

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

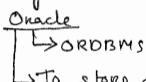
Lakst Version 12 C (Cloud Computing)

Version release in 2013 june 29

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

Q) what is database?

Ans Data base is nothing but structured data



XEROX ACADEMY

- 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

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 organisation

- 1) books & papers

- 2) flat files

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

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.

→ In databases every transaction internally having 4 properties. 17/07/2015
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.

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

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 does not provide indexing mechanism.

Organisation suffering from flatfile mechanism to store data or 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) :-

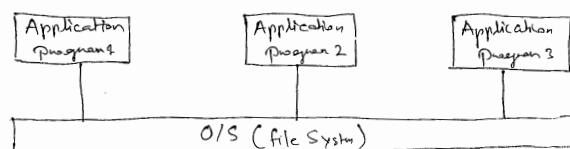
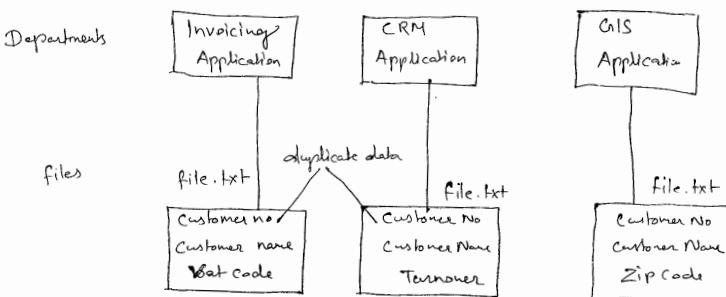
It is a collection of programs (S/W) written to manage database.

Ex:- Oracle, teradata, sqlite, mysql, sybase, ingres, informix, db2, sql server, ...;

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

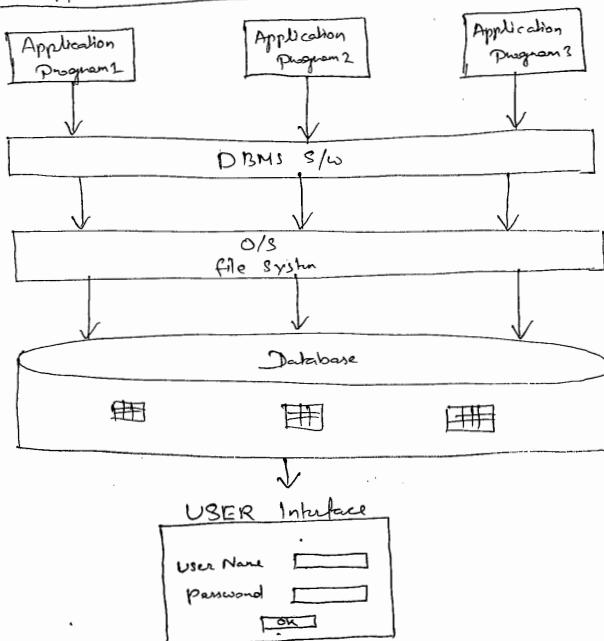
Flat file Approach for Data Management

e.g:-



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

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

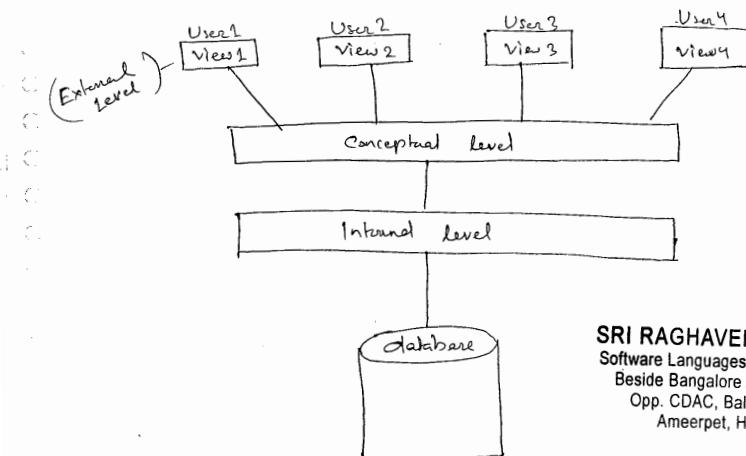
ANSI- (American National Standard Institute) has established 3 level architecture for dbms.

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)



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 ^{data} independence.
- 2) Physical Data independence.

