hundred eighty-nine.

Note:- If you want to view Julian date then we are using format 'j' within to\_char()

e.x! - SQL> select to\_char(sysdate, 'j') from dual;

2457289

Julian date:-

→ Julian date is number of dates since January 1, 4712 BC.

→ Julian date is represented as number. It is used to converting numbers into words by using to\_char, to\_date()

→ If you want to convert number into Julian date then we are using format 'j' within to\_date()

e.x! - SQL> select to\_date (123, 'j') from dual;

03-MAY-12

SQL> select to\_char (to\_date (123, 'j'), 'jsp')  
from dual;

Q) write a query to display given no. into word words by using to\_char, to\_date()?

to\_char() :-

→ to\_char is an overloading function i.e this function is used to convert number type into character type & also used to convert date type into date string.

Converting number type into character type:-Syntax:-

`to_char (number, 'fmt')`

format elements :-meanings

9

representing a number

G

group separator (,)

d

decimal indicator (.)

\$

dollar indicator

0

Leading zeros

L

local currency

e.x:- group separator (,)

`SQL> select to_char(1234567, '999,999,999') from dual;`

output

12,34,567

e.x:- decimal indicator (.)

`SQL> select to_char(1234567, '999,999,999.00') from dual;`

output

12,34,567.00

`SQL> select to_char(1234567, '999,999,999.99') from dual;`

error:- Invalid number

`SQL> select to_char(1234567, '99,99,999.99') from dual;`

error:- Invalid number

Note:- we can also use number of group separators in format model but we are allowed to use only one decimal indicator in format model

dollar ex:- `SQL> select to_char(123, '$999') from dual;`

\$123

Leading zeros

e.x:- `SQL> select to_char(123, '00999') from dual;`

00123

`SQL> select to_char(123, '99900') from dual;`

123  
→ space displayed

L → Local Currency :- (Default Currency dollar)

`SQL> select to_char(123, 'L999') from dual;`

\$123

Note!- whenever we are using local currency then oracle server returns \$, if we want to display our own local currency then we must use 3rd parameter in to\_char() this parameter must be specified within single quote in this parameter we must use our own local currency through D(nls\_currency) format

Syntax!-

'nls\_currency = own currency'

e.x:- SQL> select ename, to\_char(sal, 'L999,999,999.99', 'nls\_currency = Rs')  
from emp;

ename	to_char(sal)
SMITH	Rs 1,400,00
ALLEN	Rs 900.00

→ It is also used to convert date type into date style string

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

Output  
24/09/2015

to\_date!-

→ It is used to convert date string into date type

e.x:- SQL> select to\_date('22-aug-05') + 3 from dual;

Output  
25-AUG-05

null value functions

→ Oracle having following null value functions

- 1) nvl()
- 2) nvl2()
- 3) nullif()
- 4) coalesce()

D) nvl()!-

→ It is used to substitute (or) replace even value in place of null.

Syntax!-

nvl(exp1, exp2)

2) nvl2()!-

→ Oracle ai introduced nvl2() this function accepts 3 parameter

### Syntax:-

`nvl(exp1, exp2, exp3)`

→ Here if expression 1 is null then it returns exp3 otherwise it returns exp2

### 3) nullif () :-

→ Oracle 9i introduced nullif() this function accepts two parameters.

### Syntax:-

`nullif(exp1, exp2)`

Hence if exp1 = exp2 then it will returns null otherwise it will returns exp1

e.g. `> select nullif(10, 10) from dual;`

NULLIF(10,10)

e.g. `> select nullif(10, 20) from dual;`

Output

10

Q) write a query to display the employees who are getting less than 2000 salary & also display all other salary employees but in place of salary return null value by using nullif() from emp table?

sol) `> select ename, nullif(sal, greatest(2000, sal)) from emp;`

ename	nullif(sal)
SMITH	1400
ALLEN	900
WARD	
MARTIN	1500
KING	

### 4) coalesce () :-

→ Oracle 9i introduced coalesce() this function accepts number of expression this function returns first normal value within given expressions

### Syntax:-

`coalesce(exp1, exp2, ..., expn)`

e.g. `> select coalesce(null, null, 30, null, 50) from dual;`

Output

30

sol) `> select ename, comm, sal, coalesce(comm, sal) from emp;`

### Q) Difference between nvl(), coalesce() :-

→ nvl is a oracle() where coalesce() is a ansi() function both the functions returns same values when we are using two expressions & also nvl internally uses

implicit conversion i.e. whenever expression 2 is automatically converted into expression 1 then NVL() returns a value if expression 1, exp2 not belongs to same datatype also whereas in case of exp1, exp2 must belongs to same datatype.

e.g:-

```
SQL> select nvl('a', sysdate) from dual;
```

```
SQL> select coalesce('a', sysdate) from dual;
```

error: inconsistent datatypes!

expected CHAR got DATE

## NORMALIZATION

25/09/2015

→ Normalization is a specific process which is used to decomposing a table into number of tables. This process automatically reduces duplicate data and also automatically avoids insertion, update, deletion problems.

→ In design phase of the SDLC database designers designs logical model of the database in this logical model only designer 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 3 normal forms

### Normal forms

- 1) First normal form
- 2) Second normal form
- 3) Third normal form
- 4) BCNF
- 5) Fourth normal form
- 6) Fifth normal form

#### 1) First Normal form:-

→ If a table is in first normal form then in that table data in each columns should be atomic i.e. the data doesn't contains multiple value separated with  
(,) within single cell & also identifying a record uniquely by using a key.

(not in INF)  
Item table

Item name	color	price	tax
marker	black, red	30	0.3
pen	blue, green	20	0.2

INF.

(candidate key) Item table CINF table

Item name	color	price	tax
marker	black	30	0.3
marker	red	30	0.3
pen	blue	20	0.2
pen	green	20	0.2

Process :-

→ Identify repeating groups & put into separate table in more atomic form this is called first normal form table by default this is an child table because in this table one column having duplicate data.

Front End

Electronic Shop

order form

Order no:  ▾

order date:

cust Name:

address:

phone no.:

Item name:  Refrigerator  
 LED TV  
 add another item

Amount:

→ Primary key Order Master table

order no	order date	cust name	address	phone no.
1	-	-	-	-

→ Foreign key INF table (Order Detailed table)

order no	item name	amount
1	Refrigerator	30000
1	LED TV	40000

2) Second normal form:-

- If a table is in first normal form & also all non key attributes are fully functionally depended on total candidate key
- Generally first normal form deals with atomicity where as second normal forms deals with relationships between key, non key attribute.
- If a table is in first normal form & also that tables contains any partial non key attribute then that table not in second normal form.

process:-

- Identify partial non key attributes which depends on partial key attributes put into separate table this table is called Second NF table by ~~default~~ default this is a master table. In this table only all non key attributes fully functionally depended on total candidate key.

E.X:-

Child Table

Primary key: Item name

Non-key attribute: color

Partial dependency: Dept. dependent on item name

Partial dependency: Tax dependent on item name

2NF

Foreign key: Item name

Non-key attribute: Color

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

Primary key (Master table) 2nd Table

Item name	Price	Tax

Students Activity table

Stno	Sname	Activity 1	cost 1	Activity 2	cost 2
101	abc	cricket	\$10	football	\$20
102	xyz	cricket	\$10	golf	\$30
103	pqr	football	\$20	golf	\$30
104	zzz	football	\$20		

Stno	Sname
101	abc
102	xyz
103	pqr
104	zzz

Activity table

Stno	Sname	Activity	cost	Activity	cost
101	abc	cricket	\$10	football	\$20
102	xyz	cricket	\$10	golf	\$30
103	pqr	football	\$20	golf	\$30
104	zzz	football	\$20		

Primary key: Stno

Foreign key (Child table): Activity

Foreign key (Master table): Activity

2NF

Stno	Sname	Stno	Activity	Activity	cost
101	abc	101	cricket	cricket	\$10
102	xyz	101	football	football	\$20
103	pqr	102	cricket	football	\$20
104	zzz	102	golf	golf	\$30
		103	football	golf	\$30
		103	golf	football	\$20
		104	football	football	\$20

Primary key: Stno

Foreign key (Child table): activity

Partial dependency: Activity

Non-key attribute: cost

2NF (not in 2NF)

Non-key attribute: cost

2NF

Stno	activity	cost
101	cricket	\$10
101	football	\$20
102	cricket	\$10
102	golf	\$30
103	football	\$20
103	golf	\$30
104	football	\$20

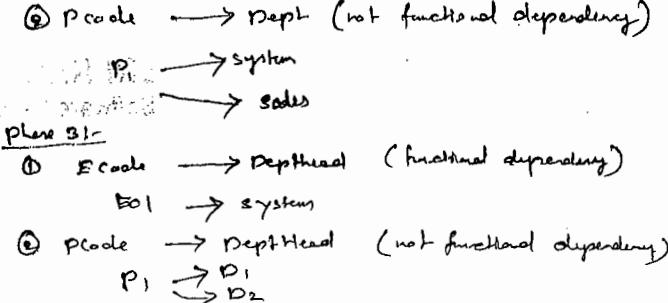
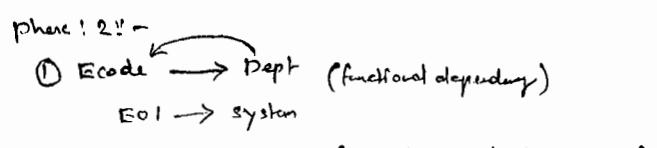
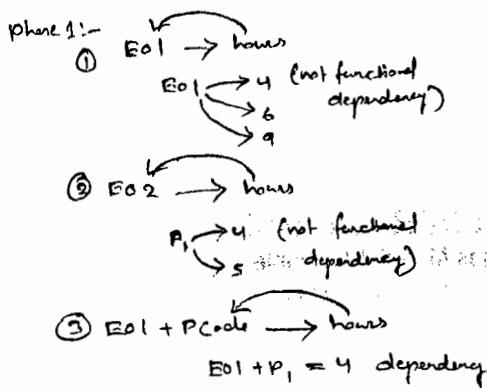
Primary key: Ecode

Non-key attribute: Dept

Non-key attribute: DeptHead

Non-key attribute: hours

Ecode	Pcode	Dept	DeptHead	hours
E01	P1	System	D1	4
E02	P1	Sales	D2	5
E03	P2	Research	D3	7
E01	P2	System	D1	6
E02	P3	Sales	D2	8
E01	P3	System	D1	9



Primary key (2nf Table) (Master Table)		
Ecode	Dept	DeptHead
E01	System	D1
E02	Sales	D2
E03	Res	D3

Foreign key (Child Table)		
Ecode	Pcode	Hname
E01	P1	4
E02	P1	5
E03	P2	7
E01	P2	6
E02	P1	8
E01	P3	9

→ In the above ex before 2nf process above resource table having insertion, updation, deletion problem.

#### Insertion Problem :-

→ In the above resource table when we are inserting a particular dept employee then we must assign project details. If those details are not available we need to supply null value for those attributes this is called insertion problem.

#### Updation Problem :-

→ In the above resource table ecode, dept, depthead attributes values are repeated whenever an employee transfer from one dept to another dept then we need to modify all these 3 attribute values correctly otherwise inconsistency problem occurred this is called updation problem.

#### Deletion Problem :-

→ In the above resource table when we are trying to delete employee record then automatically department details also deleted this is called deletion problem.

→ All these 3 problems are automatically avoided when we are using normalization process.

29/9/2015

#### Functional dependency :-

If any given two tuples in a relation  $r$  then  $x$  (an 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

$x$  is functionally determines  $y$ .

#### Q) Why Normalization process?

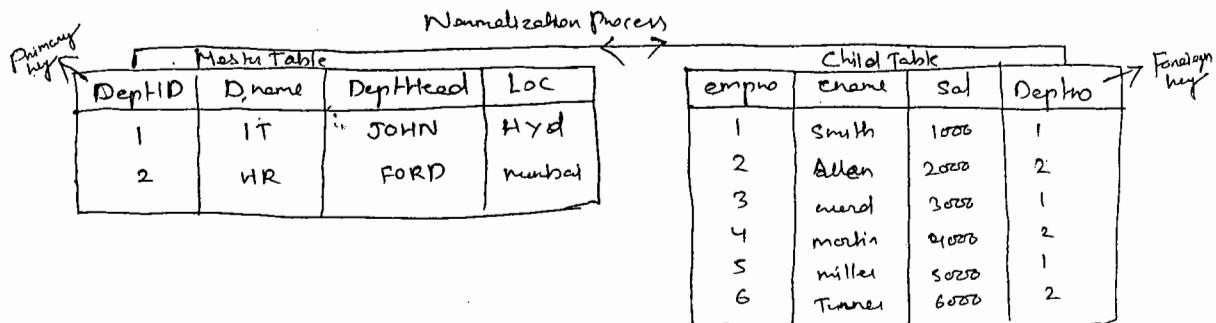
Normalisation is a scientific process which is used to reduce redundancy & also automatically avoids insertion, updation, deletions problems

**SRI RAGHAVENDRA XEROX**  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

EMP					
empno	ename	Sal	Dname	DeptHead	Loc
1	Smith	1000	IT	JOHN	Hyd
2	Allen	2000	HR	FORD	Mumbai
3	Ward	3000	IT	JOHN	Hyd
4	Martin	4000	HR	FORD	Mumbai
5	Miller	5000	IT	JOHN	Hyd
6	Turner	6000	HR	FORD	Mumbai

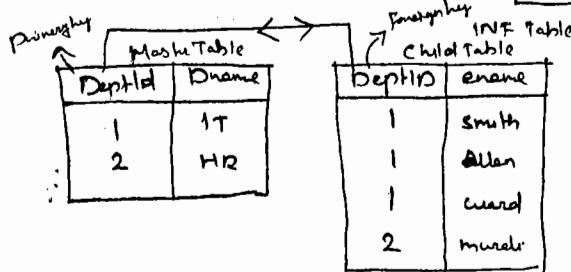
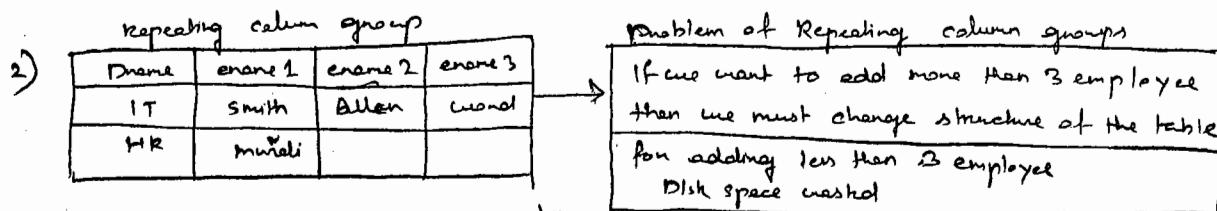
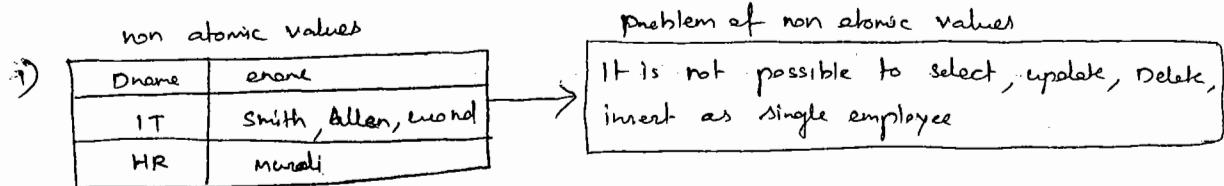
### Problems on Data Redundancy

- 1) Disk space wastage
- 2) Data inconsistency
- 3) DML query become slow.



### A table is in INF

- 1) Data in each column should be atomic. There is no multiple values separated with comma.
- 2) Table does not contain repeating column groups.
- 3) Identifying a record uniquely by using primary key.



### Third normal form! -

→ If a table is in second NF & also all non key attributes are only dependent on candidate key (primary key)

→ If a table is in 2NF & also any non key attributes which depends on another non key attributes then that table not in 3rd NF

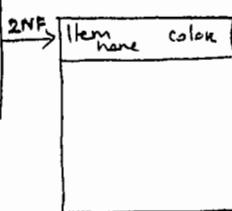
Process :-

→ Identify non key attributes which depends on another non key attributes put into separate table this table is called 3NF table by default this table also master table In this table only all non key attributes are only dependent on primary key

e.g:-  
Item Table

Item name	color	price	tax
marker	black	30	.3
marker	Red	30	.3
pen	blue	20	.2
pen	green	20	.2

nonkey attribute



Customer 2nd NF Table

Item name	price	tax

pk (Multi-table)  
2nd NF

price	tax

order form

order no \_\_\_\_\_  
order date \_\_\_\_\_

cust no. \_\_\_\_\_

item name

add another item

Amount

submit

Employee form

custno \_\_\_\_\_  
custname \_\_\_\_\_  
address \_\_\_\_\_  
phoneno \_\_\_\_\_

OK

qty  
discount

Customer Master Table  
3rd Table

custno	custname	addr (address)	phoneno.

30/09/2015  
Order Master Table

orderno	orderdate	custno.

Item Master Table

itemno	itemname	amount

Ordered details Table

orderno	itemno	qty	Discount

candidate key (not in 3NF)  
Student Table

studentno	courseid	Graade	GrValue
abc	CS111	A	4.00
xyz	CS111	B	3.00
pqr	ES222	C	2.00
z22	CS222	A	4.00

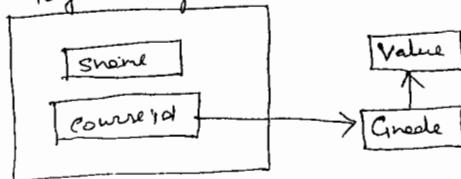
nonkey attribute

Shname	CourseId	Graade
abc	CS111	A
xyz	CS111	B
pqr	CS222	C
z22	CS222	A

FK (Master table)  
3rd table

Graade	GrValue
A	4.00
B	3.00
C	2.00

Logical Diagram



1NF → remove repeating groups

2NF → remove partial attributes

3NF → remove attributes which are not dependent on key / remove non key attributes which are dependent on another non key attributes

## cluster

- Cluster is an database object which contains group of table together & it will share same datablock.
- Cluster are used to improves performance of the joins.
- Generally clusters are created by database administrator. 1/10/2015
- Cluster tables must have common column name this common column is also called as cluster key.
- In all databases whenever we are submitting inner join or outer join a database server checks from clause tables are available in cluster or not if those tables are available in cluster then database server very fastly retrieve data from those tables.
- Generally clusters are created by database administrator at the time of table creation.

Step1:- Create a cluster using common column-name.

Syntax:-

```
create cluster clustername (common columnname col datatype (size));
```

Step2:- Create an index on cluster

Syntax:-

```
create index indexname on cluster clustername;
```

Step3:- Create cluster tables

Syntax:-

```
create table tablename (col1 datatype (size), col2 datatype (size), ...)  
cluster clustername (common columnname);
```

e.x:-

```
SQL> create cluster emp_dept (deptno number (10));
```

```
SQL> create index in1 on cluster emp_dept;
```

```
SQL> create table e1 (empno number (10), ename varchar2(10), deptno number(10))  
cluster emp_dept (deptno);
```

```
SQL> desc e1;
```

```
SQL> create table d1 (deptno number (10), dname varchar2(10), loc varchar2(10))  
cluster emp_dept (deptno);
```

```
SQL> desc d1;
```

```
SQL> insert into e1 values (1, 'a', 10);
```

```
SQL> insert into d1 values (10, 'b', 'hyd');
```

```
SQL> select * from e1;
```

```
SQL> select * from d1;
```

SQL> select rowid from e1;

SQL> select rowid from d1;

Note:- In all databases by default cluster table having same rowid.

Note:- we cannot drop clusters if cluster having tables to overcome this problem

Oracle 8i introduced including tables clause to drop cluster along with tables.

Syntax:-

drop cluster clustername including tables;

e.x:-

SQL> drop cluster emp\_dept;

error! cluster not empty

SQL> drop cluster emp\_dept including tables;

→ In oracle all clusters information stored under user clusters data dictionary

SQL> desc user\_clusters;

Super key, Candidate key:-

Superkey → A column or combination of column which uniquely identifies the record is called Superkey.

Candidate key → A minimal superkey which uniquely identify a record is called candidate key.

(orc) A superkey which is a subset of another superkey then that total key is not a candidate key

e.g:-

empno	ename	skillid	skill	votcid
1	murali	1	Oracle	V1
1	murali	2	Teradata	V1
1	murali	3	Sybase	V1
2	abc	4	DB2	V2
2	abc	1	Oracle	V2
3	xyz	5	Informix	V3
4	zzz	5	Informix	V4
4	zzz	4	DB2	V4
5	pqr	6	Sasview	V5

Superkey

- ① empno + skill
- ② empno + ename + skill
- ③ empno + votcid + skill
- ④ ename + skill
- ⑤ ename + votcid + skill
- ⑥ votcid + skill
- ⑦ empno + ename + votcid + skill

Candidate key

- ① empno + skill
- ② ename + skill
- ③ votcid + skill

not a Superkey

- ① empno + ename
- ② empno + votcid
- ③ ename + votcid

BCNF:-

→ If a table is in BCNF in that table every determinant is a candidate key.

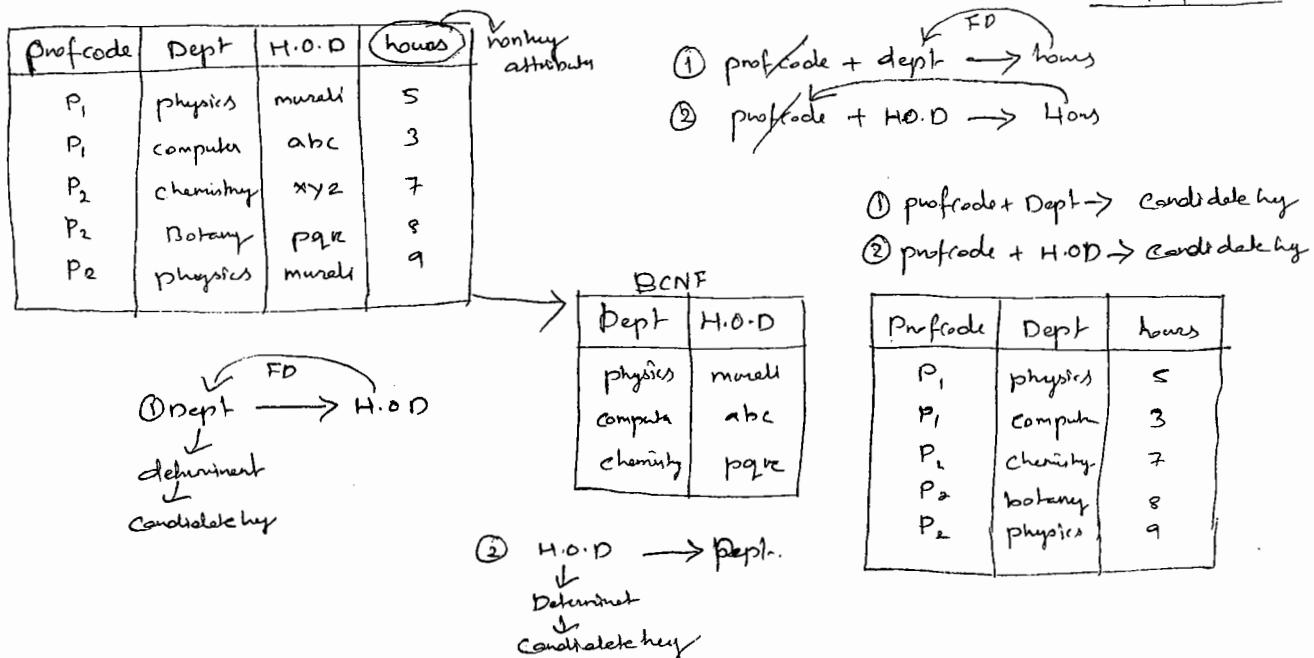
→ When a table having multiple composite candidate key and also those candidate keys are overlap and also one candidate key non-key attributes which depends on another composite candidate key non-key attribute then only we are using BCNF process.

- Generally 2nd, 3rd NF deals with relationships bet'n key, non-key attribute, whereas BCNF deals with relationships between non-key attributes within composite candidate keys itself.
- Whenever a table contains multiple composite candidate key then deletion problem occurs to over come this problem database designer uses BCNF

### Process:-

Identify non key attribute from a candidate key which depends on another non-key attributes in another candidate key put into separate table these table is called BCNF table. In this table only every determinant behaves like a candidate key. These table also behaves like a master table. This BCNF table does not contain non-key attribute.

02/10/2015

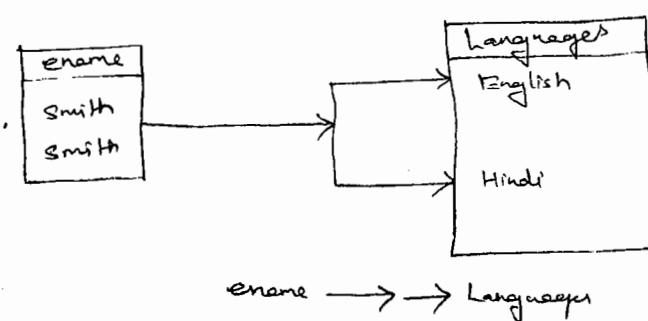


### Multivalued Dependency:-

- One column having multiple rows that matched with single value in another column within a same table is called multi value dependency.
- multivalue dependency is represented by ( $\rightarrow\rightarrow$ )

e.g:-

multivalue dependency



### Fourth Normal Form:-

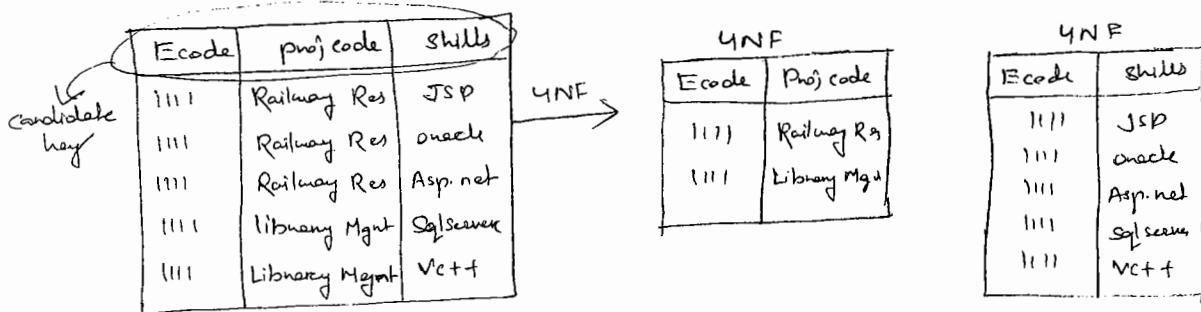
- If a table is in fourth normal form then that table doesn't contain more than one independent multivalue attribute.
- If a table contains more than two attributes & also identifying a record uniquely by using combination of all these 3 attributes & also one attribute set of values which depends on another attribute set's values & also same attribute set's values which depends on another attribute set's value and also some attributes are not logically related then only we are using 4th NF process.

### Process:-

- Identify Independent multivalue attributes putting separate tables these tables are called 4NF tables. These tables doesn't contain more than one independent multivalue attribute.
- Generally when a table having more than one independent multivalue attributes then those tables having more duplicate data for reducing these duplicate data database designer uses 4th NF.

### Assumption 1:-

- 1) Each employee having multiple projects
- 2) Each employee having multiple skills



Ecode	proj code	Skills
1111	Railway Res	JSP
1111	Railway Res	oracle
1111	Railway Res	Asp.net
1111	Library Mgmt	Sql Server
1111	Library Mgmt	Vct +

4NF

Ecode	Proj code
1111	Railway Res
1111	Library Mgmt

4NF

Ecode	Skills
1111	JSP
1111	oracle
1111	Asp.net
1111	Sql Server
1111	Vct +

- Before 4th NF process in the above resource table when an employee got new skills then we need to supply null value for the project code. & also when an employee got new project then we need to supply null values for this skills these null value problem is automatically avoided in 4th NF process tables & also these tables reduces duplicate data

### Fifth Normal Form !-

- When a relation cannot be decomposed into number of relation is called Fifth NF.
- Fifth NF is also called as projection joined normal form because here also resource tables having multivalue attribute same like a 4NF tables but here all attributes are logically related when we are decomposing this table

into no. of tables. Then we must check whether they are joining these table then result and record must be available in resource table or not.

## LOCKS

→ Locking is a mechanism which prevents unauthorised access for our resource.

→ All database system having two types of lock

- 1) Row level locks
- 2) Table level locks

### 1) 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];
```

→ Whenever we are performing locks then another user query the data but they cannot perform dml operations & also in all databases whenever we are using commit or rollback then automatically locks are released.

e.x!:-

```
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;  
[cannot perform DMLS]
```

#### Nowait:-

→ It is an optional ~~after~~ clause use along with for update clause. whenever we are using nowait clause automatically controls returns to the current session if another not releasing locks also in this case oracle never returns an error.

[ora-0054 : resource busy]

e.x!:-  
sql> conn. scott/tiger  
sql> select \* from emp where  
 deptno=10 for update nowait;  
 ora-0054: resource busy  
sql> 1

```
sql> conn. murali/murali  
sql> select * from scott.emp  
      where deptno = 10 for update;
```

3/10/2015

Note! - In all databases whenever we are using DML statement then automatically database servers internally uses exclusive locks (default locks)

Ex! - SQL> conn scott/tiger

SQL> update emp set sal = sal + 100  
where deptno = 10;  
3 rows updated

SQL> commit;  
[For releasing locks]

SQL> conn murali/murali

SQL> update emp set sal = sal + 100  
where deptno = 10;  
[We cannot perform DML operation  
because of internal locks]

(after commit/rollback statement only  
another user can update)  
3 rows updated.

### Table level locks:-

In this method DBA performs locks on a table once having 2 types of table locks

- 1) Share lock.
- 2) Exclusive lock.

#### 1/ Share lock! -

When we are using these lock another user query the data but they cannot perform DML operation and also number of users lock the resource at a time.

#### Syntax! -

Lock table tablename in share mode;

Ex! -

SQL> conn scott/tiger  
SQL> lock table emp in share  
mode;  
SQL> commit; (for releasing lock)

SQL> conn murali/murali

SQL> select \* from scott.emp (locking)  
SQL> lock table scott.emp in share mode;  
SQL> update scott.emp set sal = sal + 100;  
[We cannot perform DML]

#### 2/ Exclusive lock! -

When we are using this lock then another user query the data but they cannot perform dml operation & also at a time only one user lock the resources.

#### Syntax! -

lock table tablename in exclusive mode;

Ex! - SQL> conn scott/tiger  
SQL> lock table emp in  
exclusive mode.

SQL> conn murali/murali

SQL> lock table scott.emp in share mode  
[We cannot perform any locks]

Note:- In all databases whenever we are using cursor locking mechanism then internally database servers uses exclusive locks.

### Hierarchical Queries

- In all relational databases we can also store hierarchical data in relational tables.
- If we want to store hierarchical data then relational table must contain minimum 3 columns & also in these 3 columns 2 columns must belongs to same datatype & also data must be related.
- If we want to retrieve hierarchical data then we are using following clause
  - 1) level
  - 2) Start with
  - 3) Connect by

#### 1/ Level :-

→ Level is a pseudo column which automatically assigns numbers to each level within tree structure.

#### 2/ Start with :-

→ Through the start with clause we can specify searching condition with in hierarchy

#### Syntax :-

start with condition.

#### 3/ Connect by :-

→ Through the connect by clause we can specify relationships between hierarchical columns by using prior operator.

#### Syntax :-

connect by prior parentcolumnname = childcolumnname

#### Syntax :-

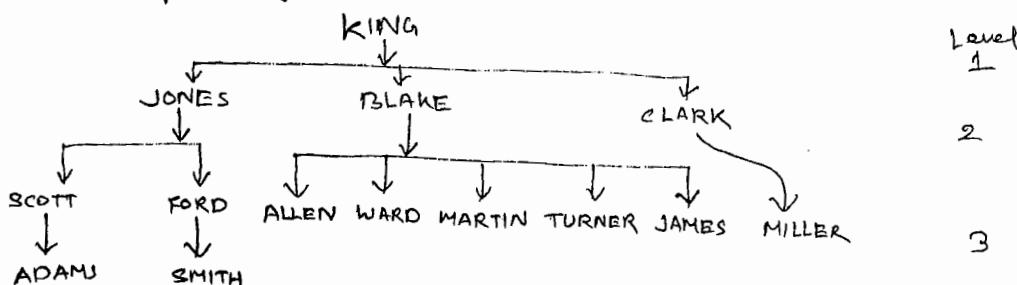
Select level, columnname from tablename where condition

Start with condition connect by prior parentcolumnname = childcolumnname;

Q) write a hierarchical query display the employees who are working under other employee & also display the manager from emp table root node onwards

Ans) Select level, ename from emp start with mgr is null connect by

prior empno = mgr;



Note:- Oracle 9i introduced sys\_connect\_by\_path() which returns path of the hierarchy within tree structure

Syntax:-

sys\_connect\_by\_path (columnname, 'delimitername')

Ex:- select level, sys\_connect\_by\_path (ename, '→')

from emp start with mgr is null connect by prior empno = mgr

Q) write a hierarchical query to display the employees 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

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 mgr

level    ename

1	→MILLER
2	→MILLER → CLARK
3	→MILLER → CLARK → KING

Prior :-

→ Prior is an unary operator used in ~~connect~~ by clause when we are using prior operator in front of the child column (empno) then oracle server uses top to bottom search within hierarchy where as when we are using prior operator in front of the parent column(mgr) then oracle server uses bottom to top search within hierarchy

Execution:-

→ whenever we are submitting hierarchical query then oracle server take a value from prior operator column based on start with condition then only oracle server ~~and~~ searches this column value available in another hierarchical column. If that value is available then oracle server retrieves corresponding data from the 2nd column



Control files:-

- It controls all other files in storage area. Their file extension is .ctl
- Control files also stores logically created databases information
- These control files are used by DBA in ~~Backup~~ recovery process.
- In oracle all control files information stored under "V\$controlfile" data dictionary

Example

```
SQL> conn sys as sysdba;
Enter password: sys
SQL> desc V$controlfile;
SQL> select name file from V$controlfile;
```

Redolog file:-

- Redolog files store committed information from Redolog Buffer. These file extension is ".log".
- Redolog files also used by database Administration in backup and recovery process.
- In oracle all Redolog file information stored under V\$log file data dictionary
- for e.g. - SQL> desc V\$log file;
- SQL> select MEMBER from V\$log file;

Instance :-

- whenever we are connecting to a database by using a tool then Oracle server creates automatically creates a memory area. This memory area is also called as Instance.
- Oracle Instance consists of 2 parts
  - SGA
  - Process
- 1) SGA :- It is also called as System Global Area (or) Shared Global Area. SGA memory having set of buffers there are
  - Database buffer cache
  - Shared pool
  - Java pool
  - Large pool
  - Redolog buffer.

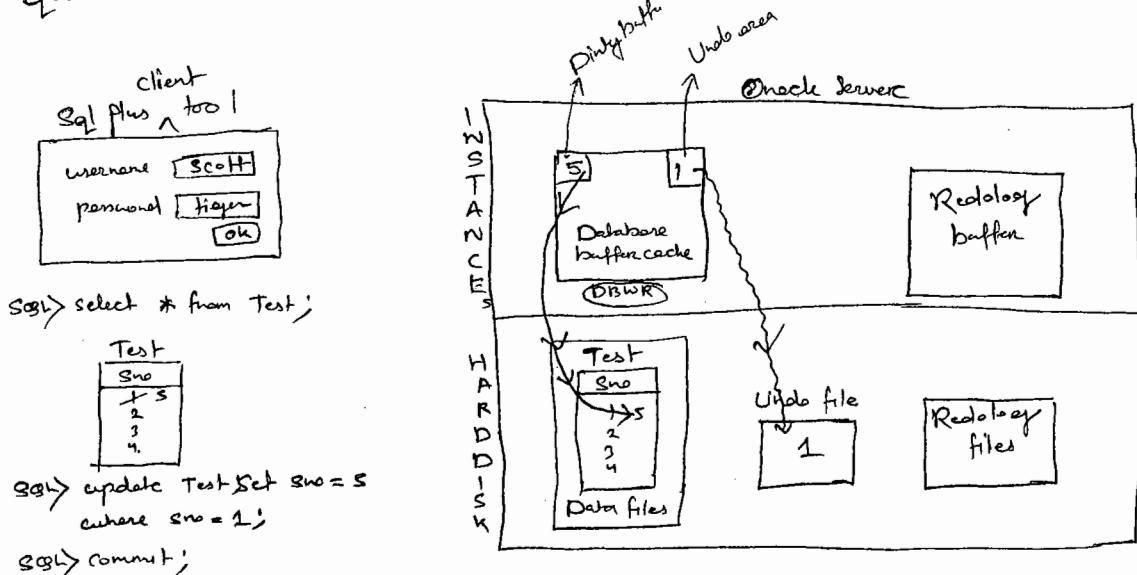
Whenever we are submitting SQL, PL/SQL code then that code is stored in Library cache within shared pool. Library cache always reduces parsing. Shared pool having dictionary cache which checks user privileges.

Whenever user requesting a table by using select statement then Server process checks request table is available in database buffer cache if requested table is not available then DBWR process checks request table is available in datafiles.

within storage area if that table is available then copy of the table is transferred into database buffer cache. Then only server process retrieve table information from database buffer cache & also Java pool executes Java related objects & also large pool used by oracle server which processes large amount of data & also redolog Buffer store new information for the ~~concept~~ transaction.

### undo!-

- In oracle whenever are are performing DML transaction then new values for the transaction stored in dirty Buffer and old value is stored in undo area within database buffer cache.
- Whenever are giving commit, then new value for the transaction is permanently stored in data files and also old value for the transaction internally stored in undo file with storage area - These files are used by oracle server in flashback queries.



### Flashback queries!-

- Oracle 9i introduced flashback queries. These queries are handled by database Administrator. Flashback query retrieve accidental data from the database after committing the transaction also i.e flashback query retrieve accidental data with reference to a specific point of time by using (as of) clauses
- Flashback query internally uses undo file.
- Through the flashback queries we are retrieving data in two ways.
  - 1) Using Time Stamp
  - 2) Using scn

## ) Using TimeStamp:-

### Syntax:-

Select \* from tablename as of timestamp (particular point of time);

→ Oracle 9i introduced timestamp datatype which is used to store & fraction of seconds within database.

### Syntax:-

columnname timestamp

→ If we want to insert data into timestamp datatype then we are using Systime timestamp function.

### e.g:-

```
SQL> Create table k1 (col1 timestamp);
SQL> desc k1;
SQL> Insert into k1 values (systimestamp);
SQL> select * from k1;
```

### col1

06-OCT-15 12.16.12.562000 PM

```
SQL> Insert into k1 values (sysdate);
SQL> select * from k1;
```

### col1

06-OCT-15 12.18.00.000000 PM

→ Oracle 9i also introduced interval function which is used to add or subtract time intervals in systimestamp function

### Syntax:-

Timestamp + interval 'number' minute (or) hour (or) sec

### e.g:-

```
SQL> Create table test (sno number(10));
SQL> Insert into test values (.....);
SQL> Commit;
SQL> Select * from test;
```

### sno

1
2
3
4

```
SQL> Delete from test;
```

4 rows deleted.

```
SQL> Commit;
SQL> Select * from test; (no row selected)
```

### flashback query:-

```
SQL> Select * from test as of timestamp (systimestamp - interval '1' minute);
```

SNO
1
2
3
4

### Process:-

→ Oracle instance also having set of background process these are

- 1) DBWR (database writer)
- 2) LGWR (log writer)
- 3) CKPTC (check pointer)
- 4) SMON (system monitor)
- 5) PMON (process monitor)

### DBWR:-

DBWR process always fetches data from database buffer cache and store the data permanently into data file.

### LGWR:-

→ Log writer fetches data from Redolog buffer and stores the data into Redolog files.

→ Generally whenever we are using DML transaction new value for the transaction also stored in redolog buffer. Whenever we are using Commit or  $\frac{1}{3}$  fill of redolog buffer then log writer process fetches data from redolog buffer into redolog file these redolog files are used by database administrator in backups & recovery process.

### CKPTC:-

→ Whenever we are performing transactions then DBWR process stored those transaction into datafiles. In this case check pointer automatically creates an unique identification number for the transaction in datafiles, control files, redolog files. This number is also called as SYSTEM\_CHANGE\_NUMBER (SCN). Using these SCN numbers also we can retrieve accidental data by using flashback queries.

### Syntax:-

Select \* from tablename as of scn scnnumber;

→ In oracle if you want to give scn number then we are using "current\_scn" properties from v\$ database data dictionary.

e.g:- SQL> conn sys as sysdba;

Enter password sys

SQL> create table t1 (sno number (10));

SQL> desc v\$database;

SQL> select CURRENT\_SCN from v\$database;

CURRENT\_SCN

2440832

SQL> Insert into b1 values (70);

SQL> commit;

### Freshback Query:-

SQL> select count(\*) from b1 as of scn 2440832;

Count (\*)

0

7/10/2015

### smon:- (System monitor)

→ System monitor process monitoring all other background process.

→ System monitor process is also called as instance recovery because this process automatically recovery any other background process if problem is occurred.

### pmon:- (process monitor)

→ pmon process automatically de-allocate user process when clients not still connected from this server also

pga!:-

→ pga is also called as private global area.

→ pga memory area is available in server process pga memory area stores session specific information i.e it stores all clients information which is connected to the server that's why pga memory area uniquely identifies each and every client connect to the server.

→ pga memory area is also having separate memory area called sql work area which is used to executes collections in pl/sql

Pga

logon	session	sql workarea
-------	---------	-----------------

### Nested table

**SRI RAGHAVENDRA XEROX**  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

→ Oracle 8.0 introduced nested table

→ Table within another table is called nested table if you want to store table within another table then we must create user defined datatype within table column. This user defined datatype is also called as nested table type. In oracle we can also create our own user defined datatype by using type keyword. Basically nested table is an user defined datatype which is used to store number of data item in a single unit.

→ In oracle if you want to store multiple data item in oracle memory area then we use using index by table in pl/sql but these tables are not allowed to stores 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 stores permanently into oracle database by using sql

### Object:-

→ Object is an user defined type which is used to represent different data types into single unit.

#### Creating nested table

Step1: Create an object type

Step2: Create an nested table type

Step3: Create an relational table

#### Step1: Create an object type :-

→ Before we are creating nested table type then we must create an object type by using following syntax.

##### Syntax:-

```
create or replace type typename as object (attribute datatype (size),  
-----);
```

#### Step2: Create an nested table type ! from object type! :-

##### Syntax:-

```
create or replace type typename as table of objecttypename;
```

#### Step3: Create an relational table !

##### Syntax:-

```
create table tablename (col1 datatype (size), col2 datatype (size),  
-----  
col nested table type) nested table col store as anyname;
```

e.g!-

```
sql> create or replace type obj1 as object (book id number (10), book name  
varchar 2(10), price number (10));
```

```
sql> /
```

```
sql> create or replace type xy as table of obj1;
```

```
sql> /
```

```
sql> create table student (stno number (10), sname varchar 2(10),
```

```
col3 xy ) nested table col3 store as aaa;
```

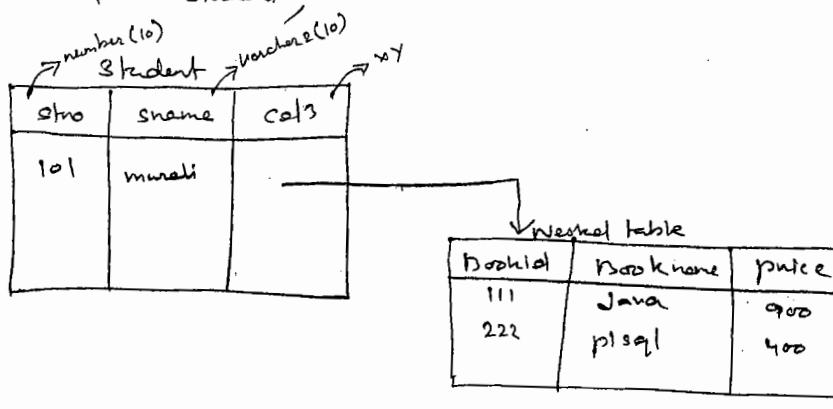
```
sql> close student; → same name → anyname
```

Name	Type
STNO	NUMBER(10)
SNAME	VARCHAR2(10)
COL3	XY

→ If you want to store data into nested table column then we must use constructor here constructor name is also same as type name

SQL> insert into student values (101, 'murali', XY (obj1(111, 'java', 900), obj1(222, 'plsql', 400)));

SQL> select \* from student;



→ We can also select, update, delete nested table data if you want to those operation then we must use table operators along with sub-query in there subquery we must specify nested table column this table operator we replace of relational table name in select, update, delete statements

e.g:-

SQL> select \* from table (select col3 from student);

BookId	BookName	price
111	Java	900
222	plsql	400

SQL> update table (select col3 from student) set price = 1000 where BookId = 111;

SQL> select \* from table (select col3 from student);

BookId	BookName	price
111	Java	1000
222	plsql	400

Partitions :-

→ A table can be logically decomposed into numbers of tables is called partitions this partitions tables are created by database Administration only partitions tables are used in data warehousing implementation.

- partition tables are treated as very large databases. partition table are used to improved performence of the application in background recover process ... partitions table are created based on key column - This key column is also called as partitioning.
- Oracle having following type of partition.

- 1) Range partition
- 2) List partition
- 3) Hash partition

### Range partition:-

8/10/2015

- In this method creating partition based on range of values

#### Syntax:-

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

partition by range (key columnname) (partition partition1 values less than (Value),
-----,
-----,
partition partitionname values less than (maxvalue));
```

#### To view particular partitions:-

#### Syntax:-

```
select * from tablename partition (partitionname1, partitionname2, ...);
```

#### e.g:-

```
SQL> create table test (sno number(10), name varchar2(10), sal number(10))

partition by range (sal) (partition p1 values less than (1000),
                           (partition 2 values less than (2000), partition other values less than (max value));
```

```
SQL> desc test;
```

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

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

#### To view particular partitions:-

```
SQL> select * from test partition (p1);
```

SNO	NAME	SAL
1	a	500

```
SQL> select * from test partition (p2);
```

SNO	NAME	SAL
2	b	1000
3	c	1500

## 2) List partition:-

→ Oracle 9i introduced list partition.

→ Generally using list partition database administrator create partitions based on character datatype column there partitions is ~~for~~ ~~so~~ created based on list of values.

### Syntax:-

```
create table tablename (col1 datatype (size),....)
partition by list (keycolumnname)
partition partitionname1 values (Value1, Value2,....)
-----;
partition partitionname values (default);
```

e.g:-

```
SQL> create table test1 (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> desc test1;
```

```
SQL> insert into test1 values (...); ..
```

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

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

```
partition (others);
```

SNO	NAME
1	us
2	england

## 3) Hash Partition:-

→ In this partition whenever we are specifying no. of partition then oracle server only internally automatically creates partitions based on Hash algorithm.

### Syntax:-

```
create table tablename (col1 datatype (size), col2 datatype (size), ....)
partition by hash (keycolumnname) partitions any number;
```

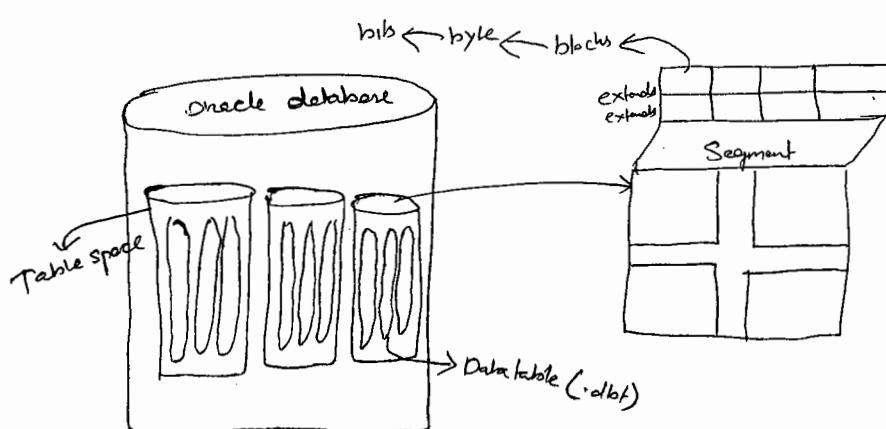
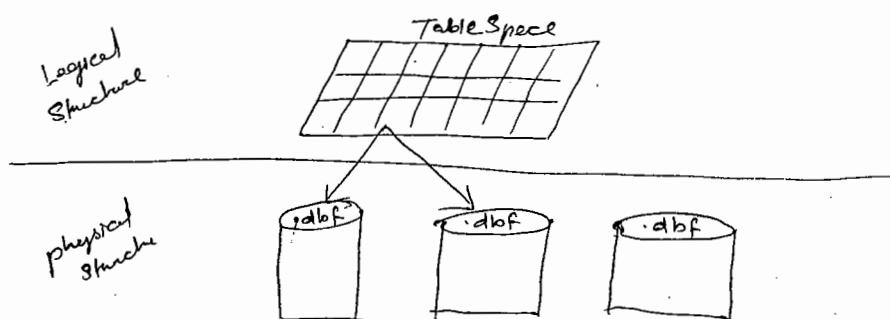
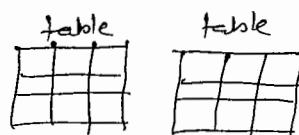
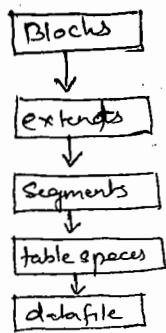
e.g:- SQL> create table test2 (sno number(10), sal number (10))  
partition by hash (sal) partitions 5;

Note:- In oracle all partitions informations stored under user\_tabs\_partitions data dictionary example:-

```
SQL> desc user_tabs_partitions;
SQL> select PARTITION_NAME from user_tabs_partitions where
table_name = 'TEST2';
```

## Oracle Storage Structure :-

→ Oracle having following storage structure



## Table Space ! -

→ Table space is a logical storage unit which contains collection of datafile physically. one datafile belongs to one datafile only. whenever we are installing oracle server then automatically 6 tablespaces are created. If you want to run Oracle minimum 2 tablespaces are required these are

System tablespace, sysaux tablespace

System tablespace stores all data dictionaries in oracle database

→ we can also create tablespace explicitly by using following syntax!

## Syntax ! -

create tablespace tablename path of the datafile;

e.g:-

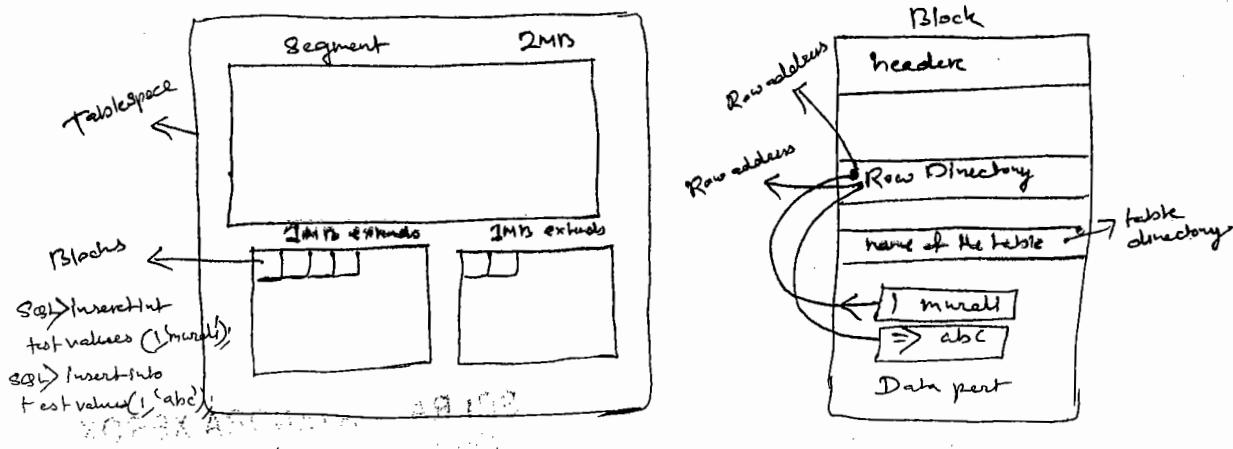
```
sql> create tablespace tab1 'c:\ORACLE\PRODUCT\10.2.0\ORADATA\ORCL\\  
SYSTEM01.DBF';
```

### Segment :-

→ Whenever we are creating a database object then automatically some storage space is allocated this storage space is also called as segment.

e.g:- In oracle whenever we are creating a table then automatically 2mb storage space is allocated there is called segment.

```
sql> create table test ( sno number(10), name varchar2(10));
```



→ Oracle having pseudo column rowid which uniquely identifying row on a table using rowid we can retrieve data very fastly from the database. Oracle rowid having 10 parts having 18 characters

→ Rowid having date of object number, relative file number, date of block number, rownumber within the block.

e.g:-

```
sql> select rowid, ename from emp where ename = 'SMITH';
```

AAAMgZAAEAAAAAAGAAA

Here

AAAMgZ → data object number

AAE → Tablespace (Relative datafile number)

AAAAAAAG → data block number

AAA → ROW number within a block

### CODD Rules

E.F. CODD introduced 12 codd rules for relational databases most of the relational databases satisfy more than 6 rules. each codd rules are

### 1) Information rule:-

→ All information in a database must be represented within tabular form i.e all data should be present to the were in tabular form.

### 2) Guaranteed access rule:-

→ Data in a table can be accessed without ambiguity i.e accessing data in a table by using table name + primary key + attribute name.

e.x! -  
SQL> select \* from emp where empno = 7788;

### 3) Systematic treatment of null value:-

#### 4) Active online catalog based relational tables.

→ In all databases data dictionary tables also stored in the format same like as relational tables

⇒ All databases having unique of SQL language

e.x! - SQL

5) view updatable rule.

6) High level insert, update, delete

e.x! - union, union all, intersect, minus

7) physical data independence

8) logical data independence

9) Integrity independence

10) Subdivision distribution

11) e.x! - distributed databases

12) there is no subversion in user interface

e.x! - through user interface we cannot establish structure of the database.

**SRI RAGHAVENDRA XEROX**  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

**SRI RAGHAVENDRA XEROX**  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

O

MURALI & JR  
ORACLE 12C

Prajithkhade

Date \_\_\_\_\_  
Page \_\_\_\_\_

## PL/SQL

### 1) PL/SQL introduction

- select... into clause
- variable , attribute (%type, %rowtype)
- conditional control stmt.
- bind variable

### 2) CURSOR

- Implicit cursor
- explicit cursor
- explicit cursor life cycle
- explicit cursor attributes
- cursor... For loop.
- parameterised cursor
- update, delete stmt used in cursor  
(without using where current of,  
for update clause)
- Implicit cursor attr.

### 3) Exceptions

- predefined exceptions
- user-defined "
- unnamed exception ,
- Error trapping Function
- raise application\_error()

### 4) Subprograms

- i) stored procedure
- ii) procedure parameters modes (in, out,
- iii) nocopy compiler hint
- iv) autonomous transaction.

SRI RAGHAVENDRA XEROX

Software Languages Material Available

Beside Bangalore Ayyagar Bakery,

Opp. CDAC, Balkampet Road,

Ameerpet, Hyderabad.

v authid = current\_user

## 2) Stored Functions

- dml stmt used in fn
- select...into clause used in fn.
- cursors in fn
- wm-concat()

## 5) Triggers

- row level triggers
- .. app'
- auto increment
- stmt level triggers
- triggers execution order
- Follow clause (11g)
- command triggers (11g)
- mutating error
- system level triggers

## 6) Packages

- global variables
- overloading procedures
- Forward declaration
- 

vvimp:

### ↓ 7) Types Used in packages

- pl/sql record
- index by table
- nested table
- varray

8) Bulk Bind

~~8~~ bulk.. collect clause

Forall stmts

indices of clause (log)

DML Errors or bulk exceptions

sql%bulk-rowcount

9) dbms-utility package

10) REF cursors

- Strong refcursors

- Weak ..

- SYS-refcursors

- Passing SYS-refcursors as in,  
out parameter to the procedure

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

11) Local Subprograms

- local procedure

- local functions

- passing refcursors as parameters  
to the local subprograms

12) utl\_file package

13) SQL\* Loader

- control files, log files, bad  
files, discard files

14) LOBS (large objects)

- 10  
Regd.
- 15) member procedures, member function
  - 16) where current of, for update clauses are used in cursors
  - 17) avoiding mutating errors by using compound triggers
  - 18) Oracle 11g features
  - 19) Oracle 12c

SRI KAVACHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

O.