1) Logical Data Independence!-

Changes in the conceptual level donot 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 donot 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 doesnot 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 defines relationship between tables by specifying

ORACLE DBMS

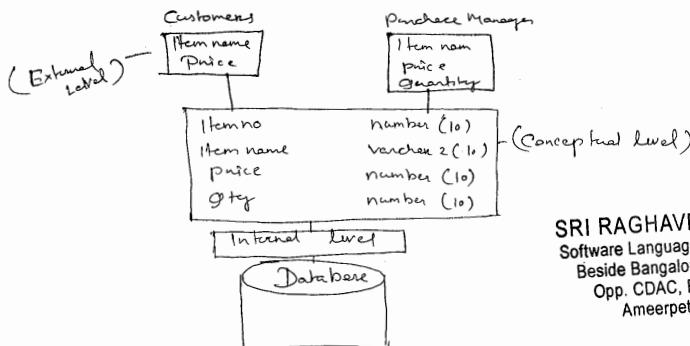
Views, Synonyms

Table

Index, Clusters

External Level :-

- External level describes user's 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 data base administrator creating a views & then only those views given to no. of users.

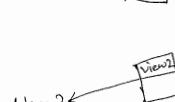


SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

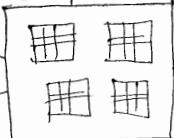
DBMS Architecture

21/07/2015

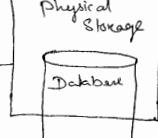
External Level



Conceptual Level



Internal 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 :-

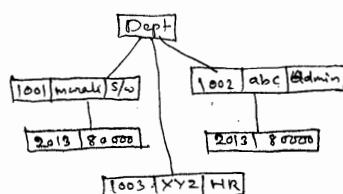
→ In hierarchical data model organizes data in tree like structure. This hierarchy is also called parent, child hierarchy. In hierarchical data model also data is represented in the format of records and also return type is also same as table in relational data model.

→ Hierarchical data model is implemented 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 language.

22/07/2015

Hierarchical datamodel



Relational Data Model

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

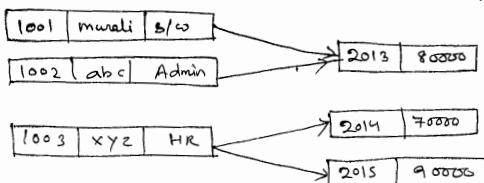
foreign key		
empno	year	Amount
1001	2013	80000
1002	2013	80000
1003	2014	80000

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.

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

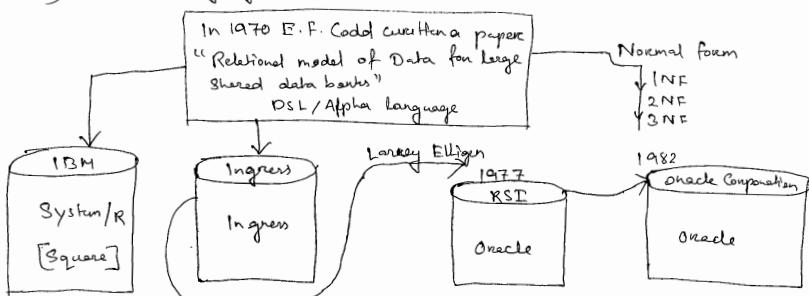
→ Relational data model mainly consists of 3 components. They are

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 these three language required.

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

→ In 1977 Larry Ellison, Bob Miner, Ed Oates founded SOD (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 varchar 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

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

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

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

Oracle 9i → 2001 : → merge statement

- ansi joins (or) all 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 10G (grid technology) → 2003 : → recydelen

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

Oracle 11G → 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 clauses used in pl/sql trigger specification.
- Sequence are used in pl/sql without using dual table.
- named, mixed notations used in a set operation excepted casting
Select Statement.

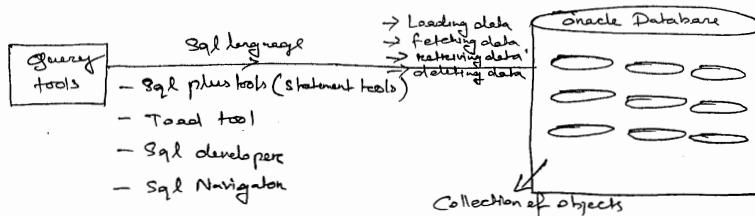
Oracle 12C → 2013 : → invisible columns

- identity columns

- 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 nonprocedural 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

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 :-

Select * from table name;

e.g:- Select * from dept;

Select * from emp;

SQL

1) Data Definition Language (DDL) :-

→ create

→ alter

→ drop

→ truncate

→ rename (Oracle 9i)

2) Data Manipulation Language (DML) :-

→ insert

→ update

→ delete

→ merge (Oracle 9i)

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

→ select

4) Transactions Control language (TCL) :-

→ Commit

→ Rollback

Data types

24/07/2015

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

- 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.

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

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! :- Here number (7,2)

P-S = $\Rightarrow 7-2 = 5$ 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:- When ever we are using number (P,S) format then we are try to insert more no. of digits after after decimal point than the specified scale the oracle server did not give error - In this case oracle server only automatically

number (p) :-

→ It is used to store fixed numbers

Syntax :-

Column name 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) charc :-

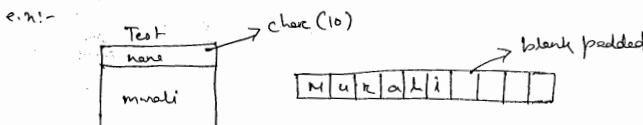
It is used to stored fixed lengthed Alpha numeric data in bytes maximum limit is upto 2000 bytes by default character datatype having one byte

Syntax :-

Column name char (size)

25/07/2015

When ever 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.



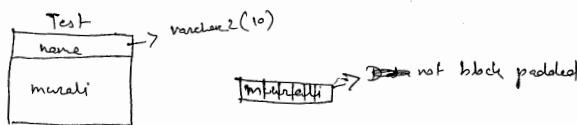
Varchar 2 (max size) :-

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 data type size Specified in varchar2 datatype then oracle server doesn't add null character.



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

i) 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

i/ Create :-

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

Creating a table:-

Syntax:-

Create table tablename (column1 datatype, . . .)

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;
```

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 the table.

method 1 → In oracle 1 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 1 if you want to drop single or multiple columns with using parenthesis then we are using following syntax.

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

SQL> desc first;

R.H:-

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:-

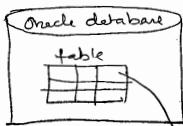
drop objecttype objectname;

- ① drop table tablename;
- ② drop view viewname;

dropping a table

before Oracle 10g

SQL> drop table tablename;



Permanently removed/deleted

Oracle 10g, 11g, 12c [Enterprise Edition]

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

got it back the table:-

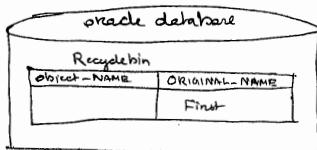
SQL> Flashback table first to before drop;

Testing !-

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

Recyclebin !-

Recyclebin is an predefined readonly table which is used to stores dropped 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 normal recyclebin we can also drop objects from
recyclebin.

Drop

drop a single table at a time from recyclebin

Syntax !:- purge table tablename;

drop all table from recyclebin

Syntax !:- purge recyclebin;

Recyclebin purged.

y/Truncate !-

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

SQL>

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.

Renameing a table

Syntax :-

```
rename oldtablename to newtablename;
```

B.x!:-

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

Renameing a column (oracle 9i):-

Syntax :-

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

Ex:-