PL  
SC

ble

## Introduction.

PL/SQL is a procedural lang. extension  
For SQL.

Oracle 6.0 introduced PL/SQL

Oracle 6.0 → PL/SQL 1.0

Oracle 7.0 → PL/SQL 2.0

Oracle 8.0 → PL/SQL 8.0

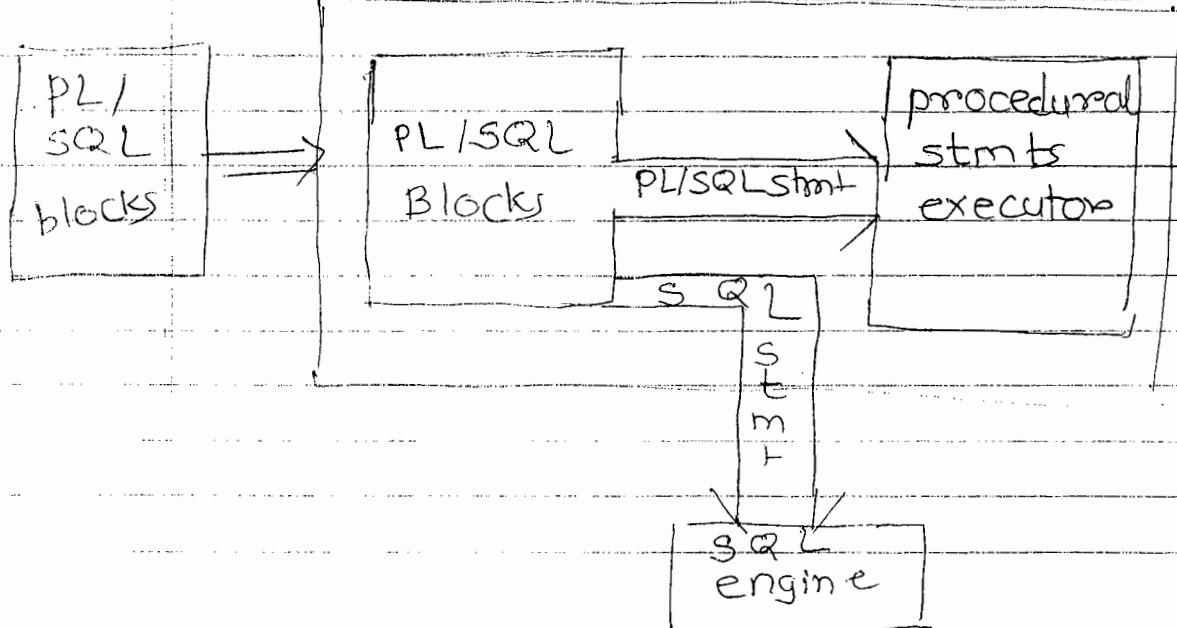
,

Oracle 12 C → PL/SQL 12C

Basically PL/SQL is a block structured programming lang. where we are submitting PL/SQL block into oracle server then all SQL stmt are executed within SQL engine & also all procedural stmt. are executed separately within procedural stmt. executor under PL/SQL engine.

## PL/SQL architecture.

Oracle server



## Block Structure

declare (optional)

→ variable declarations;

→ cursors, user defined exception

Begin (mandatory)

→ select...into clause

→ DML, DCL

→ conditional, control stmt

Exception (optional)

→ handling runtime error.

End ; (mandatory).

## Declaring a variable

Syntax

varname datatype(size);

Storing a value into variable

using assignment operator

(:=) we are storing a value  
into variable.

Syntax :

varname := value;

e.g. → declare

a {number(10)},

Object. b

```
begin
a := 50;
end;
```

### SRI RAGHAVENDRA XEROX

Software & Computer Material Available  
Beside Surya Sagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

\* display a message or variable value

Syntax:

```
dbms-output.put_line('message');
dbms-output.put_line(variable name);
-----^-----^
package           procedure name.
```

This package is used in either executable or exception section of PL/SQL block.

enviore-  
ment  
setting  
For  
SQL+

```
> set serveroutput on;
> begin
  dbms-output.put_line('welcome');
end;
/
```

Welcome

```
> declare
  a number(10);
begin
  a := 70;
  dbms-output.put_line(a);
end;
/
```

## \* Select ... into clause :-

It is used to retrieve data from table & store it into PL/SQL variables. It always ~~return~~ returns single record or single value at a time.

Syntax:

```
select col1, col2, ... into  
var1, var2  
from tablename where condition;
```

It is used in executable section of PL/SQL block.

Q. Write a PL/SQL program for user entered employee no. that display name of employee & his from emp table.

⇒ declare

```
v_ename varchar2(10);
```

```
v_sal number(16);
```

```
begin
```

```
select ename, sal
```

```
into v_ename, v_sal
```

```
From emp
```

```
where empno = &no';
```

```

dbms_output.put_line(v_ename);
  |||| v_sal);
end;
/

```

Enter value For No: 7788 .  
 SCOTT 3000

constant

→ In PL/SQL whenever we are using "not null" or '~~constraint~~ clauses' in variable declaration then we must assign the value at time of variable creation in declare section of PL/SQL block.

or syntax:-

```

varname datatype(size) not null
      := value;

```

```

varname not constraints constant datatype(size)
      := value;

```

> declare

```

a number(10) not null := 10 ;

```

```

b constant number(10) := 20 ;

```

begin

```

d-o-p-l(a);

```

```

d-o-p-l(b);

```

end;

/

SRI RAGHAVENDRA XEROX

Software & Material Available

Beside Ayyagar Bakery,

Opp. Balkampet Road,

Ameerpet, Hyderabad.

Q Write a PL/SQL program retrieve max salary from & stored sal into variable & display that sal?

⇒ declare

```
v_sal number(10);  
begin  
select max(sal) into v_sal  
from emp;  
dbms_output.put_line(v_sal);  
end;  
/
```

\* (not using ^ since max return single value & select ... into ... clause required one record at a time)

⇒ declare

```
a number(10);  
b number(10);  
c number(10);
```

begin

a := 70;

b := 30;

c := greatest(a, b);

$\rightarrow [c := \max(a, b)]$   
    Wrong ↑

dbms\_output.put\_line(a);

end;

/

70.

Note →

In PL/SQL expression we are not allowed to use group function, decode() conversion function. But we are allowed number function, character function, date function, date conversion function in PL/SQL expression.

e.g. :

> declare

a varchar2;

begin

a := upper ('xyz');

dbms\_output.put\_line (a);

end;

/

XYZ

SRI RAGHAVENDRA XEROX

Software Languages Material Available

Beside Bangalore Ayyagar Bakery,

Opp. CDAC, Balkampet Road,

Ameerpet, Hyderabad.

### Variable attributes :-

are used in place of datatypes in variable declaration whenever we are using ~~as~~ variable attribute then oracle server automatically allocates memory for the variable as same as corresponding column datatype in a table. This variable attr. are also called as anchor notation.

PL/SQL having 2 types of variable attributes. These are

- 1) column level attr.
- 2) row level attr.

### 1) column level attr.

In this method we are defining attr. For individual column. Column level attr. are represented by using (% type). When we are using column level attr. then oracle server automatically allocates same memory for variables based on corresponding column datatype in a table.

#### syntax:

variable name tablename. column-  
name % type :

↓  
keyword.

e.g.

> 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;

ble

```
dbms_output.put_line('V-enameli' ||  

    'V-sal' || '|| V_hiredate );  

end ;  

/
```

Enter value for no: 7902  
 FORD 3000

**SRI RAGHAVENDRA XEROX**  
 Software Languages Material Available  
 Beside Bangalore Ayyagar Bakery,  
 Opp. CDAC, Balkampet Road,  
 Ameerpet, Hyderabad.

### \* Row Level Attr. :-

In this method a single variable can represent all different datatype in a row within a table. Row level attr. are represented by using "%rowtype". This variable is also called record type variable. It is also same as structures in C lang.

### Syntax :-

variablename tablename%rowtype;  
 e.g.

```
>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.hired-
ate);
end;
/

```

OR

```

declare
i emp%rowtype;
begin
select * into i from emp
where empno = &no;
dbms_output.put_line
(i.ename || ' ' || i.sal || ' '
i.hiredate || ' ' || i.deptno);
end;
/

```

SRI RAGHAVENDRA XEROX  
 Software Languages Material Available  
 Beside Bangalore Ayyagar Bakery,  
 Opp. CDAC, Balkampet Road,  
 Ameerpet, Hyderabad.

declare

i emp%rowtype , i.job

	empno	ename	mgr	job	hiredate	... dept
i	7902	For D				

i.empno. i.mgr

## PL/SQL Datatypes & Variables

- 1) It supports all SQL datatype (scalar datatype) + boolean datatype.
- 2) Composite datatypes
- 3) Ref Objects
- 4) LOBs (Large Objects → clob, blob, bfile)
- 5) bind variable or Non-PL/SQL variable

### \* Bind Variable :-

Bind Variable is a session variable created at host environment that why this variable are also called as host variable.

These variable are used in SQL, PL/SQL, Dynamic SQL that why this variable are also called as non-PL/SQL variable.

We can also use this variable in PL/SQL when a sub-program having out (or) inout in out parameters.

Step 1: Creating bind variable

→ variable variablename datatype;

Step 2: Using bind Variable

Syntax :-

: variablename

Step 3: Display value from bind variable.

sql> print bindvariablename;

e.g.

```
> variable g number;  
> declare  
    a number(10) :=1000;  
begin  
    :g := a/2;  
end;  
/
```

PL/SQL completed

> print g;

G

500

Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

## Conditional Statement

- 1) if
- 2) if - else
- 3) elsif

1) if :-

Syntax if condition then  
stmts;  
end if ;

## 2) if else :-

- Syntax :

```
if condition then  
stmts;  
else  
stmts;  
end if
```

## 3) elsif :-

To check more no. of condition  
then we are using elsif.

Syntax :

```
if condition1 then  
stmts;  
elsif condition2 then  
stmts;  
elsif condition3 then  
stmts;  
else  
stmts;  
end if;
```

e.g.

```
>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');  
  
elsif v_deptno = 20 then  
    dbms_output.put_line ('twenty');  
  
elsif v_deptno = 30 then  
    dbms_output.put_line ('thirty');  
  
else  
    dbms_output.put_line ('others');  
  
end if;  
end;  
/
```

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

Enter value for deptno : 40  
Others

sql> /  
Enter : 90  
error.

\* note:- When PL/SQL block contains  
"select ... into clause" & also  
through that if requested data  
not available in a table then  
oracle server returns an  
error error:

Ora-1403: no data Found.

note2:-

Whenever a PL/SQL block having pure DML stmts and also through these stmts if requested data not available in table then Oracle server does not return any errors. For handling this type of block then we are using "implicit cursor" attributes

e.g.

```
>begin  
  delete from dept emp  
  where ename = 'welcome'  
end;  
1
```

PL/SQL procedure successfully completed.

note3:-

In PL/SQL block whenever select into clause try to return multiple records or try to return multiple values in a column at a time then oracle server returns an error.

Ora-1422:- exact fetch returns more than requested no. of rows.

e.g.

declare

i emp%rowtype;

begin

select \* into i from cmp

where deptno = &deptno;

dbms\_output.put\_line (i.ename ||

' ' || i.sal || ' ' || i.deptno);

end;

/

Enter value for deptno : 10

error: ora-1422

—oxo—

## Control Statements (or) loops

1) Simple loop

2) While loop

3) For Loop

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

1) Simple loop

This loop is also called as infinite loop. Here body of loop stmts are executed repeatedly.

Syntax:

```
loop
  stmts;
end loop;
```

>begin  
loop

dbms-output.put-line ('Welcome');  
end loop;

end;

/

\* To exit From infinite loop.

\* Method 1:-

Syntax :- exit when true condition;

e.g.

> declare

n number(10) := 1

begin

loop

dbms-output.put\_line(n);

exit when n >= 10;

n := n + 1;

end loop;

end;

/

**SRI RAGHAVENDRA XEROX**

Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

Method 2: Using IF

Syntax :- if condition then  
exit;  
endif;

e.g.

> declare

n number(10) := 1;

begin

loop

dbms-output.put\_line(n);

if n >= 10 then

exit;

end if;

n :: n + 1

2) while loop :-

Here body of loop statements  
are executed repeatedly until  
condition is false.

Syntax :

while (condition)

loop

stmts;

end loop;

declare

n number(10) := 1;

begin

while n <= 10

loop

dbms-output.put\_line(n);

n := n + 1;

end loop;

end;

/

### 3) For loop :-

syntax

for indexvariablename in  
lowerbound .. upperbound

loop

stmts;

end loop;

e.g.

> declare

n number(10);

begin

for n in 1 .. 10

loop

dbms\_output.put\_line(n);

end loop;

end;

/

=> 1. 2 3 . . . 4

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

> declare

n number(10);

begin

for n in reverse 1 .. 10

loop

dbms\_output.put\_line(n);

end loop;

end;

=> 10 9 8 . . . 1

In For loop index variable internally behaves like a integer variable that why when we are using For loop not require to declare index variable.

e.g.

```
begin
  for n in 1 .. 10
    loop
      dbms_output.put_line(n);
    end loop;
  end;
```

→ PLISQL having 2 types of block

anonymous

- these block

does not have name

- not allowed to store permanently in database

- not allowed to call in client appln

ex. declare  
begin  
end;

Named

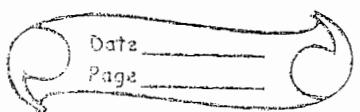
- block

- have name

- automatically permanently stored in database

- allowed to call in client appln

ex. procedure,  
function ...



## 2. CURSOR

Cursor is a private SQL memory area which is used to process multiple records & also ~~is~~ this ~~an~~ record by record process.

All database system has 2 types of static cursor.

- i) Implicit Cursor
- ii) Explicit Cursor.

Explicit Cursor → to process multiple record  
→ Record by record process.

For SQL stmts returns multiple record is called explicit cursor & also this an ~~is~~ record by record by process. Explicit cursor memory area is also called as "active set area".

### Explicit Cursor Life Cycle

Explicit cursor having 4 steps

- 1) declare
- 2) open
- 3) Fetch
- 4) close

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

## declare

In declare section of PL/SQL block we are defining cursor memory area by using following syntax:-

```
cursor cursorname is  
select * from tablename  
where condition;
```

e.g.

```
cursor c1 is  
select * from emp  
where job = 'CLERK';
```

## Open

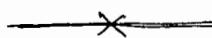
Whenever we are opening then only oracle server retrieve data from table into cursor memory. Because in all databases whenever we are opening cursor then only cursor select stmt are executed.

syntax:

```
open cursorname;
```

This statement is used in executable section of PL/SQL block.

Whenever we are opening a cursor internally cursor pointer always points to first record within cursor memory area



Fetch :- (Fetching data from cursor)

Using Fetch statement we are fetching data from cursor memory area into PL/SQL variable.

Syntax :-

Fetch cursorname into var1, var2, ...



close :-

Whenever we are closing the cursor all the resources allocated from cursor memory area are automatically released.

close cursorname ;

> 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;  
d0.PL ('v\_ename' || ' ' || v-sal);

Fetch c1 into v\_ename, v-sal;  
dbms\_output.put\_line ('my second  
employee salary is' || ' ' || v-sal);

Fetch c1 into v\_ename, v-sal;  
dbms\_output.put\_line (v\_ename ||  
' ' || ' having low salary');

close c1;  
end;

/  
o/p

SMITH 900

my second employee salary is 600.

WARD having low salary.

### Explicit Cursor Attribute

Every explicit cursor having  
4 attr. these are

- 1) %notFound
- 2) %Found
- 3) %isopen
- 4) %rowCount

al;  
al);  
When we are using these attribute  
then we must specify cursorname  
along with these attr.

il;  
ond  
sal)  
syntax:

cursorname%.attributename ;

l;  
1  
j true or False, whereas %.rowcount  
returns number datatype.

### 1) %.not %.%notfound :

This attr. always returns  
boolean value either true or False.  
This attr. returns true when fetch  
statement does not return any row,  
whereas this attr. ~~ret~~ returns  
False when fetch stmt. returns  
at least one row.

syntax: cursorname %.notfound ;

Q Write a PL/SQL cursor program to  
display all employee name & their  
salaries from emp table by using  
%.notfound

> declare

cc cursor c1 is select \* from emp;  
v\_ename varchar2(10);  
v\_sal number (10);

Date \_\_\_\_\_  
Page \_\_\_\_\_

```
begin
open c1;
loop
Fetch c1 into v_ename, v_sal;
dbms_output.put_line ('v_ename'
|| ' ' || v_sal);
end loop;
close c1;
end;
```

Q. Write a PL/SQL cursor program to display 1<sup>st</sup> 5 highest sal employee from emp by using row count

⇒ declare  
cursor c1 is select \* from emp order by sal desc;  
v\_ename varchar2(10);  
v\_sal number (10);  
begin  
open c1;  
loop  
Fetch c1 into v\_ename, v\_sal;  
dbms\_output.put\_line ('v\_ename'
|| ' ' || v\_sal);
exit when c1%rowcount = 5;
end loop;
end;

O/p KING 5500

FORD 3100

:

Q. Write a PL/SQL cursor program to display even no. of records from emp by using rowcount.

⇒

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  
exit when C1%not found;

if (C1%rowcount / 2 = 0)

if ( mod (C1%rowcount, 2) = 0 )

then

dbms\_output.put\_line (' .. . );

end if;

end loop; close C1

end;

Date \_\_\_\_\_  
Page \_\_\_\_\_

Whenever we are creating a cursor then oracle server automatically allocate 4 memory allocation alongwith cursor memory area. These memory location are identified through cursor attr. These memory location behaves like a variable i.e. these<sup>variable</sup> also store only one value at a time.

sql> declare

a number(10);

b boolean;

begin

a := 10;

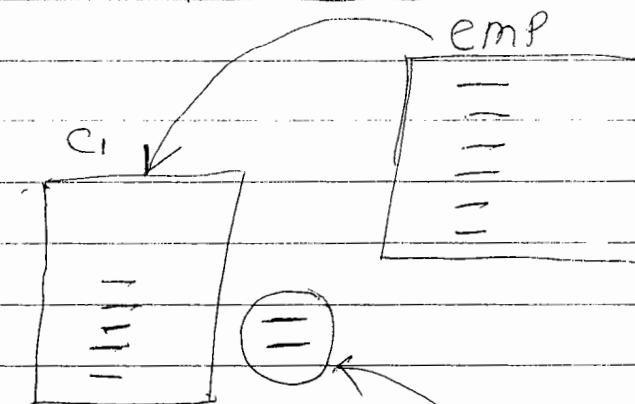
b := true

end;

a  
10

b  
true

emp



c1.notFound

True

c1.found

true

c1%isopen

true

c1.rowcount

2

\* 1. rowcount :

to -

This attribute always return no number datatype i.e. it returns number of record's number fetched from ~~cursor~~ cursor.

Syntax:- cursourname%>rowcount;

declare

cursor c1 is select ename, sal  
from emp;

v-ename varchar2(10);

v-sal number(10);

begin

Open c1;

Fetch c1 into v-ename, v-sal;

dbms\_output( );

dbms\_output( );

dbms\_output.put\_line('number of  
records number fetched is:'||' '||  
'1'||rowcount);

close c1;

end;

/

SMITH

AUEN

**SRI RAGHAVENDRA XEROX**  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

2

Q. Write a PL/SQL cursor program which display 5<sup>th</sup> record from emp by using /rowcount

⇒

```
declare
cursor c1 is select ename,
sal from emp;
v_ename
v_sal
begin
open c1;
Fetch c1 into v_ename, v_sal;
exit when c1%not found;
if (c1%rowcount = 5)
then dbms_output.put_line
(v_ename || ' ' || v_sal);
end if;
end;
1
```

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

MARTIN.

Note :- Using cursors we can also transfer data from oracle table into another oracle table and also transfer data into arrays, OS File.

Q. Write a PL/SQL cursor program which transfer who are getting more than 2000 sal @ From emp to another table.

⇒ > create table target (sno number(10), ename varchar(10), sal number(10));

> 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;

SRI RAGHAVENDRA XEROX  
 Software Languages Material Available  
 Beside Bangalore Ayyagar Bakery,  
 Opp. CDAC, Balkampet Road,  
 Ameerpet, Hyderabad.

> select \* from target

sno	ename	sal
1	JONES	3075
2	CLARKE	2850

Q. Write a PL/SQL cursor program to display total salary without using sum FN.

```

⇒ declare
    cursor c1 is select * from
    emp;
    n number(10) := 0 ;
    i emp%rowtype;
begin
    open c1; loop
        fetch c1 into i;
        exit when c1%notfound;
        n := n + nvl(i.sal, 0);
    end loop;
    dbms_output.put_line(n);
    close c1;
end;
    
```

Note :- Whenever resources column having null value & also when we are try to calculate summarize data then we must use nvl Function.

e.g.

$n := n + \text{nvl}(i.\text{sal}, 0);$

## \* Eliminating explicit cursor life cycle (or) Cursor for loop

Whenever we are using cursor for loop then no need to use open, Fetch, close statements.

Whenever we are using cursor for loop then only oracle server internally automatically open cursor & fetch data from cursor & close the cursor. This method is also called as short-cut method of cursor.

for indexvariablename in cursorname  
loop

    stmts;  
end loop;

$\downarrow$   
%.rowtype

Cursor for loop is used in executable section of PL/SQL block.

\* Note:- In cursor for indexvariable internally behaves like a record type variable (%.rowtype)

Q. Write PL/SQL program by using cursor for loop to display all emphname & sal from emp.

$\Rightarrow$  declare

cursor c1 is select \* from emp;



```
begin
for i in c1
loop
dbms_output.put_line(i.ename || ' '
|| i.sal);
end loop;
end;
```

Note:- We can also eliminate declare section of cursor by using for loop , in this we must specify select stmt within parenthesis in place of cursor name within cursor for loop.

Syntax:-

For indexvariable in (select  
stmt)

```
loop
stmt;
end loop;
```

e.g.

```
> begin
for i in (select * from emp)
loop
dbms_output.put_line(i.ename || ' '
|| i.sal);
end loop;
end;
```

Q. Write PL/SQL to display 5th record from emp by using cursor for loop.

⇒ declare

cursor c1 is select \* from emp;

begin

for i in c1

loop

if &c1%rowcount = 5

then

dbms\_output.put\_line ('ename' ||  
i.sal);

end if;

end loop;

end,

SRI RAGHAVENDRA XEROX

Software Languages Material Available

Beside Bangalore Ayyagar Bakery,

Opp. CDAC, Balkampet Road,

Ameerpet, Hyderabad.

Q. Write a PL/SQL cursor program to display total salary from emp.

⇒ declare

cursor c1 is select \* from emp;

n number(10) := 0;

begin

for i in c1

loop

n := n + nvl(i.sal, 0);

end loop;

dbms\_output.put\_line ('total salary'

||' '|| n);

end;

/

Date \_\_\_\_\_  
Page \_\_\_\_\_ C

e.g. declare  
cursor c1 is select \* from emp;  
begin  
for i in c1  
loop  
if < i.sal > 2000 then  
dbms\_output.put\_line(  
);  
else  
db  
:  
:  
end;

—oxo—

### \*%Found :

This attr. returns boolean value either true or False.

This attribute return true when Fetch stmt returns at least one record otherwise it returns False.

syntax: cursorname%.Found

declare

cursor c1 is select \* from emp where ename = '&ename';  
i emp%rowtype;

begin

open c1;

Fetch c1 into i;

emp, if C1%Found then  
dbms\_output.put\_line ('u r emp  
does not exist || ' || i.ename);  
else if C1%NotFound then,  
db\_0/p.pl ('does not exist');  
end if;  
close C1;  
end \$\$;  
/

Q. Write a PL/SQL cursor program  
which display all ename & sal from  
emp by using %Found attr.

⇒ declare

cursor C1 is select \* from emp;  
i% emp%rowtype;  
begin  
open C1;  
Fetch C1 into i;  
while (C1%Found)  
loop  
dbms\_output.put\_line( );  
Fetch C1 into i;

\*) %!open

This attr. also return boolean either true or false. This attr. is used to test whether cursor is open or not.

Syntax :- cursorname %!isopen

3)

3)

declare

cursor c1 is select \* from emp;  
| emp%rowtype

begin

if not c1%isopen then  
open c1;

end if;

loop

Fetch c1 into i ;

& exit when c1%notfound;

dbms ( . . . . ) ;

end loop;

end;

1

1) %!found → True : Fetch stmt returns at least one row

→ False : Fetch stmt does not return any row.

2) %notfound → True : Fetch stmt does not return any row

→ False : Fetch stmt returns at least one row.

4)

3) ~~EOF~~

3) `%.isopen` → True : If cursor is already opened

→ False : If cursor is not opened.

4) `%.rowcount` → number : It counts number of records number fetched from cursor.

— O X O —

## Parameterized Cursor

In oracle we can also pass parameters to the cursor same like a subprogram , 'in' parameter . These type of cursor are also called as parameterized cursor. In parameterized cursor, we are specifying formal parameter when we are declaring a cursor & also we are specifying actual parameter when we are opening a cursor.

In oracle whenever we are passing parameters to cursors, procedures, Function then we are not allowed to use datatype size in formal parameter declaration.

Syntax:

cursor

Syntax:

Formal



cursor cursorname (parameter  
datatype) is select columnname  
or \* from tablename;

open cursorname (actual parameter);

e.g. > declare

```
cursor c1( is p_deptno number)
is select * from emp;
    i emp%rowtype;
```

begin

```
open c1(10);
```

loop

```
Fetch c1 into i;
```

```
exit when c1%notfound;
```

```
dbms_output.put_line(i.ename
||' '|| i. deptno);
```

end loop;

```
close c1;
```

end;

/

- Q. Write a PL/SQL parameterized cursor for passing job as parameter whose job as 'CLERK'  
or analyst from emp. Display the output statically by following format.

Employee Working as Clerk

SMITH

ALLEN

JAMES

MILLER

Employee Working as Analyst.

declare

cursor c1(p\_job varchar2) is  
select \* From emp;  
i emp%.rowtype;

begin

open c1('CLERK');

dbms\_output.put\_line ('Employee  
Working as Clerks'); loop

Fetch c1 into i;

exit when c1%notfound;

dbms\_output.put\_line (i.ename);

end loop;

close c1;

Open c1('Analyst');

db... ('Employee working as  
Analyst');

loop

Fetch c1 into i;

exit when c1%notfound;

db... (i.ename);

end loop;

close c1;

end;

/

Before we are reopening a cursor we must close cursor properly otherwise oracle server returns a error

ORA - 6511 : cursor already is open.

Note :- Whenever we are not opening a cursor but we are try perform oprn on cursor the oracle server returns a error ora- 1001 : @ invalid cursor.

Q Write a PL/SQL parameterized cursor for passing deptno qis a parameter then display passed deptno details from emp by using cursor for loop.

⇒ declare  
cursor c1(pdeptno number) is  
select \* from emp where  
deptno = p\_deptno ;  
begin  
for i in c1(10)  
loop  
dbms\_output.put\_line(i.enamell...);  
end loop;  
end;

Not

e.g. declare

cursor c1 (p\_job varchar2) is select \* from emp where job = p-job;  
begin

dbms\_output.put\_line ('Employee working  
as CLERKS');

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 oracle whenever we are defining multiple cursor & if we want to pass data from one cursor to another cursor. Then receiving cursor must be parameterized cursor.

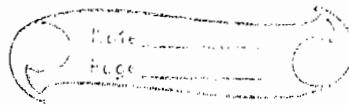
SRI RAGHAVENDRA XEROX

Software Languages Material Available

Beside Bangalore Ayyagar Bakery,

Opp. CDAC, Balkampet Road,

Ameerpet, Hyderabad.



Q Write a PL/SQL program retrieve all deptno into a static cursor & pass these deptno from this cursor another parametrized cursor which returns employee details from emp based on passed dept no.

⇒ declare  
cursor c1 is select \* from dept;  
cursor c2(p-deptno number) is  
select \* from emp where  
deptno = p-deptno;  
begin  
for i in c1  
loop  
dbms\_output.put\_line('my dept table  
deptno is '|| i.deptno);  
for j in c2(i.deptno)  
loop  
dbms\_output.put\_line(j.empname||  
' '|| j.deptno);  
end loop;  
end loop;  
end;

/

my dept table deptno is 10

CLARK 10

MILLER 10

~ ~

Functions or Expression used in cursor  
select stmt.