```
SQL> alter table emp rename column empho to sno;
SQL> select * from emp;
```

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

28/07/2015

Data Manipulation Language (DML)!:-

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

- 1) insert
- 2) update
- 3) delete

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

method 1:-

Syntax:-

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

e.g!:- SQL> create table first (sno number (10), name varchar 2 (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] msg

e.g!:- 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.g:- 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.g:- 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	

Updated.

SQL> Select * from first;

SNo	NAME	ADDRESS
1	murali	
2	sadhu	
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
get 1 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 of 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 then deleted rows by using rollback (cancel out using commit)
- When ever we are using truncate command then total data is permanently deleted from a table because truncate is a odd command of also all odd transactions are automatically committed. that's why after truncating data we cannot get it back then 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

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

- ① select (all cols) of all rows → where condition
- ② Select all cols of (particular) rows
- ③ Select (particular) cols & all rows
↳ Col1, Col2, ...
- ④ Select particular cols of 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 existingtablename 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 operations

Q) write a query to display ename, sal, annualsal from emp table?

SQL> Select ename, sal, sal*12 annualsal from emp;

ename sal annualsal
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;

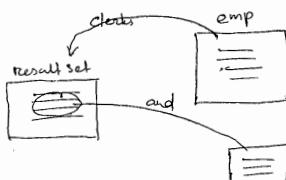
Q) write a query to display the employees who are getting more than 2000 salary from emp table?

SQL> select * from emp where sal > 2000;

30/07/2015

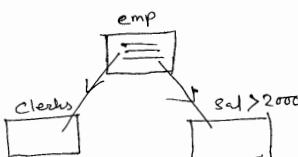
→ If you want to specify multiple condition on where clause then we are using logical operators. When ever we are using logical operator AND database server filter the data from resultset

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.

Q) write a query to display the employees who are belongs to the departments no.s 20, 30, 50, 70, 90 from emp table

L ~~SQL~~ Select * from emp where deptno = 20 or deptno = 30 or deptno = 50 or deptno = 70 or deptno = 90 ; /

Special Operations :-

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

i) In :- It is used to print pick the value one by one from list of values.
we can also use in operator in place of are operator when we are try to
retrieve multiple values from a single column in this case 'in' operator performance
is very high compare to or operation

Syntax! -

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

P. 281 -

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

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

5 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.

Q) write a query to display ename, sal, commision, sal+commision 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 overcome 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

Ex:- 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

`sal + nvl(comm, 0)`

$\Rightarrow 1400 + \underline{\text{nvl(null, 0)}}$

$\Rightarrow 1400 + 0$

$\Rightarrow 1400$

NVL2() :- Oracle 9i introduced nvl2(). NVL2(2) accepts 3 parameter

Syntax:-

`nvl2(expr1, expr2, expr3)`

→ If expr1 is null then it will returns expression3 otherwise it returns exp2

Ex:- 1) `nvl2(null, 20, 30) => 30`

2) `nvl2(10, 20, 30) => 20`

31/07/2015

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 are 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 operators

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;

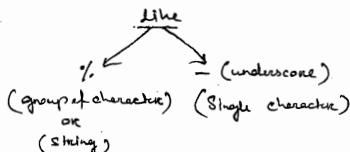
Q) write a query to display the employees whose commission is not null from emp table.

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

Like!:-

01/08/2015

→ 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 one 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
KILLER } 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 is 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_MITH');

SQL> select * from emp;

SQL> select * from emp where ename like 'S_%';

ENAME

S_MITH

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 ^{the} database server automatically escapes special meaning of the escape character wild card character in the case along with escape function we must use any escape character there 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%_%' 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~~ second letter contains -- underscore underscore those employee retrieve from emp table by using the operator ?

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

SQL> select * from emp;

SQL> select * from emp; where ename like 'S@_@_%' Escape '@';

S--MITH

@ → 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 emps table by using like operator ?

Concatenation operators :- (1) → 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 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) Userdefined 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 rnd (null, 70) from dual; → Output - 70

SQL> Select rnd (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

`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!:- say select mod (10,5) from dual; \rightarrow output - 0.

* round (m,n):-

It rounds even floated value number m based on n.

e.x!:- `SQL> Select round (1.8) from dual;` \rightarrow output - 2.

`SQL> Select round (1.23456,3) from dual`

output
1.235

execution!-

Step 1:- 1.234

out of 100
50 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;` \rightarrow 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;` \rightarrow output - 1300

Execution!-

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

Step 2:- 1200

1
1300

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

trunc (m,n):-

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

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

07/08/2015

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.

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

e.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 uppercase.

e.x!:- 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;

Q) write 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.

e.x!:- 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 + e if counts no. of characters including spaces.

e.x!:- SQL> Select length ('AB_{space}CD') from dual; → output 5 characters including space.

Substr () :- It will extract portion of the string with in 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 - CDEFG Character after FG.

Syntax!:-

numbers