In oracle we can also use functions or expressions within cursor select statement. In this case we must ~~use~~ create alias name for those functions or expression & also we must declare cursor record ~~as type~~ variable within declare section of PL/SQL block.

Syntax:-

varname cursorname%rowtype;

Q. Write a PL/SQL cursor program to display total sal from emp table by using sum function.

declare

cursor c1 is select sum(sal) a  
from emp;

i c1%rowtype;

begin

open c1;

fetch c1 into i;

~~exit~~ dbms-output.put\_line(i.a);

close c1;

end;

/

Q Write a PL/SQL parameterized cursor program for passing deptno as parameter then display, no. of employee, max, min, total sal from dept no. by using emp. in Following Format.

- Enter value for deptno:

2011

No. of employee :

Max sal :

Min : , :

Total : , :

No:

declare  
cursor c1(p\_deptno number) is  
select count(emplno) a,  
max(sal) b,  
min(sal) c,  
sum(sal) d from emp  
where dept where deptno = p\_deptno;

i c1%rowtype;

begin

open c1(&deptno);

Fetch c1 into i;

dbms\_output.put\_line('Number of  
Employee are : ' || ' || i.a');

----- ; . i.b );

----- ; . i.c );

----- ; . i.d );

close c1;

cursor -  
dbms\_output.put\_line ('Minimum salary  
is: ' || ' ' || i.b);  
dbms\_output.put\_line ('Max salary  
is: ' || ' ' || i.c);  
dbms\_output.put\_line ('Total salary is:  
' || ' ' || i.d);

20/10

Note :- In parameterized cursor we can also pass default values by using default or assignment operator (:=)

Syntax:

Parametername datatype default/[:=]  
default value

e.g.

> declare

cursor c1(p\_deptno number default  
30) 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;

/

\* Update, delete statement are used in cursor (without using current of, for update clauses);

Q. Write a PL/SQL program to modify salary of employee by following condition

- 1) if job = 'CLERK' then increment  
~~sal + 100~~ sal → 100
- 2) if job = 'SALESMAN' then decrement  
sal → 200

> 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

~~sal +~~

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;

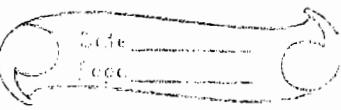
### \* implicit cursors attributes

For sql statements returns single records is called implicit cursor.

Implicit cursor memory area is called as context area.

In oracle whenever PL/SQL block having select .. into clause (or) pure DML stmts then oracle server internally automatically allocates memory area. These memory area is also called as 'SQL area' or "context area" or "implicit cursor".

These memory area returns records when PL/SQL block having select .. into clause. These memory area also returns also returns multiple record when PL/SQL block contains pure DML statement. These multiple records are process at a time by a sql engine that's way developer does not control individual records in implicit cursor.



e.g.

> declare

```
v_ename varchar2(10);  
v_sal number(10);  
begin  
select ename, sal into v_ename,  
v_sal from emp where  
empno = &no;  
dbms_output.put_line(v_ename  
||' '|| v_sal);  
end;  
/
```

Along with implicit cursor area four memory location area are automatically created. These memory locations behaves like a variable.

These memory locations are identify through implicit cursor.

These attributes are

- 1) sql%NotFound
- 2) sql%Found
- 3) sql%IsOpen
- 4) sql%RowCount

Here always sql%IsOpen returns False & also sql%Found, sql%NotFound attribute returns either true or false whereas sql%RowCount attribute returns number datatype.

```
>begin  
    delete from emp where ename  
    = 'welcome';  
end;
```

/

PL/SQL procedure successfully completed.

Using implicit cursor attributes

```
>begin  
    delete from emp where ename  
    = 'welcome';  
    if sql%found then  
        dbms_output.put_line('u r record  
exists and deleted');  
    end if;  
    if sql%notfound then  
        dbms_output.put_line('u r record  
does not exist');  
    end if;  
end;
```

/

u r record does not exist

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

Date \_\_\_\_\_  
Page \_\_\_\_\_

```
>begin
update emp set sal = sal+100
whose job = 'CLERK';
dbms-output.put_line('affected
number of clerks are : '||sql%rowcount);
end;
/
```

affected number of clerk are

## Exception

Exception is an error occurred during runtime whenever runtime error is occurred use an appropriate exception name in exception handles under exception section with in PL/SQL blocks.

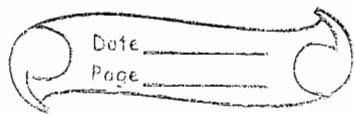
Oracle having three types of Exception.

- 1) Predefined

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

**SRI RAGHAVENDRA XEROX**  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

27/10



## 7) invalid\_number, value\_error :-

In oracle when we try to convert string type to number type or date string into date type then oracle server returns 2 type of error. These are invalid\_number, value\_error.

### invalid\_number:-

When PL/SQL block having SQL stmts. & also those stmt try to convert string type to number type. or date string into date type, then oracle server returns error Ora-1722 : invalid number.

For handling this error oracle provided invalid\_number exception name.

e.g.

begin

insert into emp(empno, ename, sal)  
values (1, 'murali', 'abc');

exception

when invalid\_number then  
dbms\_output.put\_line ('insert proper  
data only');

end;

e.g.

declare

v\_deptno varchar2(10) := '&deptno';

v\_dname varchar2(10) := '&dname';

v\_loc varchar2(10) := '&loc';

begin

insert into dept values

(v\_deptno, v\_dname, v\_loc);

exception

when invalid number then

db\_0.p.l ('enter proper data only');

end;

ex:

### 'value\_error' :

When PL/SQL block having procedural stmts and also those stmt try to convert string type to number type then oracle server returns a error.

Ora-6502: numeric or value error : character to number conversion error.

For handling this error we are using value\_error exception name.

e.g. declare

z number(10) ;

begin

z := '&x' + '&y' ;

dbms\_output.put\_line(z);

exception

when value\_error then

dbms\_output.put\_line('enter proper  
data only');

end;

1 Enter proper data only

x: a

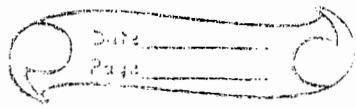
y: b

enter proper data only.

When we try to store more data than the datatype size specified in varchar2 datatype variable declaration then also oracle server returns a error

ora-6502 : numeric or value error:  
character string buffer too small.

For handling this error also we are using value\_error exception name.



When we try to store more data than datatype size specified in number datatype then also oracle server returns an error.

ora-6502 : numeric or value error : number precision too large.

For handling this error we can use value\_error exception name.

e.g.

declare

z varchar2 (3) ;

begin

z := 'abcd';

dbms\_output.put\_line (z);

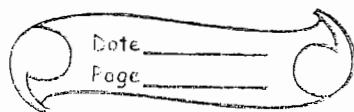
exception

when value\_error, then

dbms\_output.put\_line ('invalid string length');

end;

/



(\*) e.g.

declare

z varchar2(3) := 'abcd';

begin

dbms\_output.put\_line ('invalid string length'); (z);

exception

when value\_error then

dbms\_output.put\_line ('invalid string length');

end;

/

ora-6502 : numeric or value error.

### \* Exception Propagation

In PL/SQL exception also raised in declare section, executable section, exception section. Whenever exception raised in executable section those exception are handled either in inner blocks or in outer block whereas when exception are raised in declare or in exception section those exceptions are handled by using outer blocks only. This is called PL/SQL exception propagation.

e.g.

```
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');
end;
/
O/p => , , handled through
outer block.
```

— o —

## User Defined Exception

In oracle we can also create our own exception name & also raise that exception whenever necessary in PL/SQL block. These type of exception are called user defined.

## Handling User defined Exception

Steps

- 1) declare
- 2) raise
- 3) handling exception.

1) declare :-

In declare section of PL/SQL block we are creating our own exception name by using exception pre-defined type.

Syntax :-

userdefinename exception ;

e.g.

> declare  
a exception;

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

2) raise :-

Using raise statement we can raise user defined exception either in executable section or in exception section of PL/SQL block.

Syntax :-

raise userdefinenameexceptionname ;

### 3) handling exception

We are handling user defined exception as same as predefined exception by using exception handler under exception section of PL/SQL block.

Syntax :-

```
when userdefineexception1 then  
stmts;
```

```
when userdefineexception2 then  
stmts;
```

\* Note :-

In all databases if you want to raise messages based on client business rule then we must use userdefine exception.

Q. Write a PL/SQL program raising userdefined exception today.

```
⇒ declare  
      a exception;  
begin  
if to_char(sysdate,'DAY')  
= 'WEDNESDAY'
```

then

```
raise a;
```

```
end if;
```

exception

when a then

```
dbms_output.put_line ('my exception  
raised on wednesday');  
end;
```

SQL  
handler  
red  
d  
SQL

e.g.

declare

a exception;

v-sal number(10);

begin

select sal into v-sal from emp

where empno = 7902;

if v-sal > 2000 then

raise a

else

update emp set sal = sal + 100

where empno = 7902;

end if

using

exception

when a then

dbms\_output.put\_line ('salary already  
high');

end;

SRI RAGHAVENDRA XEROX

Software Languages Material Available

Beside Bangalore Ayyagar Bakery,

Opp. CDAC, Balkampet Road,

Ameerpet, Hyderabad.



Q. Write a PL/SQL program by using Following test table whenever user entered no. is < 1 or > 10 then raise an userdefine exception and also whenever user entered no. is in table then display corresponding empname from test table.

⇒ `create table test as select rownum empno, ename from emp where rownum >= 10;`

`> select * from test;`

Note

`> declare  
v-empno := &empno; v-ename var  
char(10);  
a exception;  
begin  
select en  
if v-empno > 10 OR  
v-empno < 1 then  
raise a;  
else  
select ename into v-ename from test  
From test  
where empno = v- v-empno;  
dbms_output.put_line(v-ename);  
end if;  
exception  
when a then`

using  
'er  
>10  
option  
tered

```
dbms_output.put_line ('empno not  
in range');
```

~~end~~ ;

```
when value_error then  
dbms_output.put_line ('enter proper  
data only');
```

end;

O/P => Enter value for empno: 5  
MARTIN

Note :- Whenever we are using exception procedure when condition is true then raise those procedures & also stop the execution of program. That's why control does not go to remaining ~~con~~ executable whereas when we <sup>?</sup> when condition is true then display message & also control goes to remaining executable section

Testing Exception Propagation through user define exception.

~~Exception raised in executable section :-~~

Method 1 (handled using inner block)

```
> declare
  a exception;
begin
raise a;
exception
when a then
dbms_output.put_line('handled
using inner block');
end;
/
```

Method 2 (handled using outer block)

```
> declare
  a exception;
begin
begin
raise a;
end;
exception
when a then
dbms_output.put_line('handled
```

ough  
using outer blocks' );  
end;

ble /

Exception raised in exception section

In PL/SQL when exception raised in exception section those exception must be handled by using outer block only.

declare

z1 exception;

z2 exception;

begin

begin

raise z1;

exception

when z1 then

dbms\_output.put\_line ('z1 handled')

);

end;

exception

when z2 then

dbms\_output.put\_line ('z2 handled')

end;



e.g. > declare

cursor c1 is select \* from  
emp where deptno = &deptno;  
i 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 a into i;  
end loop;  
end if;  
exception  
when a then  
dbms\_output.put\_line ('u r  
requested deptno does not exist  
in emp');  
end;

& deptno = 90

u r requested ..... emp

Note:- In oracle we can also raise predefined exception explicitly by using raise stmts.

Syntax :-

raise pre-defined exception name;

declare

cursor c1 is select \* from emp  
where deptno = &deptno;  
i emp%rowtype;

begin

fetch c1 into i;

if c1%rowcount = 0 then

raise no\_data\_found ;

else

while c1%found

loop

dbms\_output.put\_line (i.ename||' '  
i.deptno);

Fetch c1 into i;

end loop;

end if;

exception

when no\_data\_found then

### 3) UnNamed Exception

In oracle if you want to handle other than oracle-20 pre-define exception name error then we are using un-named exception.

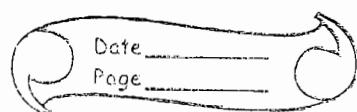
In this method we are creating our exception name by using exception pre-define type & associates this exception with appropriate error number by using EXCEPTION\_INIT( ) F<sup>n</sup>.

This F<sup>n</sup> accepts 2 parameters  
Syntax :

```
pragma Exception_INIT( user-defined exception, error number );
```

Here 'pragma' is a compiler directive. Whenever we are using this pragma @ oracle server associates error no. with exception at compile time.

This EXCEPTION\_INIT( ) F<sup>n</sup> declared in declare section of PL/SQL block.



declare

> begin

insert into emp(empno, ename)  
values (null, 'murali');  
end;

/

[ora-01400] cannot insert NULL into  
empno

Solution:

>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('cannot  
insert null values in not null  
column');

end;

/

F7

C

e.g.

> begin  
  delete from dept where deptno=10;  
end;

error : **[ora - 02292]**

> 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;

30/10

— oxo —

Q. Write a PL/SQL program by using exception\_init function handles **[ora - 2291]** based on emp, dept tables.

> declare  
  a exception;  
  pragma exception\_init (a, -2291);  
begin

```

insert into emp (empno, deptno)
values (1, 50);
exception
when a then
exception dbms_output.put_line (''
cannot insert other than primary
key values');
end;

```

SRI RAGHAVENDRA XEROX

Software Languages Material Available  
 Beside Bangalore Ayyagar Bakery,  
 Opp. CDAC, Balkampet Road,  
 Ameerpet, Hyderabad.

2)

30/10

\* raise application\_error(); :-

raise application\_error(); is a predefined exception procedure available in dbms\_standard package.

If you want to display user define exception messages in more descriptive form then only we are using this procedure i.e. if you want to display user define exception messages as same as oracle error display format then we must use raise-application-error.

syntax:

raise-application-error (error no.,  
 msg )  
 ↓  
 upto 512  
 characters

-20000 to -20999

This procedure is used in either in executable section or in exception section of PL/SQL block.

e.g.

declare

a exception;

begin

if to\_char(sysdate, 'DY') = 'FRI';

then

raise a;

end if;

exception

when a then

raise\_application\_error(-20456, '

my exception raised on Friday');

end;

O/P

ORA-20456: my exception raised  
on raised.

\* DIFF betn dbms\_output.put\_line  
procedure , raise\_application\_error()

put\_line() procedure is a normal  
procedure that's why whenever  
condition is true then it will  
display msg & also control goes

to remaining program whereas raise application\_error is an exception procedure that's why whenever condition is true it raise a message and also it stops the execution of the PL/SQL block. And also whenever cond' is true it's stop invalid data entry into our table whereas put\_line does not stop invalid data entry that's why raise application\_error() procedure always used in triggers.

e.g.

```
> begin  
  if to_char(sysdate, 'DY') = 'FRI'  
  then  
    raise_application_error (-20123,  
    'my exception raised on Friday');  
  end if;  
  dbms_output.put_line ('control does  
  not go to next line');  
end;
```

O/P :

ora-20123: my exception raised  
on Friday.

Date \_\_\_\_\_  
Page \_\_\_\_\_

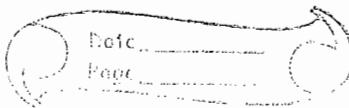
```
> begin
  if to_char(sysdate, 'DY') = 'FRI'
  then
    dbms_output.put_line('my
      exception raised on Friday');
    end if;
    dbms_output.put_line('control
      goes to next line');
    end;
  /
o/p
my exception ...
control goes .... line.
```

### \* Error trapping Functions

Oracle having 2 error trapping functions which returns error number, error number with messages these are

- 1) sqlcode
- 2) sqlerrm

These two function are used in either when others then clause or also used in our own exception handler.



sqlcode fn returns error number  
whereas sqlerrm returns error number with error message.

Generally if you want to identify error number at runtime then we must use sqlcode function

> declare

```
i emp%rowtype
begin
select * from emp where deptno
= &deptno;
dbms_output.put_line(ename || 
' ' || i.sal || ' ' || i.deptno);
```

exception

when others then

```
dbms_output.put_line(sqlcode);
end;
```

/

Enter value for deptno: 10

-1422

> Enter value for deptno: 'a'

-1722

sed

n

Note:- Using sqlcode Function we can also handle unnamed exception e.g.

```
> begin  
  delete from dept where deptno  
    = 10;  
end;
```

Error :

ORA-02292 : integrity constraint violated child record found.

Solution

```
> begin  
  delete from dept where deptno = 10;  
exception  
when others then  
if sqlcode = -2292 then  
dbms_output.put_line('can not  
delete from master record');  
end if;  
end;  
/
```

Note: Generally we are not allowed to use sqlcode, sqlerrm Function in DML stmts. To overcome this problem, we must declare

can  
in  
variable declaration section of PL/SQL block and then assign these function return value into <sup>variable</sup> and then only these variable are used in DML stmt.

Q. Write a PL/SQL program which is used to store error no., error no with msg of a PL/SQL block into another table.

⇒  
>create table test (errno number(10), errmsg varchar2(200));

>declare  
  v\_sal number(10);  
  v\_errno number(10);  
  v\_errmsg varchar2(200);  
begin  
  select sal into v\_sal from emp;  
exception  
  when others then  
    v\_errno := sqlcode;  
    v\_errmsg := sqlerrm;  
  insert into test values(~~v\_errno~~, v\_errno, v\_errmsg);  
end;  
/

## sqlcode return values

Sqlcode not only returns error number but also returns number.

sql code return values	meaning
------------------------	---------

0	→ no error
-ve	→ oracle error
100	→ no data found
1	→ user defined exception.

```
> declare
  a exception ;
begin
raise a;
exception
when a then
dbms_output.put_line(sqlcode);
      ''           (sqlerrm);
```

o/p ⇒ 1  
User-defined Exception

We can also view sqlcode return value by passing sqlcode or sqlcode return value into sglerrm function within executable section of PL/SQL block.

e.g.

begin

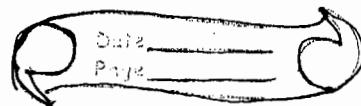
```
dbms_output.put_line (sglerrm
    (sglerrm (sqlcode)) );
dbms_output.put_line (sglerrm (100));
    "    "    "    (sglerm (1));
    "    "    "    (sglerm (-1722));
end;
```

Ora-0000: normal successful completion

Ora-01403: no data found

~~sglerm~~ User defined Exception

Ora-01722: invalid number.



## Sub Programs

Subprograms are named PL/SQL block which is used to solve particular task.

All databases system having 2 types of subprogram

- 1) Procedure
- 2) Function

show errors

### subprograms

#### Procedures

(may or may not return value)

#### Function

(must return a value)

## Stored Procedures

In oracle whenever we are using create/replace keyword in front of procedure then those procedures are automatically permanently stored in database.

That's why those procedure are also called as \$ stored procedures.

Generally procedure are used to improve performance of application because procedures having one time compilation.

Always one time compilation program units improve performance of application.

Using a Editor write a Procedure for Solving particular task.

show errors  
Using SQLplus tool loading Procedure into Oracle database.

↓  
Compiled  
↓  
Parsed  
↓  
Execute

Every procedure has 2 two parts

- 1) Procedure specification
- 2) Procedure body.

In Procedure specification we are specifying name of procedure & type of parameter. Whereas in procedure body we are solving actual task.

Syntax :-

Procedure [create or replace] procedure  
 specification  
 procedurename (formal parameter)  
 is / as  
 variable declaration, cursor declaration,  
 user defined exception ;  
 begin  
 —  
 —  
 [exception]  
 —  
 —  
 end [procedurename];

→ Formal parameter syntax :

parametername [mode] datatype  
 ——————  
 | in  
 | out  
 | in out

→ View Errors

Syntax :- show errors ;

~~Exception~~

## → Executing Procedure

Syntax :

1) sql > exec procedurename (actual parameters);

2) method 2 : (using anonymous block)

```
> begin  
procedurename (actual parameter);  
end;
```

3) method 3 : (using call stmt)

```
sql> call procedurename (actual parameter);
```

Q. Write a PL/SQL stored procedure for passing emp no as parameter that display name of emp & his salary From emp.

⇒ > create or replace procedure p1 (p\_empno number)

is

v\_ename varchar2(10);

v\_sal number(10);

begin

select into v\_ename, v\_sal from emp where empno =

p\_empno;

Data  
Page

```
dbms_output.put_line(v_ename || ' '
|| v_sal);
end;
/
```

```
> exec p1(7902);
FORD 3000
```

method 2 (using anonymous block)

```
> begin
    p1(7902);
  end;
/
```

method 3 (using call)

```
>call p1(7566);
JONES 2000
```

In Oracle all stored procedure information stored under user\_procedures, user\_source data dictionary. If you want to retriv code of procedure then we are using user\_source data dictionary.

```
> desc user_source;
```

```
> select text from user_source
  where name = 'P1';
  ↴ Capital letter
```

Q. Write a PL/SQL stored procedures  
For passing dept no as parameter  
& display emp details from emp  
table based on passed deptno.

=>

>create or replace procedure  
p1(pdeptno number)  
is  
cursor c1 is select \* from  
emp where deptno = p-deptno;  
i emp%rowtype;  
begin  
for i in c1 loop  
dbms-output.put\_line(i.empno || ' ' ||  
i.ename || ' ' || i.deptno);  
end loop;  
end;  
/

>exec p1(10)

#### \* Procedure Parameter

In oracle  
Procedure parameters are used to pass the value into procedure and return value from procedure. In oracle we can pass upto 32 parameters.

Every procedure having 2 types  
of parameter

- i) Formal Parameter
- ii) Actual Parameter

### i) Formal Parameter

are defined in specification  
of procedure. Formal parameter  
specifies name of parameter, mode  
of parameter, type of parameter.

Syntax:

parametername [mode] datatype.

Note: In oracle whenever we are  
defining formal parameter then  
we are not allowed to use  
datatype size in formal parameter  
declaration.

### MODE:

Mode specifies purpose  
of parameter. Oracle procedure  
having 3 types of modes

- i) in
- ii) out
- iii) in out mode.

i) in mode

'in' mode is used to pass value into procedure in oracle by default parameter mode is in mode.  
'in' mode behaves like constant in procedure body.

syntax:

parametername in datatype;

Q. Write a PL/SQL stored procedure for passing in parameter to insert a record into dept table.

⇒ create or replace procedure 'pl'(p\_deptno in number,  
p\_dname in varchar2,  
p\_loc in varchar2)

is

begin

insert into dept values(p\_deptno,  
p\_dname, p\_loc);

dbms\_output.put\_line('u r record  
Inserted through');

end;

/

> exec pl(1,'a','b');

## \* \* Out Mode:

Out mode is used to return values from the procedure. Out parameter behaves like an uninitialized variable in procedure body. Here, explicitly we must specify out keyword.

### Syntax:

parametername out datatype ;

```
> create procedure pl
  ( a in number, b out number)
is
begin
  b := a*a;
end;
/
```

In oracle when a procedure having out or in out parameters then those type of procedures executed by using following two methods  
method 1:- (using bind variable)  
① 2 (using anonymous block)

### Method 1 (using bind Variable)

```
> variable z number;
> exec(8,:z);
> print z;
```

Z

64

## Method 2 (Using anonymous block)

> declare

```
x number(10);
```

```
begin
```

```
p1(7, x);
```

```
dbms_output.line(x);
```

```
end;
```

```
/
```

Q. Write a PL/SQL proc for passing empname as in parameter then return sal as of employee as out parameter.

⇒ > create or replace procedure p1

```
(p_empname in number, varchar2,  
p_sal out number)
```

```
is
```

```
begin
```

```
select sal into p_sal from emp  
where empname = p_empname;
```

```
end;
```

```
/
```

➤ variable x number;

➤ exec (700, :x);

➤ point x;

```
>variable x number;
>exec ('SMITH', :x);
>print x
x
2100
```

method 2 (using anonymous block)

```
declare
  x number(10);
begin
  p('FORD', x);
  p('FORD', x);
  dbms_output(x);
end;
```

3300

Q. Write a PL/SQL stored procedure  
for passing deptno as in parameter  
then return dname, location.

From dept table by using out

⇒ create or replace pi (  
p\_deptno in number  
p\_dname out varchar2  
p\_loc out varchar2)  
is  
begin  
select dname, loc ~~into~~ into  
p\_dname, p\_loc from dept  
where deptno = p\_deptno;  
end;

1

### execution

1) using bind variables

```
> variable a varchar2(10);
> variable b varchar2(10);
> exec (1.0, :a, :b);
> print a b;
```

2) using anonymous block.

```
> declare
    x varchar2(10);
    y varchar2(10);
begin
    pi p1(20, x, y);
    dbms... (x||y);
```

Q. Write a PL/SQL stored procedure for passing deptno as in parameter then return no. of emp for that deptno. by using out from emp

⇒ > create or replace procedure pl  
(p\_deptno in number,  
p\_count out number)

is

begin

```
select count(*), deptno into
p_count, p_deptno from emp
group by "deptno"
having deptno = p_deptno;
```

Date \_\_\_\_\_  
Page \_\_\_\_\_

```
select count(*) 1 into p_count
from emp
where deptno = p_deptno;
end;
/
```

- 1) using bind variable
  - > variable a number;
  - > exec pl(10,:a)
  - > print a;

a  
3

— oxo —

## \* Pass by value, pass by references

Whenever we are using modular programming all lang. supports 2 types of passing parameter mechanism.

- i) pass by value
- ii) pass by reference.

These two parameter mechanism specifies whenever we are modifying formal parameter then actual parameter are affected or not.

In pass by value <sup>main</sup> actual value does not change in program when

we are modifying formal para. because in this method internally copy of values of are passed into calling program . To overcome this problem if you want to affect actual parameter based on Formal parameter modification then we are using pass by reference method

Oracle also supports these 2 passing parameter mechanism internally when we are using subprogram parameter. In oracle by default all 'in' parameter internally uses pass by reference & also by default all 'out' parameter uses pass by value.

Whenever we are using large amount data by using out parameter then those type of procedure degrades performance of appln because internally out parameter uses pass by value that's why internally copy of value is created. To overcome this problem

oracle gi introduced 'no copy' compiler hint alongwith out parameter .

Date \_\_\_\_\_  
Page \_\_\_\_\_

>create or replace procedure  
pl(p\_ename in varchar2, p\_sal out  
no copy number)  
is  
begin  
select sal into p\_sal from emp  
where ename = p\_ename;

### 3) in out mode

This mode is used to pass the value into procedure & also return value from procedure. This mode behaves constant, initialized variable within procedure body. Here also explicitly we can specify 'in out' keyword.

syntax:

parametername in out datatype(size)

create or replace pl procedure  
pl(a in out number)  
is  
begin  
a := a \* a;  
end;  
/

execution

method 1) (using bind variable)

```
>variable z number;
```

```
>exec :z:=8;
```

```
>exec pl(z);
```

Z

64

method 2 (using anonymous block)

procedure

```
>declare
```

```
a number(10) := & a;
```

begin

pl(a)

dbms\_output.put\_line(a);

end;

/

Enter value for a: 9

Q Write a PL/SQL store procedure by using in out parameter For passing empno then return sal of emp from emp table.

→ >create or replace procedure p1  
(a in out number)

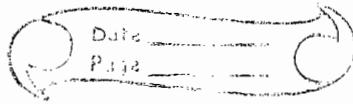
is

begin

select sal into a from emp

where empno = a;

end;



method 1

> variable z number;

> exec :z:=7902

> exec pl(z);

z

2000

SRI RAGHAVENDRA XEROX

Software Languages Material Available

Beside Bangalore Ayyagar Bakery,

Opp. CDAC, Balkampet Road,

Ameerpet, Hyderabad.

\* in parameter execution method

In oracle 'in' parameter having 3 types of execution method.  
i) positional notations  
ii) named notations ( $\Rightarrow$ )  
iii) mixed notations

i) exec pl(1,'a','b');

ii) (p\_name  $\Rightarrow$  'x', p\_loc  $\Rightarrow$  'y',  
p\_deptno  $\Rightarrow$  2);

iii) mixed notation:- It is the combination of positional, named notations. Generally after positional there can be all named but after named there cannot be positional.

e.g. > exec pl(3, p\_loc  $\Rightarrow$  'y',  
p\_cname  $\Rightarrow$  'x');

→ In oracle we can also pass default value by using in parameters through default or := operators.

syntax      datatype  
parametername in ↑ default [:=]  
default value \$

e.g.

>create or replace procedure  
pl(p-deptno, ~~p~~ 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;  
>exec pl(5,'z');

## Autonomous Transactions

are independent transaction used in anonymous block, procedures, functions, triggers. If you want to make procedure autonomous then we are using

autonomous transaction pragma ,  
commit .

syntax

pragma autonomous\_transaction ;

This pragma is used in declare  
section of procedure.

syntax:

create or replace procedure ~~#~~ '

procedurename

is/as

pragma autonomous\_transaction ;

.....  
begin

commit ;

[exception] ;

end [procedurename] ;

Generally we are using  
autonomous transaction in child  
procedure. These autonomous  
procedure transaction never affected  
by using commit or rollback  
within main transaction.

> create table test(name varchar2(10));

## Using autonomous transaction

> create or replace procedure pl  
is

pragma autonomous transaction  
begin

insert into test values('india');  
commit;

end;

/

SRI RAGHAVENDRA XEROX

Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

> begin

insert into test values('hyd');

insert into test values('mumbai');  
pl;

rollback;

end;

/

select \* from test

Name

india

Without using autonomous transaction

> create or replace procedure pl  
is

begin

insert into test values('india');

commit;

end;

Date \_\_\_\_\_  
Page \_\_\_\_\_

7 begin  
insert into test values ('hyd');  
insert into test values ('mumbai');  
pl;  
commit;  
rollback;  
end;

Name

hyd

mumbai

India.

In oracle when a procedure having commit & also when we try call this procedure in another PLSQL block then this procedure commit not only save procedure transaction but also all the above procedure transaction within database.

To overcome this problem oracle

8.1.6 introduced autonomous transaction within procedure

\* Autonomous transaction used in anonymous block.

> create table test (sno number(2));

Session 1Main transaction

```
'');
ai');
>insert into test values(1);
>insert into test values(2);

child Transaction (using autonomous)
>declare
    pragma autonomous transaction
    begin
        for i in 3.. 10
loop
        insert into test values(i);
        end loop;
        commit;
    end;
```

```
procedure
;
begin
    > select * from test;
        1 ... 10
    > rollback;
```

```
> select * from test
        3 ... 10 ;
```

Session 2

```
> select * from test
no rows selected
(before child trnx )
```

```
> select * from test (after )
        3 ... 10
```

\* authid current\_user :

When a procedure having authid current user clause then owner of procedure only allowed to execute that procedure.

Another user not allowed to execute that procedure if they are giving permission also.

Generally in oracle database whenever we are reading data from table & also perform some table operation then only data security point of view some user uses authid current\_user clause within procedure.

This clause is used in specification of procedure.

Syntax :

```
* create or replace procedure
procedurename(Formal parameters)
authid current_user
is/as
begin
```

In oracle using execute object privilege we can give procedure, function, pre define packages from one user into another user.

Syntax:

grant execute on procedurename to username1, username2...;

> conn & scott/tiger;

> create or replace procedure pl(p-empho  
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-empho;

dbms\_output.put\_line( v-ename || ''  
|| v-sal );

end;

/

> grant execute on pl to murali;

> conn murali/murali;

> exec scott.pl(7902)

Error: table or view does not exist.

## Handled (or) Unhandled exception in Procedure

In oracle whenever we are calling inner procedure into outer procedure then we must implement inner procedure exception handler within inner procedure otherwise oracle server automatically execute outer procedure default exception handler.

→ W.  
inner procedure

create or replace procedure  
p1(x in number, y in number)  
is

```
begin
dbms_output.put_line (x/y);
exception
when zero_divide then
dbms_output.put_line ('y cannot
be zero');
end;
/
```

Outer procedure

create procedure p2
is
begin
p1(5,0)

exception

when others then  
d

more

procedure

) -

lure

→ We can also drop procedure by  
using  
drop procedure procedurename;

) e  
ren)

not

## Function

Function is a named PL/SQL block which is used to solve particular task & also function must return value.

Function also having 2 parts  
1) Function specification  
2) Function body.

Note-

In F<sup>n</sup> specif<sup>n</sup> we are specifying name of function & type of parameter whereas in Function body we are solving actual task.

Specifi  
cation } create or replace Function  
            Functioname (Formal parameter)  
            return datatype  
            is/as  
            variable declaration, cursors,  
            user-defined exception  
            begin  
            —  
            return Expression ;  
            [Exception]  
            —  
            —  
            end [Function name];

## Executing a Function

PL/SQL

live  
action

2

on

Note:- are  
nd  
s  
solving

Method 1 : (using select stmt)

> select Functionname(actual parameter)  
From dual;

When a Function does not have or  
Function having all in parameter then  
those are allowed to execute by  
using select stmt.

Method 2 : (using anonymous block)

> begin  
varname := Functionname(actual  
parameters).

e.g.

> create or replace Function  
F1(a varchar2)

return varchar2

is → begin

return a;

end;

/

0711



Method 1

> select f('hi') from dual;

M

Method (Anonymous block)

7.

E

> declare

a varchar2(10);

N

begin

a := f1('hi');

7.

end dbms\_output.put\_line(a);

:

end;

/

:

Q. Write a PL/SQL stored function

E

For passing no. as a parameter

M

that return a message either

even or odd.

M



> create or replace Function

7

f1(a in number)

7

return varchar2

is

begin

if mod(a, 2) = 0

7

then

return 'even';

else

return 'odd';

M

end if

end;

/

7

### Method 1 (using select)

> select F(4) from dual;  
even

### Method 2

> declare  
(a);  
x varchar2(10);  
begin  
x := F1(7);  
dbms\_output.put\_line (x);  
end;

### Method 3 (using bind variable)

> variable z varchar2(10);

> begin  
:z := F1(8);  
end;

/  
> print z;

z

even

SRI RAGHAVENDRA XEROX

Software Languages Material Available

Beside Bangalore Ayyagar Bakery,

Opp. CDAC, Balkampet Road,

Ameerpet, Hyderabad.

### Method 4

> exec dbms\_output.put\_line (F1(19));

## Method 5

```
>begin  
dbms_output.put_line(f1(8));  
end;  
/
```

We can also use user defined Function in DML stmts.

```
>create table test(msg varchar2(10));  
>insert into test values(f1(8));
```

DML Statements are used in F# Functions.

Q. Write PL/SQL stored function

For passing empno as parameter then delete emp record in emp table & also return deleted no. of records number.

```
>create or replace function
```

```
f1(p empno number)
```

```
return number
```

```
is
```

```
l r count number(10);
```

```
begin
```

5

```

delete from emp where empno =
    p.empno;
v_count := sql%rowcount;
dbms_output.put_line(v_count);
end;
/

```

In oracle we can also use DML stmts in user defined function but we are not allowed to execute those function by using select stmt. And allowed to execute using anonymous block.

>select f1() from dual;  
error: cannot perform dml oprn  
inside a query (select stmt).

```

> declare
    a number(10);
begin
    a := f1();
    dbms_output.put_line('a' || a);
end;
/
0

```

In oracle if you want to execute a DML Function by using select stmts then we must use

autonomous transaction within user defined Functions.

7C  
E

e.g.

> create or replace Function

F1(p\_empho number)

return number

is

v\_count number(10);

pragma autonomous\_transaction;

begin

delete from emp where

empno = p\_empho ;

v\_count := sql%rowcount;

commit;

return v\_count;

end;

/

> ~~se~~ select F1(1) From dual;

F1(1)

0

g11

Q. W.

Fc

jc

F

C

F

Q. Write a PL/SQL stored Function which returns max(sal) From emp.

x

i:

j

>create or replace function f1

ff return number

is

v-sal number(10);

begin

select <sup>max</sup>(sal) into v-sal from emp;

end;

/

In oracle we can also use predefined function within user defined function

& also execute these user defined function by using same table or by using another table.

> select ename, sal, f1 From emp;

>select f1 From dual;

9/11

Q. Write a PL/SQL stored Function for passing emp-name & return job of employee based on passed parameter.

action → create or replace function

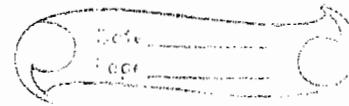
f1(p-empno)

return varchar2;

is

v-job varchar2(10);

begin



select job into v-job From  
emp where ename = p-ename;  
return  
↓ job; exception

when no-data Found then  
dbms\_out.put\_line(' emp name  
does not exist');  
when others then  
dbms\_output.put\_line(sqlerrm);  
end;

/

return v-job;

exception

when no-data found then  
return 'u r employee does  
not exist';  
end;

/

select F1('SMITH') From dual;

CLERK

select F('abc') From dual;  
u r employee does not exist.

Q. Write a PL/SQL stored F?  
For passing empno as  
parameter then check sal  
of emp if that sal is  
more than the avg(sal) of

their deptno then return 1  
otherwise return 0.

⇒ create or replace function

F1(p-empno, number)

return number

is

    v-sal number(10);

    cursor c1 is sele

    v-maxsal number(10);

    v-deptno number(10);

begin

    select v/m:max(sal)

    select v-sal, v-deptno

    select sal, deptno into

        into v-sal, v-deptno from emp

    where empno = p-empno;

    select avg(sal) into v-maxsal

    from emp

    where deptno = v-deptno;

    if v-sal > v-maxsal then

        return 1

    else

        return 0

    end if;

    exception

        when no\_data\_found then

            return('u/r emp does not exist')

    end;

create or replace Function

Q. h

F1 ( p-empno number )

d

return number

Jr

is

G

v\_sal number(10);

→

v\_avgsal number(10);

v\_deptno number(10);

begin

Q. In

select sal, deptno into v\_sal,

Pt

v\_deptno from emp

t

where empno = p-empno;

e

→

select avg(sal) into v\_avgsal

c

from emp where deptno = v\_deptno;

I

if v\_sal > @ v\_avgsal then

i

return 1;

A

else

:

return 0;

S

end if;

F

exception

when no data found then

V

return -1;

end;

:

1

8

> select F1(7902) from dual;

1

Q. Write a query by using this user defined fn to retrieve the employees who are getting more sal than avg(sal) of their deptno from emp

→ select ename, sal from emp  
where f1(empno) = 1;

Q. Write a PL/SQL stored Function for passing empno , date as parameter then return no. of years that employee is working from emp based on passed date.

→

```

sal
deptno;
create or replace function
f1(p_empno number, p_date date)
return number
is
v_hiredate date; v_years number
begin

```

```

select hiredate into v_hiredate
from emp where empno = p_empno;

```

```

v_years := round((months_between
(v_hiredate, p_date))/12);

```

```

return v_years;

```

el





x number(10);  
begin  
select months\_between(p\_date,  
hiredate)/12 from emp  
where empno = p.empno;  
return round(x);  
end;  
/

> select f1(7902, sysdate)  
from dual;

> select empno, ename, hiredate,  
f1(empno, sysdate) || ' ' || years  
"Experience" from emp  
where empno = 7902;

Q. Write a PL/SQL Function which  
is used to calculate tax of  
the employee based on passing  
empno as a parameter from  
emp table by using following  
condn.

- 1) if annsal > 10000 then  
tax → 10% of annsal
- 2) if annsal > 15000 then  
tax → 20% of annsal
- 3) if annsal > 2000 then  
tax → 30% of annsal

> create or replace function tax(p\_empho)  
return number  
number)

is

v\_sal number(10);

annsal number(10);

x number(10);

begin

select sal into v\_sal from emp

where empno = p\_empho;

annsal = v\_sal \* 12;

if annsal > 10000 and annsal < 15000 then

x := annsal \* 0.1 ;

else if annsal > 15000 and annsal < 20000 then

x := annsal \* 0.2 ;

else if annsal > 20000 then

x := annsal \* 0.3 ;

else

x := 0 ;

end if ;

return x ;

end ;

### Out Mode:

Out mode is used to return value from F<sup>n</sup> if u want to return more than one value of the F<sup>n</sup> then only we are using out mode parameter.



exists collection method used in index by table.

exists collection method used in index by table, nested table, varray. exist collection method also returns boolean value either true or False.

Generally if you want to test requested data available or not available in collection then we are using exist collection method.

Syntax :-

collectionvarname.exists(indexvarname)

> declare

type t1 is table of number(10)  
index by binary\_integer;

v\_t t1;

z boolean;

begin

v\_t(1) := 10;

v\_t(2) := 20;

v\_t(3) := 30;

v\_t(4) := 40;

v\_t(5) := 50;

in  
used  
able,  
I  
either  
,  
=0  
when  
?  
on name  
> 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

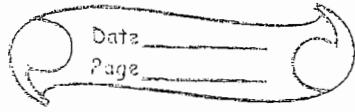
ora-1403 no data

In oracle whenever index by table & nested table having gaps then we try to display data in those collection then oracle server returns an error ora-1403 no data Found.

For handling this error then we are using exist collection method.

Solution (using exist collection method)

```
>declare
  type t1 is table 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 loop;
  end;
```



## Nested table, varray

Oracle 8.0 introduced nested table, varray these also user defined type which is used to store number of data item in single item. Before we are storing data in those collection we must initialize these collection by using constructor.

Here constructor name is same as type name

### Nested Table

is user defined type which is used to store no. of data item in single unit. These collection does not have key-value Field. Here always index starts with 1. These indexes also consecutive. These indexes are integer only.

Generally we are not allowed to store index by table permanently into oracle database & also in index by table we are not allowed to add or remove indexes. To overcome this problem oracle 8.0 introduced extension of index by table

called nested table which is used to store permanently into database by using SQL and also we are allowed to add or remove indexes by using extend, trim collection method.

Nested table having extend, trim, first, last, prior, next, count, exists, delete(index), delete(index1, indexn); delete collection method.

This is an user defined type so we are creating a two step process i.e.  
1<sup>st</sup> we are creating type then only we are creating variable from type.

Syntax:

- 1) Type typename is table  
of datatype(size);
- 2) variablename Typename  
:= Typename();  
→ constructorname

> declare

type t1 is table of number(10)  
index by binary\_integer;  
v t t1;

v

```
s begin
into v_t(500) := 30;
2/50 dbms_output.put_line(v_t(500));
z end; /
30
```

hod.  
Index by table are basically sparse i.e. no need to allocate memory explicitly upto those indexes.

> declare

```
method type t1 is table of number(10);
ned v_t t1 := t1();
g begin
then v_t(500) := 30;
ble dbms_output.put_line(v_t(500));
end;
/
```

Error : Subscript beyond count.

Nested table are basically dense i.e. th we must reserve memory explicitly before we are storing actual data.

> declare

```
name type t1 is table of number(10);
der(10) v_t t1 := t1();
begin
v_t.extend(500);
v_t(500) := 30;
```

Date \_\_\_\_\_  
Page \_\_\_\_\_

```
dbms_output.put_line(v-t(500));  
end;  
/
```

→ 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.

e.g.

```
> 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;
```

e.g. 2) > 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');
```

2)

3)

```

500);
else
dbms_output.put_line ('v_t1 is
not null');
end if;
so
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;

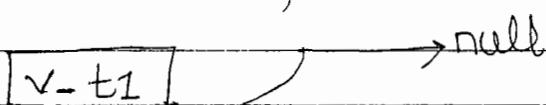
```

er(10);  
, 50);

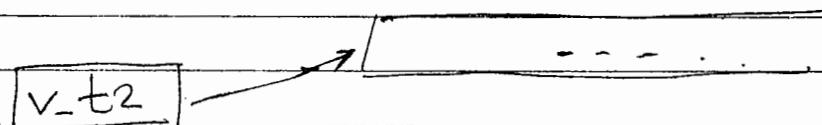
O/P

v\_t1 is null  
v\_t2 is not null

: (i) 1) v\_t1 t1;



2) v\_t2 t1 := t1();



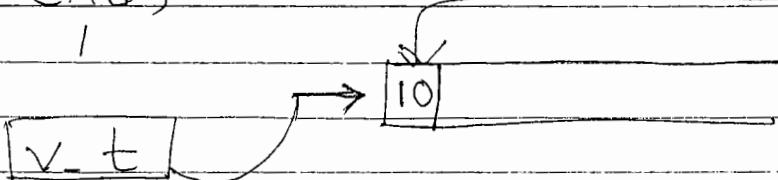
3) > declare

type t1 is table of number(10);  
v\_t t1 := t1();

```

begin
  v_t.extend;
  v_t(1) := 10;
  dbms_output.put_line(v_t);
end;

```



e.g > declare

```

type t1 is table of number(10);
v_t t1 := t1(10, 20, 30, 40);
begin
  dbms_output.put_line(v_t.first);
  ..
  (v_t.last);
  ,
  (v_t.count);
  (v_t.prior(3));
  (v_t.next(3));

```

```

v_t.extend;
v_t(5) := 50; → value at
v_t.extend(3, 2); index 2
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;

```

```

v-t.delete;
dbms_output.put_line(v-t.count);
end;
/

```

difference between trim, delete.

> 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
trimming 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;

/

Q. Write a PL/SQL program which is used to transfer all emp name & stored it into nested table & display content from nested table.

> declare

```
type t1 is table of varchar2(10);
vt t1 := t1();
cursor c1 is select ename
from emp;
n number(10) := 1;
begin
for i in c1
loop
vt.extend;
vt(n) := i.ename;
n := n + 1;
end loop;
```

For i in vt.first

When we are transferring data from table into nested table through bulk collect clause then we are not allowed to use extend collection method

because bulk collect clause internally  
all reserved memory in nested table.

declare

```
type t1 is table of varchar2(10);  
vt t1 := t1();
```

begin

```
select ename bulk collect into  
vt from emp;
```

for i in vt.first .. vt.last

loop

```
dbms_output.put_line(vt(i));
```

end loop;

end;

/

SRI RAGHAVENDRA XEROX

Software Languages Material Available

Beside Bangalore Ayyagar Bakery,

Opp. CDAC, Balkampet Road,

Ameerpet, Hyderabad.

Varray :-

varray also user defined type which is used to store no. of data item in single unit. It stores upto 2gb data, here also indexes are starts with 1 and also those indexes are consecutive.

Before we are storing actual data in varray, we must use constructor.

(This is an user define type so we are creating 2 steps.

## Syntax :-

1) type typename is varray(maxsize)  
of datatype (size);

2) variablename typename  
:= typename();

> declare

type t1 is varray(10)  
of varchar2(10);

Q.

v\_t t1 := t1('a','b','c','d');

begin

dbms\_output.put\_line(v\_t.limit);  
" (v\_t.count);  
" (v\_t.prior(3));  
" (v\_t.next(3));

⇒

v\_t.extend(3, 2);

v\_t.trim;

For i in v\_t.first .. v\_t.last

loop

dbms\_output.put\_line(v\_t(i));

end loop;

v\_t.delete;

dbms\_output.put\_line(v\_t.count);

end;

✓  
loop

In varrays we are not allowed to delete particular element or range of indexes by using delete collection method.

But we are allowed to delete all indexes by using delete collection method.

Q. Write a PL/SQL program which is used to transfer first 10 emp name from emp table into varray and also display content from varray.

```

it)j    → declare
it)j      type t1 is varray of varchar2(10);
))j      v.t t1 := t1();
)j

begin
  select ename bulk collect into
  v.t from emp
  where rownum <= 10;
)j

loop → for i in v.t.first .. v.t.last
      dbms_output.put_line(v.t(i));
    end loop;
int); end;
)j
curr
  
```

DIFF bet'n index by table  
nested table, varray.

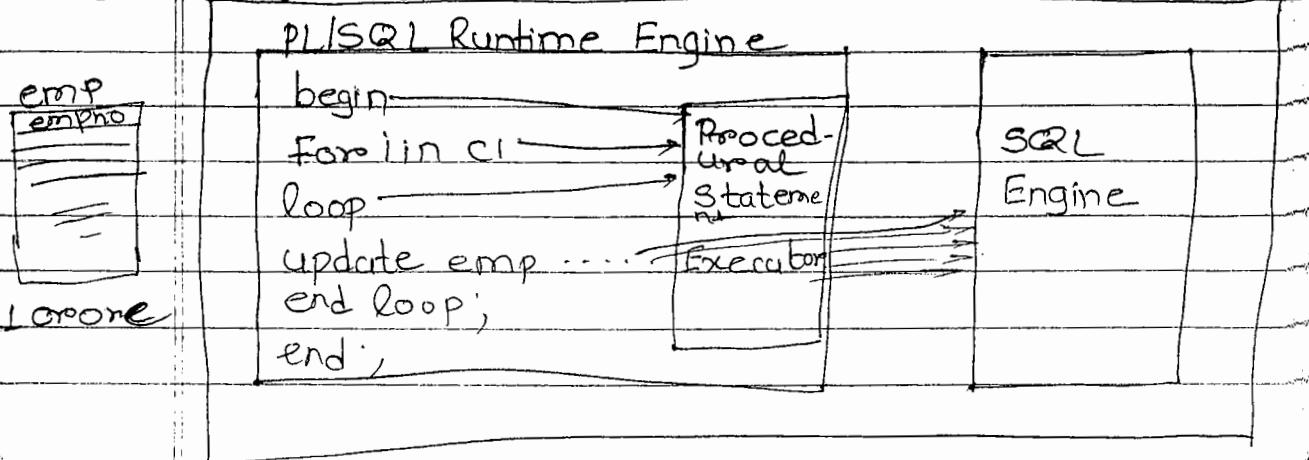
index by table	nested table	varray
1) This is unbound table having keyvalue pairs	This is unbound table b.	This is bounded table stores upto 2gb data.
2) We cannot add or remove indexes	2) We can add or remove indexes by using extend, trim, collection method	3) We can add or remove indexes by using extend, trim.
3) Here indexes are integer, character, and also indexes are +ve, -ve numbers	3) Here indexes are number & by default starts with 1.	3) Here indexes are number & starts with 1.
4) We are not allowed to store index by table permanently in oracle	4) We are allowed to store nested by table permanently in oracle by using SQL	4) We can store varray permanently in oracle dbase by using SQL
5) Index by table having exist, first, last, prior, next, count, delete (index), delete(index1, indexn) delete collection method	5) Nested table having exist ..... delete, extend, trim	5) Varray has exists, limit, extend, trim, first, last, prior, next, count, delete.

## B Bulk Bind

Whenever we are submitting PL/SQL block into oracle server then all SQL stmts. are executed by using SQL engine and also all procedures stmt are executed separately by using procedural stmt executor within PL/SQL engine. These type of execution method are also called as context switching execution method.

Whenever PL/SQL block having more number of SQL, procedure stmt. Then theses context switching execution method degrades performance of appln". To overcome this problem to improve performance of appln Oracle 8i introduced bulk bind process through collection.

Without using bulk bind process (Performance penalties For many Oracle Server context switches)



Oracle 8i introduced bulk bind process which is used to improve performance of application.

In this bulk bind process 1<sup>st</sup> we are fetching data from table into collection and then process all data in collection by using SQL engine through Forall stmts.

Syntax :-

Forall indexvarname in  
collectionvarname.first . . .  
collectionvarname.last

step 1

DML stmts where columnname  
= collectionvarname(indexvarname);

In bulk bind process we are executing multiple DML stmts at a time. In this process we are using bulk update, bulk delete, bulk insert.

## Oracle server

## PL/SQL runtime engine

begin

For all i in v\_t.first  
.. v\_t.lastupdate emp set sal =  
sal+100 where  
empno = v\_t(i);

end;

Proced-  
eral  
stmt  
executorSQL  
Engine

Bulk bind is a 2 step process

- step 1) Fetching data from resource into collection by using bulk collect clause.
- 2) Process all data in a collection at a time by using SQL engine through forall stmts.

Step 1 :-

Before we are executing multiple DML stmts at a time by SQL engine through forall stmt then we must fetch data from resource & store it into collection by using bulk collect clause.

bulk collect <sup>clause</sup> used in

- 1) select into clause
- 2) cursor fetch stmt

3) DML... returning ... into clause. 2)

- bulk collect clause used in select into clause.

syntax:

```
select * bulk collect into
collectionvarname from tablename
where condition;
```

e.g.

declare

```
type t1 is table of emp%rowtype
index by binary integer;
v_t t1;
```

begin

```
select * bulk collect into v_t
From emp;
```

For i in v\_t.first .. v\_t.last

loop

```
dbms_output.put_line(v_t(i).ename);
```

end loop;

end;

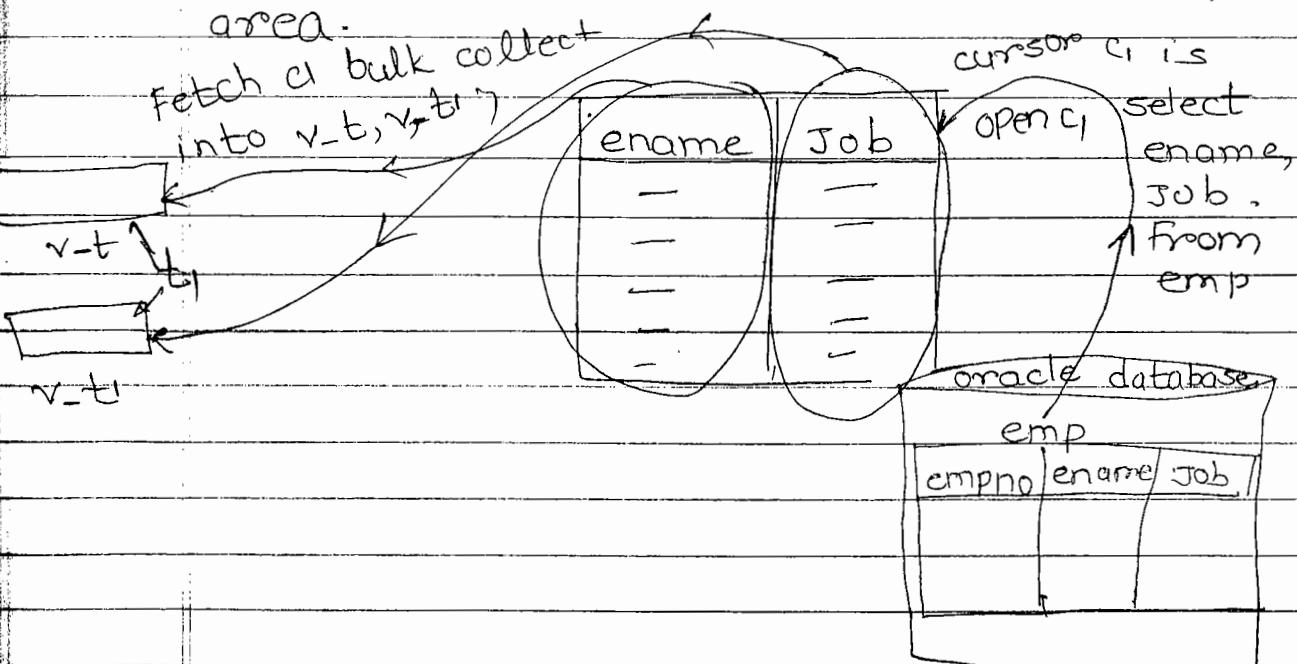
/

se. 2) bulk collect clause used in cursor  
fetch stmt.

Syntax :-

Fetch cursorname bulk collect into  
collectionvarname [limit anynumber]

Here limit is an option clause which  
is used to restrict no. of rows  
within pga memory area. Because  
collection are executed in pga memory  
area.



e.g.  
declare

```
type t1 is table of varchar2(10)
index by binary_integer;
v_t t1;
v_t1 t1;
```

Spiral  
Page

```
Fetch c1
into v-t
v-t1

cursor c1 is select ename,job
from emp;
begin
open c1;
Fetch c1 into 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 you want to calculate elapsed time of PL/SQL block then we are using get\_time fn from dbms\_utility package. This Function always return F no. datatype.

Syntax

```
var := dbms_utility.get_time;
```

Fetch or bulk collect

Page

Page

into v-<sup>b</sup>)

$v - t$

### Cursor C.

Object-

cursor si is select

~~Open CI owner, object-~~

name from  
all objects;

data dictionary

11

$$z-t \rightarrow t_1$$

111)

v-t<sub>1</sub>

— 1 —

e.g.

> declare

Type `tl` is table of all objects.

object-name %. type

index by binary-integer;

$$v-t \quad t_1;$$

$$v - t_1 + t_1 ;$$

cursor C1 is select owner,  
object\_name from all-objects ;

Z1 number(10);

22 number(10);

begin

```
z1 := dbms_utility.get_time;
```

~~dbms\_output.put\_line~~

20/11/15

For i in .c1  
loop  
null;  
end loop;

3)

z2 := dbms\_utility.get\_time;

dbms\_output.put\_line('Elapsed  
time for normal fetch' ||  
' ' || z2 - z1 || ' ' || hsecs);

for z1 := dbms\_utility.get\_time;  
Fetch c1 bulk collect into  
v\_t, v\_t1)  
close c1

z2 := dbms\_utility.get\_time;

dbms\_output.put\_line('Elapsed  
time for bulk collect fetch' ||  
' ' || z2 - z1 || ' ' || hsecs);  
end;

1

20/12/15

R Date \_\_\_\_\_  
Page \_\_\_\_\_

### 3) dml returning into clause :-

returning into clauses are used in DML stmts only. These clauses are used to return transaction value from DML stmts into variable.

e.g. > variable a varchar2(10);  
> update emp set sal=sal+100  
where ename = 'KING' returning  
job into :a

> print a;

In oracle 8i onwards we can also use bulk collect clause within DML returning into clauses.

In this case oracle server returns multiple processed rows from table into collection.

> declare

type t1 is table of number(10);  
index by binary integer;

v\_t t1;

begin

update emp set sal=sal+100  
where ename = 'KING' job = 'CLERK'  
returning sal bulk collect  
into v\_t;

dbms\_output.put\_line ('affected  
number of clerks are : ' || ' ||  
sql%rowcount );

For i in v\_t.First .. v\_t.last  
loop

dbms\_output.put\_line (v\_t(i));  
end loop;

end;

/

step 2) process all data in a collection  
at a time by using sql engine  
through forall stmts.

Once data is in a collection  
then we can process data  
by using this collection data with  
in database. Then automatically  
Oracle server reduces no. of  
context switches through forall  
stmts. This is called bulk  
bind.

In bulk bind process  
we are executing multiple DML  
stmts at a time by using  
forall stmt.



ed  
 '11  
 2st  
 1) i  
 Using Forall stmts we can perform bulk updates, bulk delete, bulk inserts.

Forall indexvarname collectionvarname.  
 First .. collectionvarname.last  
 DML stmt where columnname =  
 collectionvarname (indexvarname);

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;

1

with  
 ally  
 F  
 all  
 -  
 is  
 L  
 write a PL/SQL bulk bind program  
 which is used to retrieve all  
 empno from emp table into index  
 by table by using bulk collect  
 & then only use forall stmt  
 which updates sal of these emp  
 at a time within emp table.

⇒ 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;

For all

forall i in v\_t.first .. v\_t.last  
update emp set sal=sal+100  
where empno = v\_t(i);

end;

/

→ >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 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 are not allowed to use bulk bind process (Forall stmts). To overcome this problem we are using varrays in bulk bind process because varrays does not have any gaps but using varrays we can store upto 2gb data to overcome all these problem, oracle 10g introduced "indices of" clause in bulk bind process.

This clause is used in forall stmts.

Syntax :

forall indexvarname in indices of collectionvarname ~~.First .. collectionvarname.last~~

DML stmt where columnname = collectionvarname(indexvarname);

before Oracle 10g

empno

First

7369  
7566  
7902  
7839  
7788

update all  
rows in  
order of  
array listing

empno

7369

7902

7839

at Oracle 10g (indices of)

$\forall(1) :=$

7369

$\forall(2) :=$

7902

$\forall(3) :=$

7839

$\forall(4) :=$

7788

update all rows within a array  
inorder of array variables  
(or) indexes.

soln

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 indices of v\_t  
update emp set sal = sal + 100  
where empno = v\_t(i);  
end;

1

sql%bulk\_rowcount:

In bulk bind process if  
you want to return affected  
no. of rows within each process  
then we are using  
sql%bulk\_rowcount attribute.

sql%bulk\_rowcount(indexvarname)

say

> declare

type t1 is varray of number(10);  
 vt t1 := t1(10, 20, 30, 40, 50);  
 begin

forall i in vt.First .. vt.last  
 update emp set sal = sal + 100  
 where deptno = vt(i);

for i in vt.First .. vt.last  
 loop

dbms\_output.put\_line ('Affected  
 no. of rows in deptno ' || ' ||  
 vt(i) || ' || ' is ' || ' ||  
 sql%bulk\_rowcount(i));  
 end loop;  
 end;

bulk delete

SRI RAGHAVENDRA XEROX  
 Software Languages Material Available  
 Beside Bangalore Ayyagar Bakery,  
 Opp. CDAC, Balkampet Road,  
 Ameerpet, Hyderabad.

> declare

type t1 is varray(10) of number(10);  
 vt t1 := t1(10, 20, 30, 40, 50);

begin

forall i in vt.first .. vt.last  
 delete from emp where  
 deptno = vt(i);  
 end;

## bulk insert

> create table target(name  
varchar2(10));

> 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(3) := 'abc';

v\_t(4) := 'xyz';

Forall i in v\_t.first - v\_t.last  
insert into targets values(v\_t(i));  
end;

/

### \* forall & DML errors / Bulk Exception

In oracle forall stmts  
typically executes multiple DML  
stmts whenever an exception  
occurs in one of those DML  
stmts then default behaviour is

1) that stmt is rollbacked and forall stops.

2) all (previous) successful stmt are not rollbacked.

If you want to continue forall processing even if an error occurs in one of those DML stmts then

just add "save exception" clause in bulk bind process

save exception clause tells to oracle , save exception information and continue processing all DML stmts.

Whenever we are using save exception clause , for each exception raised oracle automatically populates sql%bulk\_exceptions pseudo collection. This collection having only collection method count.

In bulk bind process whenever exception raised

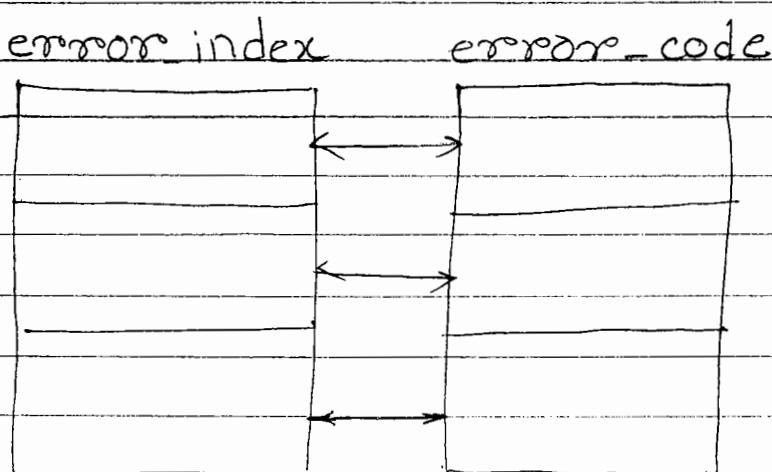
sql%bulk\_exception index by table. This index by table having two fields these are error\_index , error\_code.

error\_index:-

error index stores  
index no. of error.

error\_code :- It stores error number (only positive) of error occurred in bulk bind process.

## sql%bulk-exception



If you want to handle bulk exception then we must follow 2 steps.

Step 1) Store no. of exceptions in a variable by using count collection method sql%.- bulk exception index by table.

This collection method is used in exception section of PL/SQL block

by using following syntax.

### syntax

or      varname := sql%bulk\_exceptions.count;

### Step 2

Before we are using this process  
we must use save exception clause  
in forall stmts.

### syntax:

forall indexvarname in  
collectionvarname.first ..  
collectionvarname.last save exception

DML stmts where columnname =  
collectionvarname(indexvarname);

e.g.

>create table target (sno number(10)  
not null);

>declare

type t1 is table of number(10);  
v\_t t1 := t1(10, 20, 30, 40, 50);

# begin

v\_t(2) := null;

v\_t(3) := null;

Forall i in v\_t.first .. v\_t.last  
insert into target values  
(v\_t(i));  
end;  
1

error : ora-1400 : cannot insert  
null into not null.

### solution

```
>declare
  type t1 is table of number(10);
  v_t t1 := t1(10, 20, 30, 40, 50);
  z number(10);
begin
  v_t(2) := null;
  v_t(3) := null;
```

Forall i in v\_t.First .. v\_t.last  
save exception  
insert into target values  
(v\_t(i));

exception

when others then

```
z := sql%bulk_exceptions.count;
dbms_output.put_line(z);
end;
```

1

2

5

last

> select \* from target

SNO

10

40

50

but

→ If you want to display index no. of errors, error nos. within bulk bind process then we are using error\_index, error\_code fields within a loop by using following syntax

10);

);

sql%bulk-exceptions(i).indexvarname).

error\_index

sql%bulk\_exceptions

(indexvarname).error\_code

last

> declare

type t1 is table of number(10);

l\_v\_t t1 := t1(10, 20, 30, 40, 50);

z number(10);

begin

v\_t(2) := null;

v\_t(3) := null;

forall i in v\_t.first .. v\_t.last

save exception

insert into target values(v\_t(i));

unt;

exception

when others then

z := sql%bulk\_exceptions.count;

for j in 1..z

loop

dbms\_output.put\_line('Error  
index is ' || ' ||

sql%bulk\_exceptions(j).error\_index  
|| ' ' || ' error code is ' || ' ||

|| sql%bulk\_exceptions(j).error\_code);

end loop;

end;

)

## dbms\_utility

dbms\_utility package internally having index by table and also this package having following procedures

- 1) comma\_to\_table
- 2) table\_to\_comma.

### comma\_to\_table :-

It is used to store comma separated strings into index by table. This procedure accepts 3 parameters.

#### syntax :

dbms\_utility.comma\_to\_table(  
string, varname  
binary\_integer, varname)  
varname varname

### table\_to\_comma :-

It is used to convert index by table values into comma separated strings. This procedure also accepts 3 parameters.

#### syntax :

dbms\_utility.table\_to\_comma(index\_by  
table, binary\_integer, string)  
varname varname varname



Before we are using these two procedure we must declare index by table variable in declare section of PL/SQL block by using `uncl_array` predefined type from `dbms_utility` package.

### Syntax

~~dbms~~ variablename `dbms_utility.uncl_array;`

Q. Write a PL/SQL program which is used to convert comma separated string into index by table values by using `dbms_utility` and also display content from index by table.

→ declare  
    `v_t dbms_utility.uncl_array;`  
    `z binary_integer,`  
    `str varchar(200);`  
begin  
    `str := 'a, b, c, d, e, f';`  
    `dbms_utility.comma_to_table`  
    `(str, z, v_t);`  
    for i in v\_t.first .. v\_t.last  
    loop  
        `dbms_output.put_line(v_t(i));`

end loop;

end;

1

Q. Write a PL/SQL program to retrieve all dept names from dept table display those dept name into comma separated string by using dbms-utility package.

→ declare

v-t dbms\_utility.uncl\_array;  
z binary\_integer;  
str varchar2(200);

begin

select dname bulk collect into v-t  
From dept;

dbms\_utility.table\_to\_comma  
(v-t, z, str);

dbms\_output.put\_line(str);

end;

1

Date \_\_\_\_\_  
Page \_\_\_\_\_

## Ref Cursor / Cursor Variable Dynamic Cursor.

Oracle 7.2 introduced ref cursors.

Ref cursor is an user defined type which is used to process multiple record and also this is an record by record process.

Generally in static cursor oracle server executes only one select stmt at a time for a single active set area, whereas in ref cursor oracle server executes number of select stmts dynamically for a single active set area. That's why these cursors are also called as dynamic cursor.

Generally we are not allowed to pass static cursor as parameter to sub programs whereas we are allowed to pass ref cursor as parameter to the sub programs because basically ref cursors are user defined type in all database we can pass user defined types to sub program.

Generally, a static cursor does not return multiple records into client appln whereas ref cursors returns multiple record into client appln.

This is an user defined type so we are creating in two step process i.e. 1st we are creating then only we can created variable from that type. That's why ref cursors are also called as cursor variable.

Oracle having 2 types

of ref

1) strong ref cursors

2) weak ref cursor

Strong ref cursor is a ref cursor having a return type whereas weak ref cursor does not have return type.

Syntax :-

```
type typename is ref cursor  
return recordtype datatype;  
variablename typename;
```

Strong Refcursor variable name.

2) type typename is ref cursors;  
variablename typename;

Weak Ref Cursors Variable

from emp

In ref cursor we  
are specifying select stmt in  
executable section of PL/SQL block  
by using open... for ... stmt

Syntax:

open ref\_cursorvarname for  
select stmt;

e.g.

>declare  
type t1 is ref cursor;  
v-t t1;  
i emp%rowtype;  
begin  
open v-t for select \* from emp  
loop

where sal > 2000;

Fetch v-t into i  
exit when v-t%notfound;  
dbms\_output.put\_line(i.empno||' '||  
i.sal);  
end loop;  
end close v-t;  
end;

rs;

Q. Write a PLISQL program by using ref cursor whenever user entered dept no. 10, then display 10<sup>th</sup> dept details.  
From emp dept no. 20 → display 20<sup>th</sup>, "

From dept.

→ declare

type t1 is ref cursor;  
int t1;

i dept%rowtype;

v\_deptno number(10);

begin

v\_deptno := &deptno;

if (v\_deptno = 10) then

open v-t For select \* From

dept where deptno = 10;

elsif v\_deptno = 20 then

open v-t For select \* from dept  
where deptno = 20;

else

dbms\_output.put\_line ("bad input");

end if

emp

2;

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

► 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 = <del>10;
loop
fetch v_t into i
exit when v_t%notfound;
dbms_output.put_line ('iENAME' '
|| i.sal || ' ' || i.deptno );
end loop;
```

```
elsif v_deptno = 20 then
open v_t for select * from dept
where deptno = v_deptno;
loop
```

```
Fetch v_t into j
dbms_output.put_line ('j.DNAME' '
|| j.loc );
```

else

```
dbms_output.put_line ('bad input');
end if;
close v_t;
end;
```

1

Oracle 9i introduced sys\_refcursor predefined type in place of weak ref cursor

refcursor varname sys\_refcursor;

e.g.

declare

mp v-t sys\_refcursors;

i emp%rowtype;

begin

open v-t for select \* from emp  
where sal > 2000;

for i in v-t loop

dbms\_output.put\_line('i.ename' || i.sal);

end loop;

end;

1

Lept

\* Passing ref cursors as parameter to sub program.

Passing sys\_refcursors as in parameter to stored procedure

Q. Write a PL/SQL stored procedure for passing sys-refcursors as in which display emp details from emp

>create or replace procedure pl

(p-t in & F cursor sys\_refcursor)

is

i emp%.%rowtype;

begin

for i in p-t

loop

dbms\_output.put\_line(i.ename || ' ' || i.sal || ' ' || i.deptno);

end loop;

end;

/

Note :- In oracle whenever we are passing ref cursor as in parameter to this subprogram then we are not allowed to use

open .. for .. st stmts within subprograms.

Execution

>declare

v-t sys\_refcursor;

begin

open v-t for select \* from emp;

pl(v-t);

close v-t;

end;

/

epl

\* Passing sys\_refcursors to as out to the procedure

Q. Write a PL/SQL stored procedure for passing sys\_refcursor as out parameter to procedure which returns emp detail from emp.

→ create or replace procedure PI

(v\_t out sys\_refcursor)

is

begin

open v\_t for select \* from emp;

end;

1

Execution using PL/SQL client

> declare

v\_t sys\_refcursor;

i emp%rowtype;

begin

pi(v\_t); loop

fetch v\_t into i;

exit when v\_t%notfound;

dbms\_output.put\_line(i.ename || ' ' || i.sal);

end loop;

close v\_t;

end;

Q. Write a PL/SQL stored function by using sys\_refcursor as return type which returns emp details.

→ create or replace Function F1

~~(out parameter)~~ return sys\_refcursor;

is

vt sys\_refcursor;

begin

open vt for select \* from emp;

return vt;

end;

Execution

>select F1 from dual;

Q. using pkg

>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;

tion  
P

```
procedure p2(v-t2 out t2);  
end;  
1
```

> create or replace package body pj1  
is

```
procedure p1(v-t1 out t1)
```

is

begin

```
open v-t1 for select * from emp;  
end sp1;
```

```
procedure p2(v-t2 out t2)
```

is

begin

```
open v-t2 for select * from dept;
```

end p2;

end;

1

Execution (using bind variable)

> variable a refcursor;

> variable b refcursor;

> exec pj1.p1(:a);

> exec pj1.p2(:b);

> print a b;

→ In oracle we are not allowed to create ref cursor variable in packages.

e.g.

```
> create or replace package p1
  is
    type t1 is ref cursor;
    vt t1;
  end;
  /
```

Package created with compilation error.

```
> show errors;
cursor variable cannot be declared as part of package.
```

## Local Subprograms

Local subprograms are named PLISQL block which is used to solve particular task.

Oracle having two types of local subprograms

- 1) Local Procedure
- 2) Local Function

Local subprogram does not have create or replace keyword. These sub programs are not stored in d/b.

Local subprogram are defined in anonymous blocks, within stored procedures.

In oracle Local subprograms are defined in declare section of PLISQL block & call these subprogram in immediate executable section.

In oracle we must define local sub program in bottom of declare section. When we are defining these sub programs with type, cursors, variables and also sub-programs are executed in immediate executable section.

Syntax :

declare

variable, constant declarations ;

cursor declaration ;

Type declaration ;

procedure procedurename (formal  
parameter)

is/as

- -

begin

- -

Exception

=

end [Procedure name] ;

Function Functionname (formal param)

return datatype

is/as

- -

begin

-

Exception

-

end (Function name) ;

begin

procedurename (actual parameter) ;

varname := functionname (actual parameter) ;

end;

> declare

procedure P1

is

begin

dbms.output.put\_line('Local Proc');

end P1;

begin

P1;

end;

1

local Proc;

> create or replace procedure p2

is

procedure P1 → local

↳ stored

procedure

is

begin

dbms.output.put\_line('local proc');

end P1;

begin

P1;

end;

> exec P2;

local procedure

e.g.

```
>declare
  type t1 is ref cursor emp%rowtype;
  v_t t1;
  procedure p1(p_t in t1)
  is
    i emp%rowtype;
  begin
    loop
      for i 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;
```

```
  Open v_t for select * from emp
    where ename like 'M%';
```

```
  p1(v_t);
```

```
  close v_t;
```

```
* Open v_t for select * from emp
  where job = 'CLERK';
```

```
  p1(v_t);
```

```
  close v_t;
```

```
  end;
```

\* Passing index by table as in parameters  
to local procedure.

> declare

type t1 is table of emp1.rowtype  
index by binary\_integer;  
v-t t1;

procedure p1(p-t in t1)

is

begin

for i in \* p-t.first .. p-t.last

loop

dbms-output.put\_line(p-t(i).ename);

end loop;

end p1;

begin

select \* bulk collect into v-t from  
emp;

p1(v-t);

end;

/

\* Passing index by table as out  
parameter to the local parameter

> declare

type t1 is table of emp1.rowtype  
index by binary\_integer;

v-t t1;

procedure p1(v-t out t1)

is

Date \_\_\_\_\_  
Page \_\_\_\_\_

```
begin
select * Bulk collect into p-t
from emp;
end pl;
begin
pl(v-t);
For i in v-t.first .. v-t.last
loop
dbms-output.put-line(v-t(i).ename);
end loop;
/
```

?

\* Using index by table as return type from local function.

```
declare
type t1 is table of emp%rowtype
index by binary_integer;
procedure pl( p-t in t1)
is
i emp%rowtype;
begin
for i in p-t.first .. p-t.last
loop
dbms-output.put-line(p-t(i).ename);
end loop;
end pl;
```

Function F1 returns t1  
is

v\_t t1;

begin

select \* bulk collect into v\_t from  
emp;

return v\_t;

end F1;

begin

p1(F1);

1

**SRI RAGHAVENDRA XEROX**  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

## PL/SQL Records.

This is an user defined type which is used to represent diff. data types into single unit.

It is also same as structure in C lang.

This is an user defined type so we are creating in 2 step Syntax:

```
type typename is record (attr1  
datatype(size), ...);
```

```
variablename typename;
```

```
>declare
```

```
type t1 is record (a1 number(10),  
a2 varchar2(10), a3 number(10));  
v_t t1;
```

```
begin
```

```
v_t.a1 := 101; v_t.a2 := 'murali';  
v_t.a3 := 2000;  
dbms_output.put_line(v_t.a1 || '||'  
v_t.a2 || '||' || v_t.a3);
```

```
end;
```

```
/
```

```
e.g.
```

```
> create or replace package body p1
```

```
is
```

```
type type t1 is record (a1 number(10),  
a2 varchar2(10), a3 number(10));  
procedure P1;  
end;
```

Date \_\_\_\_\_  
Page \_\_\_\_\_

```
> create or replace package body pj1
is
procedure pl
is
vt t1;
begin
select empno, ename, sal into
vt from emp where ename='KING';
dbms_output.put_line(vt.a1 || '' ||
vt.a2 || '' || vt.a3);
end pl;
end;
> cexec pj1.pl;
```

oxo

## utl\_file Package

Oracle 7.3 introduced utl\_file package. This package is used to write data into an OS file. And also read data from an OS file.

In oracle before we are using utl\_file package, LOBS then we must create alias directory, related to physical directory by using following synt

create or replace directory directoryname  
as 'path' ;

Before we are creating alias directory then db administrator 1st give create any directory system

privilege to user otherwise oracle server returns error.

Syntax :

> grant create any directory to username;

e.g. > conn sys as sysdba;  
Enter passwd : sys

> grant create any directory to scott;

> create or replace directory  
XYZ as 'c:\';  
Directory created.

Before we are using read, write oprn then we must give read, write object privileges in alias directory by using following syntax.

syntax :

grant read, write on directory  
directoryname to username;

> conn sys as sysdba;  
passwd: sys

> grant read, write on directory XYZ  
to scott;

> conn scott/tiger;

If you want to write data into File then we are using either PUTF or PUT\_LINE procedure

6 also if you want to read from file then we are using get\_line procedure from utl package.

step 4

syntax:

Wrting data into os file.

step 1 :- Before we are opening we must create file pointer variable by using File-type from utl\_file package in declare section of PL/SQL block.

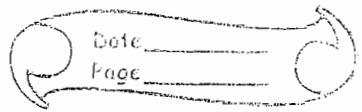
Filepointer variablename utl\_file.File-type;

2:- Before we are writing data into file then we must open file by using EOpen() function From utl\_file. This fn is used in executable section of PL/SQL block. This fn accepts 3 parameters & returns file\_type datatype.

Syntax:- filepointervarname := utl\_file.Fopen  
( 'aliasdirname' , 'filename' , 'mode' );  
w- write, read - r, a

3:- After opening file if you want to store 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 pkg.  
syntax: utl\_file.Fclose(Filepointer varname);

e.g. > declare

```
    Fp utl_file.File type;
begin
    Fp := utl_file.Fopen('xyz', 'File1.txt',
                           'w');
    utl_file.putf(Fp, 'abcdef');
    utl_file.close(Fp);
end;
/
```

Q. Write a PL/SQL program to retrieve all empname from emp table & store it into a file by utl\_file pkg



Q.

```

→ > 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;

```

In oracle when we are trying to write table column data into a os file by using putf procedure then oracle server selects column data & then write into os file in horizontal manner. To overcome this problem if you want to write data into our own format then we must use ''s'' access specifier within 2<sup>nd</sup> parameter of putf procedure and also if you want to store column data in vertical manner '\n' new line character alongwith ''s'' access specifier.

Syntax :-

utl\_file.putf (filepointer varname,  
'%s \n', variablename);

e.g.

> declare

  fp utl\_file.file\_type;

  cursor c1 is select ename  
    from emp;

begin

  fp := utl\_file.FileTypFopen('XYZ',  
    'File2', 'w');

  for i in c1

    loop

      utl\_file.putf (fp, '%s\n',  
                   i.ename);

    end loop;

  utl\_file.Fclose (fp);

end;

/

In oracle we can also write  
data into an OS file by using  
put\_line() procedure from utl\_  
file package.

This procedure also accept 2  
parameters.

Syntax :-

utl-

syntax

```
utl_file.put_line(filepointer varname,  
variablename);
```

> 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.empno || ' ' ||  
                      i.sal);
```

end loop;

```
utl_file.Fclose(Fp);
```

end;

/

## Reading Data From File

In oracle if you want to read data from file then we are using get\_line() procedure from utl\_file package. This procedure also accepts 2 parameters.

Before we are using this proc. then we must use read mode

within Fopen Function From  
utl\_file package

syntax

utl\_file.get\_line(filepointervarname,  
buffervariablename);

Q. Write a PL/SQL program which  
reads data from File1.txt by  
using utl\_file pkg & display that  
data

→ > declare  
 Fp utl\_file.file\_type;  
 Z varchar2(200);  
 begin  
 Fp := utl\_file.Fopen('XYZ', 'File1.txt'  
                   , 'R');  
 utl\_file.get\_line(Fp, Z);  
 dbms\_output.put\_line(Z);  
 utl\_file.Fclose(Fp);  
 end;  
 /

Q. Write a PL/SQL program which  
reads multiple data item from  
File2.txt by using utl\_file pkg  
& also display those multiple item.

```
>declare
  fp utl_file.file_type;
  z varchar2(200);
begin
  fp := utl_file.Fopen('XYZ', 'file2.txt',
                        'r');
loop
  exit when z is null;
  utl_file.get_line(fp, z);
  dbms_output.put_line(z);
end loop;
utl_file.Fclose(fp);
end;
/
t utl_file.Fclose(fp);
exception
when no_data_found then
  null;
end;
```

In oracle when we are trying to read multiple data items from an OS file by using utl\_file package then oracle server returns an error : **ora-1403** no data found . When control ~~not~~ reach end of file . To overcome this problem we must use no\_data\_found exception name within PL/SQL block .

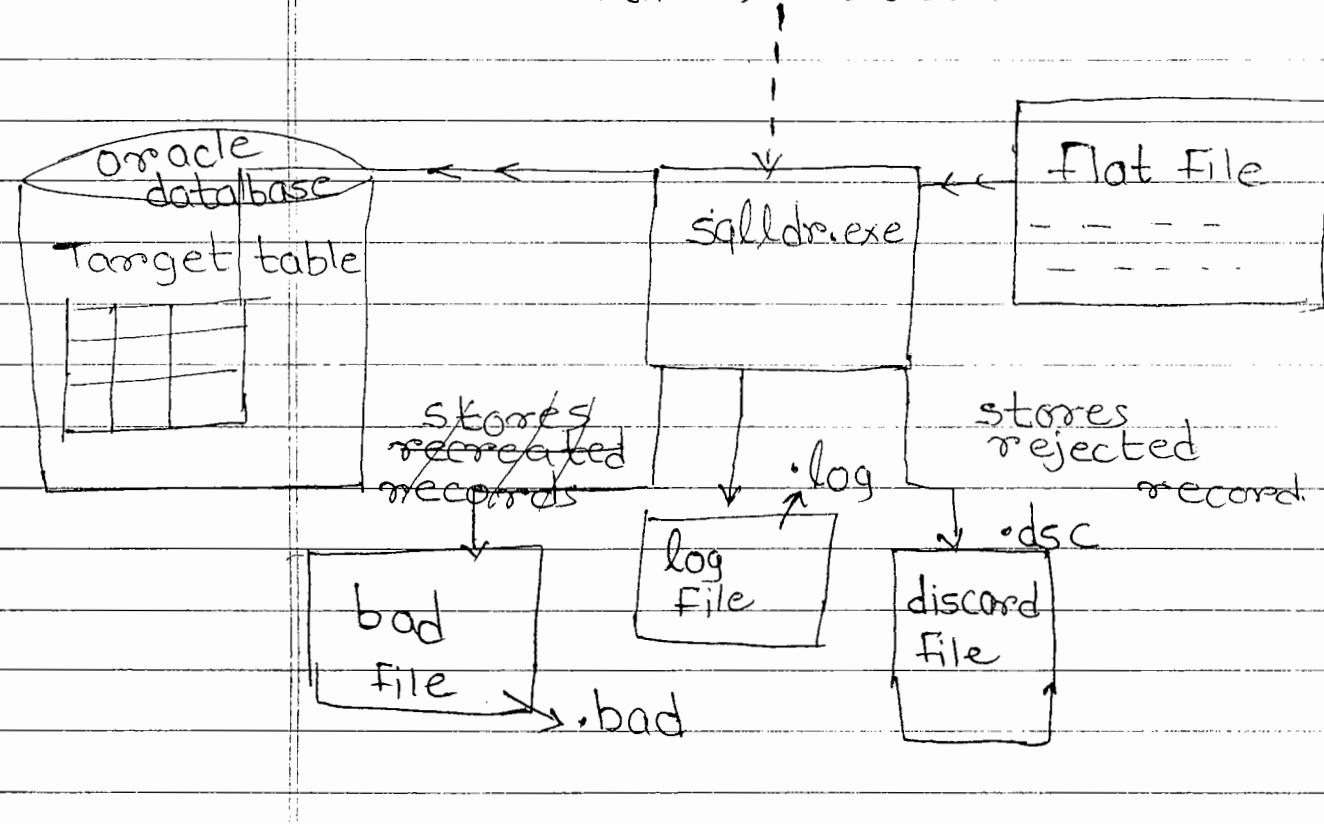
## SQL \* Loader.

SQL loader is a utility program which is used to transfer data from Flat file into oracle database.

SQL loader tool is also called as bulk loader.

SQL loader always executes control file this file extension is ".ctl". Based on type of flat file we are creating control file then only SQL loader take this ctrl file & then only transfer data from oracle into database.

control file (.ctl)



During this process sqlldr automatically creates log file as same name as control file. This log file stores all other files information and also stores loaded, rejected no. of records number and oracle error number, messages.

Based on some reasons, some records are rejected from database. Those rejected records are automatically stored in bad file, discard file.

Bad file stores rejected record based on datatype mismatch & business rule violation.

Discard file stores rejected record based on where condition within control file.

### Flat File :-

Flat file is a structured file which having any extension with no. of records.

Oracle having 2 types of Flat Files.

- 1) variable length record flat file.
- 2) Fixed length record flat file.

## Variable length record Flat file

A Flat File which having a delimiters is called variable length record.

e.g.

- \* 101, abc, 7000
- 102, xyz, 4000
- 103, jjj, 9000.

## Fixed Length record Flat file

A Flat File which does not have delimiter is called fixed length record Flat file.

e.g.

- 101abc7000
- 102xyz4000
- 103jjj9000

## Control File

Always sqlldr executes control File. Based on type of flat File e then we are creating control. And then submit this

control file to oracle server,  
then only sql loader transfer data  
from flat files into database.

Syntax :-

start → run → cmd → C:\

C:\> sqlldr userid = scott/tiger

control = path of ctrl file;

\* Creating control file Based on variable length record <sup>for</sup>

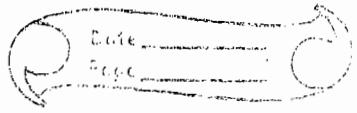
Always control file execution start with load data clause. After load data clause we must file path of Flat File by using "infile" clause.

Syntax

load data

infile :path of flat file'

We can also specify flat file data within control file itself in this case we must use '\*' in place of path of flat file



within infile clause & also use "begindata" clause in the above flat file data within control file itself.

e.g.

```
load B data  
infile *
```

...  
...

begindata

```
101, abc, 2000  
102, xyz, 4000
```

After specifying path of flat file then we must specify into table tablename clause.  
For loading data into oracle database.

In the above of into table tablename we can also use insert / truncate / replace / append clauses. If target table is an empty table then we are using insert clause. By default clause is ins insert.

Syntax: insert / truncate / append / replace  
into table tablename

After specifying target table  
then we are using following clauses  
based on type of data within flat file.  
These clauses are

- 1) Fields terminated by 'delimitername'
- 2) optionally enclosed by 'delimitername'
- 3) trailing nullcols

After specifying these clause we  
must specify target table columnname  
columns within parenthesis

Syntax  
control file (-ctl) :

load data  
infile 'path of flat file'  
insert / truncate / append

badfile 'path of bad file'  
discardfile 'path of discard file'

insert / truncate / append / replace  
into table tablename

Field terminated by 'delimitername'  
optionally enclosed by 'delimitername'  
trailing nullcols  
(columnname1, colname2, ....)

## 11 File1.txt

101, abc, 2000  
102, xyz, 3000  
103, kkk, 8000  
104, jjj, 9000

no. of

> create table target(emphno number(10),  
ename varchar2(10), sal number(10));

→ control file

load data  
infile 'C:\file1.txt'

insert  
into table target  
Fields terminated by ','  
(emphno, ename, sal)

save murali.ctl

→ Execution

C:\> sqlldr userid = scott/tiger

control = C:\murali.ctl

During this process sql loader creates log file as same name as control file. This log file stores all other files information & also store loaded, rejected record & also stores oracle error no., error messages.

e.g

load data

infile \*

insert into table target  
Fields terminated by ','

(empno, ename, sal)

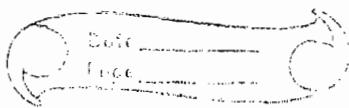
begin data

101, abc, 2000

102, xyz, 3000

constant, Filler clauses are used in control file

If you want to store default values in oracle database then we must use constant clause within control file. Whenever flat file having less no. of column & target table require more no. of column then we are using constant clause.



syntax

columnname constant 'defaultvalue'

IF you want to skip column from flatfile then we are using filler clause.

Syntax

columnname filler

e.g.

File1.txt

101, abc

102, xyz

>create table target ( empno number(1),  
loc varchar2(10) );

control file

load data

infile 'C:\File1.txt'

insert

into table target

fields terminated by ','

(empno, ename filler, loc constant  
'hyd')

### bad File :-

bad file stores rejected records  
this file extension is .bad. Bad file also automatically created as same name flat file name.

We can also create bad file explicitly by specifying filename within bad file clause.

### Syntax :-

bad file

Bad file stores rejected record based on following reason

- 1) datatype mismatch
- 2) Business rule violation

- 1) datatype mismatch.

File1.txt

101, abc

102, xyz

103, kkk

104, nnn

rt  
> create table target (empno number(10),  
ename varchar2(10));

control file

load data

infile 'C:\file1.txt'

insert

into table target

Fields terminated by, ','

(empno, ename)

File1.bad

'102', xyz

'103', kkk

2) Business rule violation

File1.txt

101, abc, 1000

102, xyz, 4000

103, kkk, 2000

104, nnn, 9000

> create table target (empno

number(10), ename, varchar2(10'),

sal number(10) check(sal > 2000))

control file

load data

infile 'C:\file.txt'

insert  
into table target  
Fields terminated by ','  
(empno, ename, sal)

~~File1.txt~~ → ~~discarded~~ File1.bad  
101, abc, 1000  
103, kkk, 2000

→ In flat file records trailing fields are null values then those record also automatically rejected & those records are stored in bad file if you want to store those null value record also into database then we are using "trailing nullcols" clause within control file.

e.g.

101, abc, 1000  
102, xyz,  
103, kkk  
104, nnn, 9000

>create table target (empno number(10)  
ename varchar2(10), sal number(10));

control file  
load data  
infile 'C:\File.txt'

Date: \_\_\_\_\_

insert  
into table target  
Fields terminated by ','  
trailing nullcols  
(empro, ename, sal)

recnum :-

Whenever we are using  
recnum clause then oracle server  
automatically assign numbers to  
loaded, rejected no. of record

syntax

columnname recnum

e.g. File1.txt

101, abc

'102', xyz

'103', klc

104, nnn

> create table target (empro number,  
ename varchar2(10), rno number(10);

control file

load data

infile ' c:\file1.txt'

insert

into table target

fields terminated by ','

(empno, ename, rno recnum)

0/p

empno ename rno

101 xyz 1

104 mnk 4

SRI RAGHAVENDRA XEROX

Software Languages Material Available

Beside Bangalore Ayyagar Bakery,

Opp. CDAC, Balkampet Road,

Ameerpet, Hyderabad.

### \* Discard File

Discard file also stores rejected record based on when condition within control file. "when" cond must be specified "into table tablename" clause.

syntax :

when condition

Generally bad file <sup>is</sup> created automatically as same name as flat file whereas discard |

In this case we must specify filename within discard file clause within control file

Discarded file extension is .dsc

Syntax :

discordfile ' path of discarded file '

file1.txt

101, abc , 10

102, xyz , 20

103, kkk, 10

104, nnn , 10

105, yyy, 30

>create table target (empno number(10),  
ename varchar2(10), deptno number(10));

control file

load data

infile 'c:\file1.txt'

discordfile ' c:\file3.dsc '

insert

into table target when deptno = '10'

fields terminated by ','

(empno, ename, deptno)

File3.dsc (discarded file)

102, xyz, 20 } discarded

105, yyy, 30 } records

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 <sup>are allow</sup> we can't use 'and'.

#### \* Functions used in control File.

We can also use oracle predefined function within control file. In this case we must specify function's functionality within " " & also we must use ':' colon operator in front of the columnname within function functionality.

Syntax :

columnname " Functionname (:columnname)"

File1.txt

101, abc, m

102, xyz, F

103, KIK, m

104, nnn, F

> create table target (empno number(10),  
ename varchar2(10), gender varchar2(10))

load data  
infile 'c:\file1.txt'  
insert  
into table target  
Fields terminated by ','  
(empno, ename, gender  
"decode(:gender,'m','male','f','female'))

\* dates used in control file:-

method 1 : (using date type)

method 2 : (using to\_date() Function)

method 1

syntax

columnname date "FlatFile date format"

e.g.

File1.txt

101, abc, 130507

102, xyz, 250809

103, kkk, 171203

ber(10),  
am2(10));  
>create table target (empno number(10),  
ename varchar2(10), col3 date);

### control file

load data  
infile 'c:\File1.txt'  
insert  
into table target  
Fields terminated by ','  
(empno, ename, col3 date "DDMMYY")

### method 2 (using to\_date() function)

load data  
infile 'c:\File.txt'  
insert  
into table target  
Fields terminated by ','  
(empno, ename, col3 "to\_date  
(:col3, 'DD/MM/YY')")

\* Sequence used in control file

### Syntax

columnname " sequencename.nextval "



File1.txt

101

102

103

104

105

>create sequence S1;

> create table target (sno number(0));

control file

load data

infile 'c:\File1.txt'

insert

into table target

Fields terminated by ','

(sno "S1.nextval")

O/P

1

2

3

4

5

## \* Creating Control File for Fixed length record Flat File

When flat file does not have delimiter those flat file are called fixed length record. When we are using fixed length record flat file then must use position clause within control file. In this position clause we must specify starting, ending position of every field by using colon operator. Alongwith position clause we must use SQL loader datatype SQL loader having 3 datatypes

- 1) integer external
- 2) decimal external
- 3) char

### Syntax

columnname position (startingpos : endingpos)  
sqlloaderdatatypes.

→ Whenever we are using "F" then we are not allowed to use SQL Loader datatype in place of SQL Loader datatype we must use "F" functionality within "" (double quotes).

columnname position (startingpos, endingpos)  
"Functionname (:columnname)"

Date 3/12/15

e.g:-

File1.txt

101 abc 3000

102 xyz 2000

103 zzz 9000

104 yyy 7000

number

f  
5n

> create table target (empno  
number(10), ename varchar2(10),  
sal number(10));

control file

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 )

sno	value	Time	col1	col2	col3
	default value 50	HH:MM: ss	col1/100	upper( )	to_date( )

create sequence s1;

load data

infile \*

insert

into table target

```
( sno "s1.nextval" , value constant '50' ,
    time "to_char(sysdate,'HH:MM:ss')",
    {col1 position(01:05) integer external
/100 ":"col1/100"),
    col2 position(06:11) "upper(:col2)" ,
    col3 position(12:17) "to_date(:col3,
        'DDMMYY'))"
```

begin data

10000 aaaaaaa050103

20000 bbbbbbb180905

In oracle we are not allowed to use : colon & also we are not allowed to use : new , : old qualifiers in front of sysdate , user functions.



Using sql loader we can also transfer no. of flat file data into single target table by specifying no. of infile clauses within control file. Using sql loader we can also transfer single flat file data no. of target table by specifying no. of into table tablename clauses.

But using sql loader we are not allowed to transfer, if resources as diff database data, or if resources as combination of flat file & database data.

In this case we are using database tools.

Specifi  
cation

## Triggers

Trigger is also same as stored procedure & also it will automatically invoked whenever DML oprn performed on table or view.

Oracle having 2 types of triggers

- 1) Statement Level
- 2) Row Level.

In statement level trigger, trigger body is executed only once per DML stmt, whereas in row level triggers ; trigger body is executed for each row per DML stmt.

Trigger also having 2 parts

- 1) Trigger specification
- 2) Trigger Body.

Syntax

Specification	Trigger Timing	Trigger Event
	create / or replace trigger / triggername before / after	insert / update / delete on tablename

[For each row] → for Row level  
[when condition] Trigger

[declare]

variable declaration, cursor,  
user defined exception

begin

[exception]

end ;

DIFF b/w statement level &  
row level.

statement level trigger

> create or replace trigger t1

after update on emp

begin

dbms\_out.put\_line('statement  
level');

end;

/

ex-1

> update emp set sal = sal + 100  
where deptno = 10;

statement level

3 rows updated.

> update emp set sal = sal + 100  
where deptno = 90;  
statement level  
0 rows updated.

### Row level Trigger

> create or replace trigger t1  
after update on emp  
for each row  
begin  
dbms\_output.put\_line('row level');  
end;  
/

ex.1

> update emp set sal = sal + 100 where  
deptno = 10;  
row level  
row level  
row level  
3 rows updated.

> update emp set sal = sal + 100  
where deptno = 10;  
0 rows updated.

## Row level Trigger

In row level trigger, body is executed for each row in DML statement. That's why we must use "for each row" clause in trigger specification. And also DML transactional values are internally automatically stored in two rollback segment qualifiers. These are :old, :new.

### Syntax

:old.columnname

:new.columnname

These qualifiers are also called as record type variables.

These qualifiers are used in trigger specification or trigger body

- When we are using these qualifiers in trigger specification then we are not allowed to use : in front of qualifier name

### Syntax

old.columnname

new.columnname

e.g.

### Insertion

>insert into emp values(1, 'murali', 2000);

	empno	ename	sal
:new	1	murali	sal

:new.ename.

### update

>update emp set sal=5000 where  
empno = 1;

	empno	ename	sal
:old	1	murali	2000

:old.sal

	empno	ename	sal
:new	1	murali	5000

:new.sal

### Deletion

>delete from emp where empno=1;

	empno	ename	sal
:old	1	murali	5000

:old.sal

insert update delete

:new	✓	✓	✗
:old	✗	✗	✓

Q Write a PL/SQL row level trigger  
on emp table whenever user  
inserting sal then that sal  
should be more than 5000.

→ create or replace trigger tij1

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

```
>insert into emp(empno,ename,sal)  
values (1,'murali',2000);
```

Ora-20123: salary should be  
above 5000

```
>insert into emp(empno,ename,sal)  
values (1,'murali',9000);  
1 row created.
```

er Q. Write a PL/SQL row level trigger in  
Following table not allow to insert  
duplicate data i.e. trigger acts like  
primary key.

→ Test

sno
10
10
10
20
20
30

> create or replace trigger t1  
before insert on test  
for each row  
declare  
cursor c1 is select \* from test;  
begin  
for i in c1  
loop  
if i.sno = :sno then  
raise application error (-20789,  
'we cannot insert duplicate data');  
end if;  
end loop;  
end;

## Testing

>insert into test values(10);

ora-20789 : We cannot insert  
duplicate values.

error

>insert into test values(90);

1 row created.

Q. Write PL/SQL row level trigger  
in above table not to accept  
duplicate data without using cursor.

→ create or replace trigger t1  
before insert on emp 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 cannot insert duplicate data');

elsif v\_count = 0 then

dbms\_output.put\_line ('ur  
record inserted');

end if;

end;

/

Testing:

> insert into test values(10);  
error: ora- 20123 : we cannot insert duplicate data

>insert into test values(1);  
1 record inserted.

Q. Write PL/SQL row level trigger on emp implement Following business rule.

Company does not allow any bonus to job as analyst.

→ create or replace trigger tw2  
before insert on emp  
For each row  
~~when (new.job = 'ANALYST')~~  
when (new.job = 'ANALYST')  
begin  
IF :new.comm is not null then  
:new.comm := null;  
end if;  
end;

Testing

>insert into emp(empno,ename,job,  
comm) values (1,'murali','ANALYST',  
300);

*Q.* Write a PL/SQL row level trigger on emp table whenever user modifying sal then automatically display old sal, new sal, sal difference.

→ create or replace trigger th1 before update on emp

for each row

declare

x number(10);

begin

x := :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('salary diff  
is :'|| :x);

end;

/

*Testing*

>update emp set sal=sal+100  
where ename = 'KING';

*Q.*

*Q.*

→

Q. Write a PL/SQL row level trigger on ~~emp~~<sup>dept</sup> whenever user modify dept no in dept table then automatically those modification are applied on emp table.

→ create or replace trigger t2  
before after update on dept

For each row

declare

begin cursor c1 is select \* from emp;

begin

update emp set deptno = :new.deptno  
where deptno = :old.deptno;  
end;

/

Testing

>update dept set deptno = 1  
where deptno = 10;

Q. Write PL/SQL row level trigger on ~~dept~~<sup>emp</sup> table whenever user deleting data from emp table then automatically deleted data stored in another table.

→ >create table target (empno number(10),  
ename varchar2(10), sal number(10));

create or replace trigger th2  
after delete on emp

for each row

begin

insert into target values

(:old.empno, :old.ename, :old.sal);  
end;

Testing

> delete from emp where sal > 3000;  
> select \* from target;

row level trigger Application

## 1) auditing a column

In all databases whenever we are modifying column data then those transaction are stored in another table is called auditing a column.

In oracle if you want to implement auditing column appln then we must use update of clauses in trigger specification of row level triggers

Syntax :-

update of columnname

2)

e.g.

>create or replace trigger t1  
after update of sal on emp  
for each row

begin

insert into target values(:old.empno,  
:old.sal, :old.:new.sal, sysdate,  
user);

end;

/

:000;

>create table target (empno number(10),  
oldsalary number(10), newsalary number(10),  
colt date, username varchar2(10));

>update emp set sal=sal+100  
where deptno = 10;

→ In oracle we are not allowed to  
use :old, :new qualifier in front  
of the sysdate, user functions.

## 2) Auto increment :-

In all databases generating  
primary key values automatically is  
called auto-increment. In oracle if  
you want to implement auto increment  
concept then we are using sequence  
database obj, row level trigger i.e.

Date \_\_\_\_\_  
Page \_\_\_\_\_

we must create sequence in  
sql & then use that sequence  
in PL/SQL row level trigger

e.g.

> create table test (sno number(10)  
primary key , name varchar2(10));

> create sequence s1  
starts with 1;

> create or replace trigger t1  
before insert on test  
For each row  
begin  
select s1.nextval into :new.sno  
from dual;  
end;  
/

Testing

> insert into test (names)  
values ('& name');

→ Oracle 11g introduced variable  
assignment concept when we are  
using sequences in PL/SQL block

In this case we are not  
allowed to use dual table,  
select into clause.

syntax:-

begin

varname := sequencename.nextval ;  
end ;

e.g.

>create or replace trigger tv1  
before insert on test  
For each row

begin

:new.sno := s1.nextval ;  
end ;

/

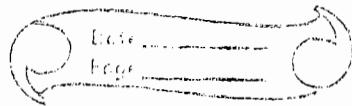
sno \* Generating alpha-numeric data as  
primary key in auto-increment  
concept

e.g.

>create table test (sno varchar2(10)  
primary key , name varchar2(10));

>create sequence s1 ;

k >create or replace trigger tv2  
before insert on test  
For each row  
begin



```
select 'ABC' || lpad(s1.nextval, 10, '0')  
into :new.sno from dual;  
end;  
/
```

testing

```
>insert into test(name) values  
(&name),
```

Enter value for name: murali

```
>/
```

"

"

:xyz

```
>/
```

"

"

:ABCper

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

sno	name
ABC0000000001	murali
ABC0000000002	xyz
ABC000 000 0003	pqr

```
>create or replace trigger tr2
```

[after] insert on test

For each row

begin

select s1.nextval into :new.sno

From dual;

end;

ora- :error → cannot change new  
value for this trigger type.

'0')

## Trigger Timing (before / after) :-

Oracle DML triggers having  
2 timing points.

- 1) before
- 2) after timing

Whenever we are using before timing oracle server executes trigger code before DML stmts are executed in database i.e. whenever we are using DML stmts those transaction value are not affected directly on db; before that one those transaction value are affected on trigger.

Whenever we are inserting data using :new qualifiers & also when we are assigning & modifying value then we must use before timing otherwise oracle server returns error "cannot change new value for this trigger type"

begin

:new.columnname := value;

end;

Whenever we are using after timing DML transactional values are 1<sup>st</sup> affected in database then only trigger code is executed. Generally in all db one table transaction value are affected in another table then only we are using after timing.

\* In oracle if you want to set particular s cell values at in a table, then not also we are using :new qualifier \*

Q. Write PL/SQL row level trigger on emp table whenever user inserting data into ename column then user inserted data automatically converted into uppercase in ename col.

→ >create or replace trigger t1  
before insert on emp  
For each row  
begin  
:new.ename := upper(:new.ename);  
end;  
/

## Statement Level Trigger

In stmt level trigger, trigger body is executed only once per DML stmt. Statement level trigger does not have old, new qualifier, It does not have "For each row" clause.

In all databases, if you want to develop time component based appln through trigger then we are using stmt level trigger.

Q. Write PL/SQL trigger on emp table not to perform DML oprn on saturday, sunday.

→ Create or replace trigger st1  
before insert or update or delete on  
emp

begin

if to\_char(sysdate, 'DY') in ('SAT', 'SUN')  
then  
raise application\_error ('cannot perform  
DML operation on ' || to\_char(sysdate, 'DAY'))  
end;

+

if to\_char(sysdate, 'DY') in ('SAT', 'SUN')  
then  
raise application\_error (-20123, 'cannot

not perform DML oprn');  
end;

/

→ In all databases we are not  
allowed to use when clause in  
statement level trigger.

create or replace trigger tui  
before insert or update or delete  
on emp

For each row

when (to\_char(sysdate, 'DY') in  
( 'SUN', 'SAT' )

begin

raise application\_error ('cannot  
perform DML oprn');

end;

/

→ In all databases statement level  
trigger performance is very high  
compared to row level trigger.

Q. Write a PL/SQL statement level trigger.  
For emp not to perform DML  
oprn in last day of month

→ create or replace trigger tui  
before insert or update or delete  
on emp

begin

```
if sysdate = last_day(sysdate) then  
raise application_error(-20123, 'cannot  
perform DML opn on last day of month');  
end if;  
end;
```

/

## Triggering Events

In oracle if you want to define different oprn on no. of tables within trigger body then we are using triggering event these are inserting, updating, deleting clause. These are also called as trigger predicate clauses. These clauses are used in either in statement or row level triggers.

### Syntax

```
if inserting then  
— stmts;  
—
```

```
elsif updating then  
stmts;
```

```
elsif deleting then  
stmts;  
end if;
```

Q. Write a PLSQL stmt level trigger  
on emp table not to perform  
any DML oprn in any days using  
triggering events.

→ create or replace trigger tr1  
before insert or update or  
delete on emp

```
begin
  if inserting then
    raise application_error (-20143, 'cannot perform
      insertion');
  elsif updating then
    raise application_error (-20345, 'cannot
      perform updation');
  elsif deleting then
    raise application_error (-20123, 'we
      cannot perform deletion');
  end if;
end;
```

1

o/p →

e.g.

```
>create table target (msg varchar2(20));
>create or replace trigger tr1
  after insert or update or delete
  on emp
```

Q.

declare

z varchar2(20);

begin

```

;en      if inserting then
;g        z := 'rows inserted';
;g      elsif deleting then
;g        z := 'rows deleted';
;g      elsif updating then
;g        z := 'rows updated';
;g      end if;
;g      insert into target values(z);
;g    end;
Form
/
Testing
>insert into emp(empno) values(1);
1 row created
7 insert into emp(empno) values(2);

>update emp set sal=sal+100 where
deptno=10;
3 rows updated
>select * from target;
o/p → msg
row inserted
row inserted
row updated
r2(20); )

```

- Q Write PL/SQL row level trigger on emp table whenever user inserting data on emp table those inserted record are stored in another table when user updating those transaction are stored in another table whenever user

deleting data then those record  
are stored in another table.

> create table z1 (empno number(10),  
ename varchar2(10), sal number(10));

> create table z2 (,, , , );

> create table z3 (,, , , );

> create or replace trigger tr1  
after insert or update or delete  
on emp

for each row

if inserting then

insert into z1 values (:new.empno,  
:new.ename, :new.sal);

elsif updating then

insert into z2 values (:new.empno,  
:new.ename, :old.sal);

elsif deleting then

insert into z3 values (:old.empno,  
:old.ename, :old.sal);

end if;

end;

/

Testing

> delete from emp where  
sal > 2000;

> select \* from z3;

## Execution order of trigger

- r(10); 1) before statement level
- ; 2) before row level
- ; 3) after row level
- ; 4) after statement level.

e.g.

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

```
>create or replace trigger th1
```

after insert on test

for each row

begin

```
dbms_output.put_line('after row level');
```

end;

/

```
>create or replace trigger th2
```

before insert on test

begin

```
dbms_output.put_line('before insert  
statement level');
```

end;

```
>create or replace trigger th3
```

after insert on test

begin

```
dbms_output.put_line('after statement  
level');
```

end;

>create or replace trigger th4  
before insert on test  
for each row  
begin  
dbms\_output.put\_line('before  
row level');  
end;  
/

Testing

>insert into test values (10);  
o/p

before statement level

before row level

after row level

after statement level.

#### \* Compound Trigger (Oracle 11g):-

Oracle 11g introduced compound trigger.

Compound trigger allows diff. blocks within a trigger to be executed at a different timing points. Compound trigger also having global declaration section same like packages.

## Syntax

create or replace trigger triggername  
for insert / update / delete on tablename

compound trigger

global variable declaration ;

Types declaration ;

Local Subprogram implementation ;

before statement is

begin

—

end [before statement ~~is~~] ;

before each row is

begin

end [before each row] ;

after each row is

begin

—

end [after each row] ;

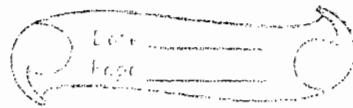
after statement is

begin

—

end [after statement] ;

end;



Q. Write PL/SQL compound trigger  
to display default execution  
order of table \*

→ >create or replace trigger tk1  
for insert on test

compound trigger  
before statement is

begin

dbms.output.put\_line('before  
statement level');

end;

before each row is

begin

dbms.output.put\_line('before  
row level');

end;

after each row is

begin

dbms.output.put\_line('after  
row level');

end;

after statement is

begin

dbms.output.put\_line('after  
statement level');

end;

end;

## \* Follows clause (11g)

of Whenever we are using same level trigger on same table then we can not control execution order to overcome this problem oracle 11g introduced follows clause in trigger specification.

Using Follows clause we can control execution order of trigger explicitly when we are using same level of trigger on same table.

Using Follows we are providing guarantee execution order of trigger explicitly

### Syntax:

create or replace trigger triggername  
before insert/update/delete on  
tablename

[For each row]

[When condition]

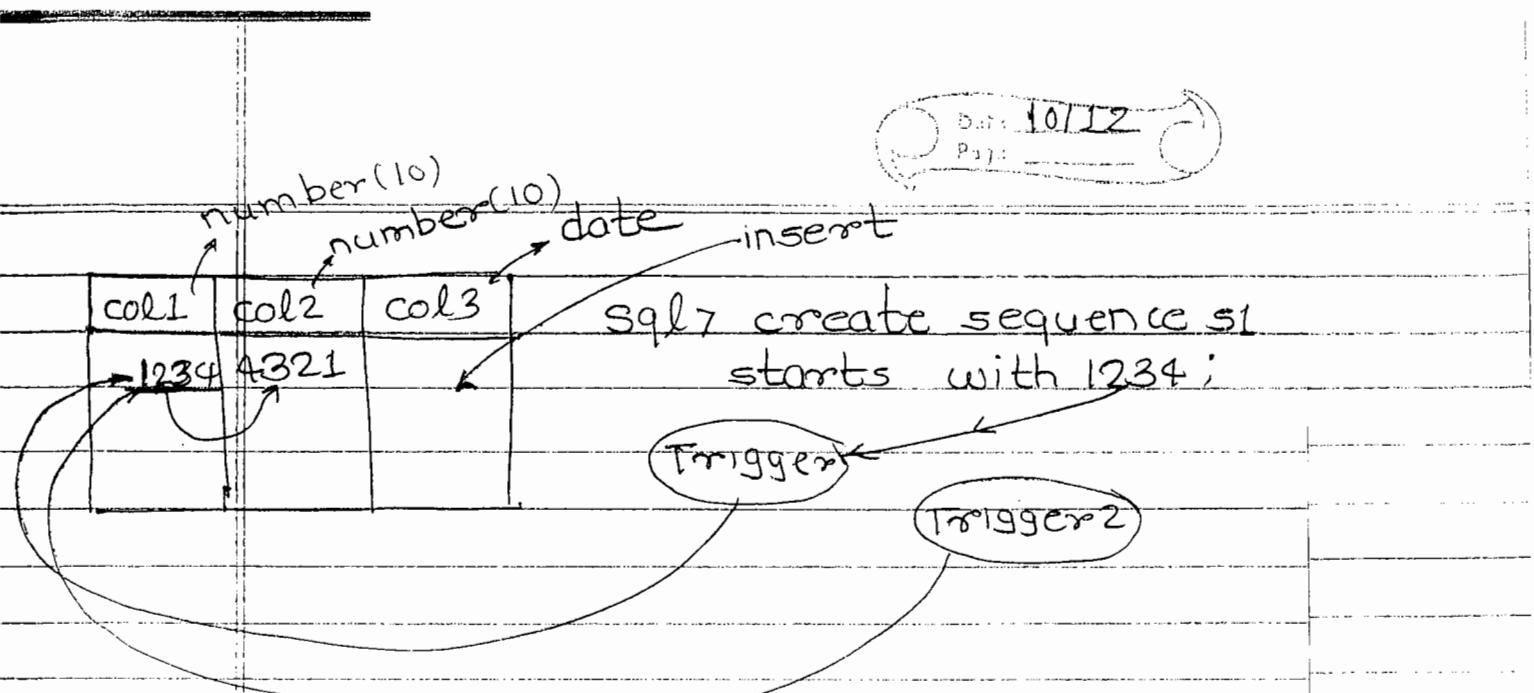
[Follows anothertriggername]

[declare]

begin

....

end;

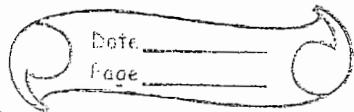


>create table test (col1 number(10),  
col2 number(10) , col3 date);

>create sequence s1  
start with 1234;

> create or replace trigger tk1  
before insert on test  
for each row  
begin  
select s1.nextval() into :new.col  
from dual;  
dbms\_output.put\_line ('Trigger1  
fired');  
end;

> create or replace trigger tk2  
before insert on test  
for each row  
begin



```
s1      select reverse(to_char(:new.col1))  
;       into :new.col2 from dual;  
       dbms_output.put_line('Trigger2 Fired');  
       end;  
       /
```

### Testing

```
>insert into test(col3) values(sysdate);  
trigger 2 fired  
trigger 1 fired
```

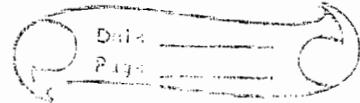
```
>select * from test;  
col1    col2    col3  
1234    4321   10-DEC-15
```

### Solution:-

By using follows (11g)

```
>create or replace trigger tk1  
ol  
insert into  
before insert on test  
For each row  
begin  
:new.col1 := s1.nextval();  
dbms_output.put_line('Trigger 1  
created');  
end;  
/
```

Before firing same level triggers are executed in reverse order of creation.



> create or replace trigger tk2  
before insert on test

For each row

cell trans. | Follows tk1

Formation in

trigger

requires select

into clause

From dual.

begin  
→ select reverse (to char(:new.col1))  
into :new.col2 from dual;  
dbms.output.put\_line('Trigger 2  
created');

end;

/

Testing

> insert into test(col3) values (sysdate);

O/p

trigger 1 Fired

trigger 2 Fired

100% mutation  
error

> select \* from test

col1	col2	col3
1234	4321	10-DEC-15

Q. Write a PL/SQL trigger whenever user deleting rows from emp table then automatically display remaining no. records number at bottom of delete statement.

→ create or replace trigger tk1  
after delete on emp  
declare

z number(10);

```
begin
    select count(*) into z
    from emp;
    dbms_output.put_line('z ' || ' ' || '
        number of remaining rows');
end;
/
```

e.g.

> create or replace trigger tki  
✓ after delete on emp

✓ For each row

declare

✓ count number(10);

begin

select count(\*) into v\_count

from emp ✓

dbms\_output.put\_line(v\_count);

end;

/

Trigger Created

Testing

> delete from emp where empno=2;

[ora-04091]: table SCOTT.EMP is

mutating.

## Mutating Error

If a row level trigger based on table that trigger body cannot read data from same table and also we cannot perform DML oprn on same table. If you are trying this oracle server returns an error ora-4091 : mut table is mutating.

Mutating error is a runtime error , occurred in row level triggers only. \*

In oracle whenever we are using statement level trigger the DML transaction are automatically committed into database. When we try to read this committed data by using triggers then oracle server does not give any error whereas in row level trigger DML transaction are not committed in db when we try to read these data by using triggers then database servers returns mutating error.

Thats why mutating error occurred in row level trigger only.

To avoid mutating error then we are using autonomous transaction in trigger. But autonomous transaction always return previous results.

```
>create or replace trigger tv1  
after delete on emp  
for each row  
declare  
    pragma autonomous_transaction;  
    v_count number(10);  
begin  
    select count(*) into v_count from emp;  
    commit;  
    dbms_output.put_line(v_count);  
end;
```

### Testing

```
>delete from emp where empno=7560;
```

14

### \* When Condition

When condition are allowed to use `o` in row level trigger only.

It is used in trigger specification.

It always return boolean value either true or false. When cond<sup>n</sup> is true then only trigger body is executed.

When condition must be  
a SQL expr<sup>n</sup>.

### syntax

when ( condition )

→ In oracle we are not allowed  
to use ; in front of new qualifiers  
within when clause.

c.g.

create or replace trigger tgl

after insert on emp

For each row

when (new.empno = 1)

~~be~~ begin

dbms.output.put\_line ('trigger body fired');

end;

/

### Testing

>insert into emp(empno) value(1);

trigger body fired

1 row created.

>insert into emp(empno) values(2);

1 row created.

## Calling Procedure in Trigger

In oracle we can also call a procedure in trigger by using call statement.

Syntax :-

create or replace trigger triggername  
before/after insert/update/delete on  
tablename

call procedurename

/

E.g.

>create table target (totsal number(10));

>create or replace procedure p1

is

v-sal number(20);

begin

delete from target;

select sum(sal) into v-sal from emp;

end;

/

>create or replace trigger tr1

after insert or update or delete on  
emp

call p1

/

>update emp set sal=sal-200  
where deptno =10;

>select \* from target;  
TotSal  
51650

## System Trigger / DDL Trigger

All databases also supports triggers on DDL command these type of triggers are also called as DDL trigger / System trigger These trigger are created by database administrator.

In oracle we are creating triggers in two level  
1) Schema level  
2) Database level

### Syntax :

→  
>create or replace trigger triggername  
before/after create/drop/alter/  
truncate / rename  
on username.schema / database  
declare  
--  
begin  
end;

## database level

> conn sys as sysdba/sys ;

> create or replace trigger tv4

after create on database

begin

dbms\_output.put\_line ('database object  
created');

end;

/

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

database object created

Table created.

In oracle if you want to write  
DDL trigger then we are using  
predefined trigger event attr. FN.

Q. PLSQL trigger on scott schema  
not to delete emp table.

→ > create or replace trigger tv1

before drop on scott.schema

begin

if ora\_dict\_obj\_name = 'EMP' and  
ora\_dict\_obj\_TYPE = 'TABLE' then  
raise application\_error (-20123, 'we  
cannot drop emp table');

end if;

end;



## Testing

>drop table emp;



Ora-20123 : we cannot drop table.

Oracle having 12 types of trigger based on statement level / row level, before, after, I / U / D.

Oracle also supports system triggers same like other db's.

Oracle also supports 'instead of' triggers on views.

## Enable / Disable Trigger

enable / disable a single trigger

Syntax: alter trigger triggername  
enable / disable;

enable / disable all triggers in a table

## Syntax

alter table tablename  
enable / disable all trigger;

e.g.

>alter table emp disable all triggers;

→ Oracle 11g introduced enable / disable clauses in trigger specification.

### syntax

>create or replace trigger triggername  
before / after insert / update / delete  
on tablename

[For each row]

[When condition]

[Follows clause & another trigger name]

[enable / disable]

[declare]

.....

begin

.....

end ;

/

All trigger information stored under  
[user-trigger] data dictionary

>desc user\_trigger;

We can also drop trigger using  
>drop trigger triggername ;

~~Note~~ where current of, for update clauses are used in cursor

(or) update, delete statements are used in cursors (or)

### cursor locking mechanism

\* Q. Write PL/SQL cursor program to update sal of emp in emp table by using following cond's

if job = clerk then increment sal -> 100

if job = salesman then decre. sal -> 200

→ 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 ~~the~~ emp set sal = sal + 100

where empno = i.empno;

elsif i.job = 'SALESMAN' then

update emp set sal = sal - 200;

end if

end loop;

close c1;

end;

)

7

Whenever we are using DML stmts then Db server internally uses default locks. But if you want to perform locks before update or delete stmts then we are using explicit locking mechanism through cursors.

If you want to perform locks explicitly then we are using For update clause within cursor select stmts.

syntax :

cursor cursorname is select \* from tablename where for update;

When we are specifying for update clause also when we are opening <sup>cursor</sup> then only oracle server establishes locks by default these locks exclusive locks.

where current of

clause uniquely identifying a record because it uses internally rowid. where current of clause is used in update, delete stmt. only.

syntax :-

update tablename set columnname = newvalue  
where current of cursorname;

syntax

delete from tablename where  
current of cursorname;

Whenever we are using where  
current of clause then we must  
use for update clause in cursor  
select stmt otherwise oracle server  
returns an error.

\*\*> where current of clause is used  
to update or delete & latestly  
fetched row from the cursor

After processing data by using  
where current of, for update  
then we must release the locks  
by using commit within PL/SQL  
block.

>create table test (sno <sup>varchar2</sup> number(10),  
sal number(10));

ename	sal
a	1000
b	2000
c	3000
b	4000
c	5000

>commit;

without using where current of clause:

> declare

cursor c1 is select \* from test;

begin

for i in c1

loop

update test set sal = sal + 1000

where ename = i.ename;

end loop;

end;

/

> select \* from test;

ename	sal
-------	-----

a	3000
---	------

b	4000
---	------

a	5000
---	------

b	6000
---	------

c	6000
---	------

Solution (using where current of clause)

Whenever resource table having duplicate data then if you want to update or delete record using cursor then we must use where current of clause otherwise it returns wrong result.

> rollback;

```

> declare
  cursor c1 is select * from test
  for update;
begin
  for i in c1
  loop
    update test set sal = sal + 1000
    where current of c1;
  end loop;
  commit;
end;
  
```

ename	sal
a	2000
b	3000
a	4000
b	5000
c	6000

**SRI RAGHAVENDRA XEROX**  
 Software Languages Material Available  
 Beside Bangalore Ayyagar Bakery,  
 Opp CDAC, Baikampet Road,  
 Ameerpet, Hyderabad.

- Q. Write a PL/SQL cursor program by using cursor locking mechanism increment 5th employee sal to 100 in emp.

→ > declare

```

cursor c1 is select * from emp
for update;
begin
  for i in c1
  loop
    
```

```

) if c1%rowcount = 5 then
;it update emp set sal = sal +100
where current of ;
end if;
end loop; commit;
end;
/

```

Q. Write PL/SQL cursor program which update column c based on Following cond<sup>n</sup> from following table.

- 1) For first 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  $c=a/b$

> create table test (a number(10),  
 b number(10), c number(10));  
> insert into test values(5, 4, null);  
:

> commit

> select \* from test

a	b	c
5	4	
5	4	
5	4	
5	4	

```
>declare  
cursor c1 is select * from test  
For update;  
begin  
For i in c1  
loop  
if c1%rowcount = 1 then  
update test set c = a+b where  
current of c1;  
elsif c1%rowcount = 2 then  
update test set c = a-b where  
current of c1;  
elsif c1%rowcount = 3 then  
update test set c = a*b where  
current of c1;  
elsif c1%rowcount = 4 then  
update test set c = a/b where  
current of c1;  
end if;  
end loop; commit;  
else end;
```

## LOBs (Large Objects)

Oracle 8.0 introduced LOBS, LOBS are predefined datatypes which is used to store large amount of data, images in databases.

In oracle if you want to store more than 2000 bytes alpha numeric data then we are using varchar2 datatype. This datatype stores upto 4000 byte.

If you want to store more than 4000 bytes of alpha numeric data then we are using long datatype.

Long datatype stores upto 2Gb of data but there can be only one long column for a table & also it cannot create primary key on long col.

To overcome this problem, Oracle 8.0 introduced 'clob' datatype

Syntax

columnname long

e.g.

```
>create table test (col1 long, col2 long)
error: a table may contain only one
column of type Long.
```

>create table test (col1 long primary key);

key column cannot be of long datatype

IF you want to store binary data then we are using raw datatype  
It stores upto 2000 bytes of binary data.

syntax :

columnname raw(maxsize)

IF you want to store more than 2000 bytes of binary data then we are using long raw datatype.

It stores upto 2gb data but there can be only long raw column for table.

To overcome this problem oracle 8.0 introduced blob datatype

syntax

{columnname long raw}

>create table test (col1 raw(2000));

>insert into test ('1010100');

✓

or  
>create table test1 (col1 long raw);  
>desc test1;

x  
All database system having 2 types  
of LOBS

- arry  
ry  
type  
ry  
1) internal large objects  
2) external large object

### Internal Large object

Internal large objs are stored  
in within database. Oracle having 2  
types of internal large objs.

- i) clob (character large object)  
ii) blob (binary large object)

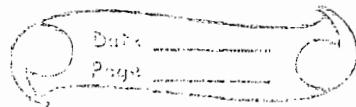
cle  
syntax

columnname clob

columnname blob

### External Large object

are stored in outside of  
database i.e. these objs are stored in  
os file.



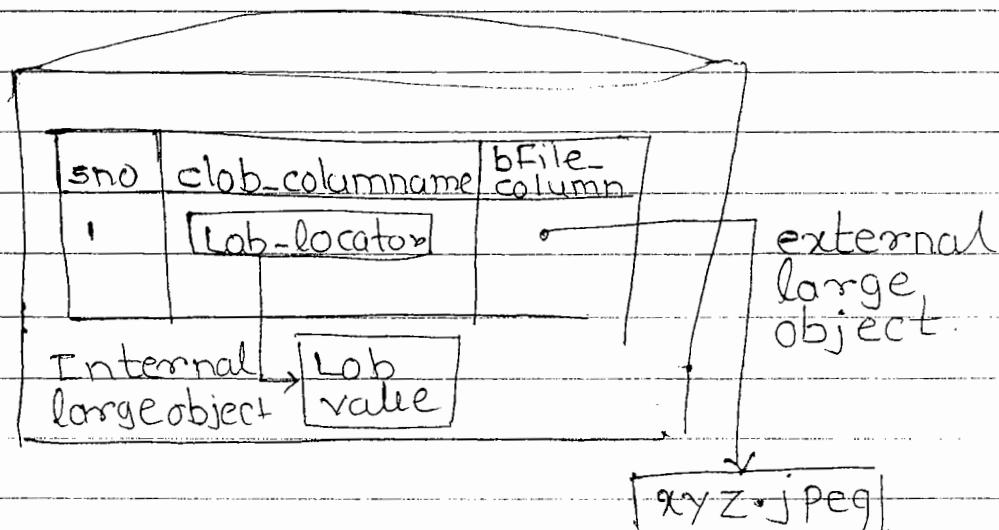
Diff b/w long, lob datatype

Long

Lob

- 1) can contain upto 2gb data      1) can contain upto 4gb data
- 2) A table does not contain more than one long column      2) A table can contain more than one lob column
- 3) Subquery cannot select a long datatype column.      3) subquery can select lob datatype column

LOB locator :-



The data in lob-column is not stored in the row along with the other column of the row. Instead of this one a locator is stored in database & actual data is stored elsewhere.

Locator is similar to a pointer & this pointer points a location where data is actually stored.

In internal LOB data is stored in within database. In internal lob this pointer is also called as lob locator.

lumn

empty\_clob, empty\_blob:

These predefine functions are part of the SQL DML, these functions are used to initialize lob locator into empty locator in insert, update command.

→ Before we are writing data into lob column by using either oci or dbms\_lob package then we must initialize lob locator into empty locator by using these function.

Difference b/w null, empty

>create table test (sno number(10),  
col2 clob);

>insert into test values(1, 'abc');

> " " (2, empty\_clob());

> " " (3, null);

> select \* from test

sno	col2
1	abc
2	
3	

> select \* from test where col2  
is not null ;

<u>sno</u>	<u>col2</u>
1	abc
2	

> select \* from test where col2 is null ;

<u>sno</u>	<u>col2</u>
3	

Storing large amount of data into  
clob column / storing an image  
into blob column.

Step1) Create an alias directory  
related to physical directory by  
using following syntax.

> create or replace directory  
directoryname as 'path' ;

> conn sys as sysdba ;  
sys

> grant create any directory to scott ;

> conn scott/tiger ;

> create or replace directory XYZ as  
'C:\';

Step2 :- create a table by using LOB column

>create table tablename (col1 clob,  
col2 blob);

ii) Step 3 :- Write PL/SQL block to store large amount of data or image into lob column by using dbms\_lob package

i) before we are opening a file we must declare lob variable in declare section of PL/SQL block.

syntax :-

varname clob;  
varname blob;  
varname bfile;

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

iii) before we are using dbms\_lob pkg then we must initialize lob locator into empty by using empty\_clob()  
empty\_blob() Functions. And also by using returning into clauses we can store pointer into appropriate lob variable.

This insert stmt must be used in executable section of PL/SQL block.

syntax

insert into tablename values  
(empty\_clob()) returning into  
lobcolumnname into lobvarname;

iii)

Before we are using dbms\_lob pkg  
then we must specify aliasdirectory  
with actualfilename by using  
bfilename() function.

This f<sup>n</sup> accepts two parameters &  
returns bfile datatype.

syntax

bfilename := bfilename('aliasdirnum',  
'filename')

v

iv) Before we are loading data into  
lobcolumn then we must open the  
file using Fileopen procedure  
from dbms\_lob pkg.

Syntax :

dbms\_lob.Fileopen(bfilename);

v) After opening the file we can load data into lob column by using `fo! "loadFromFile"` procedure from `"dbms.lob"` package.

This proc. accepts 3 parameters.

syntax

`dbms.lob.LoadFromFile(lobvarname,  
bFilevarname, length of bFile);`

→ IF you want to return length of bfile then we are using `getlength()` function from `dbms.lob` pkg.

syntax

`dbms.lob.getlength(bFilevarname);`

vi) After loading data then we must close the file by using `"Fileclose"` procedure from `dbms.lob` pkg.

`dbms.lob.Fileclose(bFilevarname);`

```
>create table test ( sno number(10),  
    col2 clob );
```

```
> 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.fileopen(v_bfile);
```

```
        dbms_lob.loadFromFile(v_clob, v_bfile,  
            dbms_lob.getlength);
```

```
        dbms_lob.fileclose(v_bfile);
```

```
    end;
```

```
/
```

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

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

→ We can also store large amount of data / images into oracle database by using bfile datatype directly. But we cannot display this data in sqlt environment

e.g.

```
>create table test (col1 bfile);
```

```
>insert into test values (bfilename  
    ('xyz', 'file1.txt'));
```

),  
> select \* from test;  
error: column or attribute type can  
not be displayed.

— 0x0 —

## Dynamic SQL

Oracle 7.1 introduced dynamic SQL.  
In dynamic SQL always SQL stmt  
are executed at runtime. But  
these SQL stmt must be specified  
by ~~or~~ using execute  
immediate clause within PL/SQL block.

Here SQL stmt must be  
specified within ''.

syntax

begin

execute immediate 'SQL stmt';

end;

Generally we are not  
allowed to use DDL, DCL stmt  
in PL/SQL, if you want to use these  
stmt in PL/SQL then we must use  
dynamic SQL constructs.

Date \_\_\_\_\_  
Page \_\_\_\_\_

e.g.

begin

i execute immediate 'create table  
k1(sno number(10));'

end;



Q. Write PL/SQL program to create role.

5

→ begin  
execute immediat 'create role r1';  
end;



→ conn sys as sysdba  
sys

Retriev Data from dynamic SQL Stmt

Through into clauses we can  
retriv data from dynamic SQL  
stmt.

Q. Write dynamic sql program which is  
used to display max sal from emp  
→ declare

v\_maxsal number(10);

begin

execute immediate 'select max(sal)  
from emp' into v\_maxsal ;

```
dbms_output.put_line(v_maxsal);
end;
```

→ We can also use bulk collect clause in dynamic SQL stmt.

Q. Write dynamic sql program which retrieves all emp name from emp table & stored it into index by table & display content

→ 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;

## Passing value into dynamic sql stmt

IF you want to pass values into dynamic sql stmt then we must use using clause i.e. within sql stmt we are not allowed to pass direct value in place of that we must use placeholder.

In dynamic sql placeholders represented by using : operator.

After dynamic sql stmt we must specify actual value through using clause. These placeholders are used in insert, delete, select, update stmts.

Q. Write dynamic sql program which is used to insert record into dept

→ begin 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;

/

Whenever we are using, 'using', 'into' clauses in a single dynamic sql stmt at a time then always into clause precedes than using clause.

Q. Write dynamic sql For passing deptno from dept table display dname, loc from dept table.

→ declare

v\_deptno number(10) := &deptno ;

begin

execute immediate 'select

v\_dname varchar2(10) ;

v\_loc varchar2(10) ;

begin

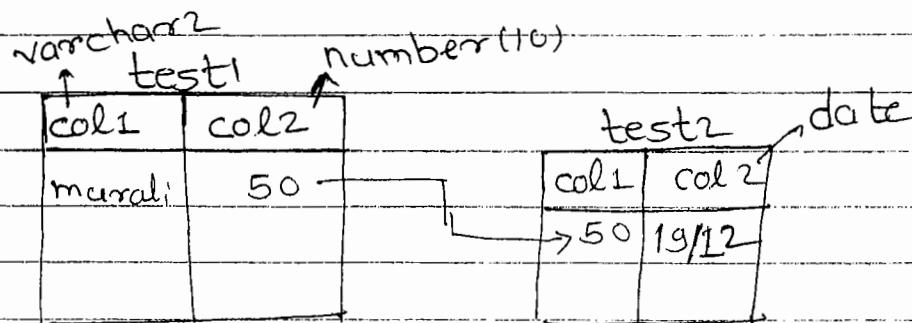
execute immediate ' select dname ,  
loc From dept where deptno = :1 '  
into v\_dname, v\_loc using v\_deptno ;

dbms\_output.put\_line (v\_dname || ' ' ||  
v\_loc );

end;

1

## Avoiding mutating error by using compound trigger



> create table test1 (col1 varchar2(10),  
col2 number(10));

> create table test2 (col1 number(10),  
col2 date);

> create or replace trigger tv1  
after insert on test1

For each row

declare

id number(10);

begin

select col2 into id From test1

where col2 = :new.col2;

insert into test2 values (id, sysdate);

end;

/

Trigger created

> insert into test1 values ('murali', 70);

Ora-04091: mutating error.

To avoid mutating  
by using packaged global  
variable along with after row level,  
after stmt level trigger.

- 1) Create packaged global variable.
- 2) Create after row level trigger
- 3) Create statement level trigger.

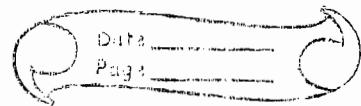
### solution

```
>create or replace package pc1  
10),  
is  
    id number(10);  
;
```

```
>create or replace trigger tr1  
after insert on test1  
For each row  
begin  
    pc1.id := :new.col2;  
end;  
/
```

date);

```
>create or replace trigger tr2  
after insert on test1  
begin  
    insert into test2 values (pc1.id,  
                           sysdate);  
end;
```



oracle 11g onwards we can also avoid mutating error by using compound trigger. Because compound trigger having global dm declaration & also compound trigger allows diff. block within a trigger to be executed at diff. timing points.

- Q. Write PL/SQL trigger which avoids mutating error based on above tables.

→

create or replace trigger tr1

for insert on test1

compound trigger

id number(10);

after each row is

begin

id := :new.col2;

end;

after statement is

begin

insert into test2 values(id,  
sysdate);

end; after statement is;

end

## Member Subprogram

Oracle 8.0 introduced Object technology. Oracle having an Object type which is an user defined by represent diff. datatype into simple unit.

> create or replace type typename ~~object~~ as object(attribute datatype(size), ...);

Once we are creating an object type then we must instantiate object in declare section of PL/SQL block.

### syntax

varname objecttypename;

### assigning values into object type

After instantiating object then we must assign values into object type by using constructor ~~by using~~ in executable section of PL/SQL block

### syntax

varname := objecttypename(actual value);

>create or replace type employee as  
object (stno number(10), sname  
varchar2(10));

>1

instantiating object

> declare  
obj1 employee;  
begin  
obj1 := employee(101, 'murali');  
dbms\_output.put\_line(obj1.stno || ' ' ||  
'employee name' || ' ' || obj1.sname);  
end;  
/

Oracle having two types of  
member subprogram.

- 1) member procedure
- 2) member Function.

Member subprogram are  
declared in object specification  
whereas member ~~sub~~ subprogram  
are implemented in object body.

Object type having 2 parts

- 1) Object specification
- 2) Object body.

## Syntax

(10))

create or replace type typename  
as object (attribute datatype(sized),  
member procedure declaration,  
member function declaration);

## Object body

>create or replace type body typename  
as

member procedure implementation;  
member function implementation;

e.g.

>create or replace type circle as object  
(# number, member procedure P1,  
member function f1 return number);

1

>create or replace type body circle  
as

member procedure p1

is

begin

dbms.output.put\_line('my proc');

end p1;

member function f1 return number

is

begin

Date \_\_\_\_\_  
Page \_\_\_\_\_

```
    return 2 * 3.14 * r;
end FL;
end;
```

Execution by PL/SQL client

```
>declare
  c1 circle;
begin
  c1 := circle(8);
  dbms_output.put_line(c1.fl);
end;
/
```

With

sql> with function FL(a number)

return number

is

begin

return a\*a;

end;

? select FL(4) From dual.

\* Oracle 12 c introduced accessible by clause in stored procedure specification which is used to call given procedure into specified within specified procedure through accessible by clause.

syntax

>create or replace procedure procedure-name (formal parameter)  
accessible by (procedurename1,  
procedurename2, ... )

is / as

...

begin

...

end;

'

We are not allowed execute accessible by procedure.

>create or replace procedure p1  
accessible by (P3)

is

begin

dbms\_output.put\_line ('myproc');

end p1;

/

>create or replace procedure p2

is

begin

p1;

end p2;

/

error

>create or replace procedure p3

is

begin

p1;

end;

/

o/p: myproc.

## Oracle 11g Features

- 1) Oracle 11g introduced read only table by using alter command.

In this table we are not allowed to perform DML oprn.

syntax:

alter table tablename read only;

alter table tablename read write;

- 2) Oracle 11g introduced virtual column which is used to store stored expression directly in oracle Db by using "generated always as" clause.

- 3) Oracle 11g introduced simple integer datatype in PL/SQL

Simple integer datatype internally having not null thats why we must assign value when we declare declaring variable in declare section.

varname simple integer := value;

e.g.

```
> declare
  a simple_integer := 50;
begin
dbms_output.put_line(a);
end;
/
```

Simple integer datatype performance  
is very high compared to pls-integer  
datatype.

```
declare
a pls_integer;
begin
a:=50;
```

```
dbms_output.put_line(a);
end;
/
```

SRI RAGHAVENDRA XEROX  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.

- 4) Oracle 11g introduced continue statement  
in PL/SQL loop.

It is also same as C lang  
continue statement. It sends control  
to beginning of the loop.

syntax

continue;

```

> begin
  for i in 1 .. 10
  loop
    if i=5 then
      continue;
    end;
    dbms_output.put_line(i);
  end loop;
  !

```

OLP

1

SRI RAGHAVENDRA XEROX  
 Software Languages Material Available  
 Beside Bangalore Ayyagar Baker  
 Opp. CDAC, Balkampet Road,  
 Ameerpet, Hyderabad.

6

7

8

9

10

per

- 5) Oracle 11g introduced pivot() function which is used to display aggregate function value in tabular format & also convert rows into column.
- 6) Oracle 11g introduced Follows clause in triggers, ~~&~~ compound triggers.
- 7) Variable assignment concept when we are sequences in PLSQL. in this case we are not allowed use dual table.
- 8) Oracle 11g introduced enable/disable clauses within trigger specification.
- 9) Oracle 11g introduced named, mixed notation when subprogram is executed by using select stmt.

**SRI RAGHAVENDRA XEROX**  
Software Languages Material Available  
Beside Bangalore Ayyagar Bakery,  
Opp. CDAC, Balkampet Road,  
Ameerpet, Hyderabad.