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

Ans :- 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().

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

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

Ans :- 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 clauses.

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 with in given string.

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

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

SQL> Select instr('ABCDEFGHIJKLMNCDTGM', 'CD') from dual;
Output → 3

SQL> Select instr('ABCDEF²GHIJKLMNCDTGM', 'CD', -6, 2) from dual;
Output → 3

SQL> Select instr('ABCDEF²GHIJKLMNCDTGM', 'CD', -5, 2) from dual;
Output → 9

numbers

Syntax:-

Column name ('.') , Searchchar Position, number of)

Lpad:-

It will pad 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
or
Stringname , total length , 'filled character');
→ number datatype`

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
or
String name , {set of character});`

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

output → MISSTHSS

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

output → JOB Ltrim (job)

CLERK	LERK
SALESMAN	ALESMAN
MANAGER	ANAGER

Rtrim():-

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

output → SMISSTH

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>
xy dx a	xy dia

Syntax:- translate (string1, string2, string3)

06/03/2015

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

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

e.x:- SQL> Select job, replace (job, ' SALESMAN', ' MARKETING') from emp;
SQL> Select replace ('SSMISSTHSS', 'S') from dual;

* Q) write a query which counts number of occurrences of the character E within the given string 'sleep' by using replace function?

e.g.) Select length ('sleep') - length (replace ('sleep', 'e')) from dual;
output - 2

Concat (Str1, Str2):-

It is used to concat given two strings.

e.g.) 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.g.) 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)


e.g.) select add_months (sysdate, 1) from dual;
output - 06-SEP-15

g.) 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')

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 ✗
- 4) Date1 - Date2 ✓

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

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;`

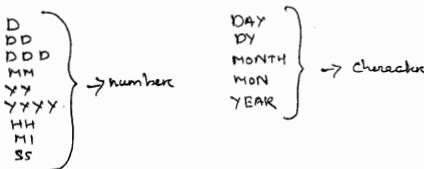
07/08/2015

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.

SQL> select to_char(sysdate, 'day') from dual;
output - friday.



SQL> select to_char(sysdate, 'DY') from dual;
output - FRI

SQL> select to_char(sysdate, 'D') from dual;
output - 5 (Day of the week (Sunday->1, monday->2, ...))

SQL> select to_char(sysdate, 'DD') from dual;
output -

SQL> select to_char(sysdate, 'DDD') from dual;

SQL> select to_char(sysdate, 'DDSPTH') from dual;
output - SEVENTH SP - SPELLED OUT

SQL> select to_char(sysdate, '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.g:- 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 value must match with the default date format return value otherwise oracle server return an error. To overcome this problem, we can use a second parameter

`SQL> select to_date ('15/06/05') from dual;`

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

`ex-SQL> Select to_date ('23/06/05', 'DD-MM-YY') from dual;`

Output - 23-JUN-05

`ex-SQL> Select to_date ('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

(ok)

`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) :-

whenver 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()

`ex-SQL> Select to_char (sysdate, 'DD/MM/YY') from dual;`

Output - 08/AUGUST/15

`SQL> select to_char (sysdate, 'DD/FMMONTH/YY') from dual;`

Output - 08/AUGUST/15

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';

(2) write 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';

error - showing four digit yyyy

SQL> select * from emp where to_char(hiredate, 'YYYY') from emp;

(3) Note! - In oracle culumnes we are passing date string into predefined oracle date functions (ADD_MONTHS(), Last_day(), ...) then oracle 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

ex1!- (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 (coll date);

SQL> insert into test values (sysdate);

SQL> Select * from test;

Output → 08-AUG-15

SQL> Select * from test;

Output

15 AUG 07

SQL> Select last_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'));

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

round(), trunc() functions used in dates:-

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 converts one date to the other 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 inherently 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?

Ans:- SQL> insert into emp (empno, ename, hiredate) values(1, 'manu', 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')

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 year's 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 operate over no. of values in a column &
returns a single value.

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

r.n!:- SQL> Select max(sal) from emp;
output → 5400

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;

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

e.g. 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?

e.g. SQL> select job, count(*) from emp group by job;

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

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

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

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

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

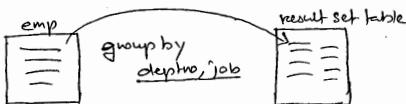
Solution

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 form based on specified columns after shown by class. i.e.

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

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



Note:- In all database when we are try to display group function with another column then database servers returns errors to over come this problem we must use group by class.

ex:- 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;`

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 ?

`= SQL> select deptno, count(*) from emp group by deptno where count(*) > 3;`

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 where as if you want to restrict groups after group by then we must using 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

output →

DEPTNO	SUM(SAL)
30	10300
20	11875

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

SQL > select to_char(hiredate, 'YYYY'), count(*)/from emp group by

output →

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

 to_char(hiredate, 'YYYY')
→ Here it will displayed year

SQL > select to_char(hiredate, 'YYYY') "year", count(*) from emp group by
to_char(hiredate, 'YYYY') having count(*) >

output →

Year	Count(*)
1987	2
1981	10

to_char(hiredate, 'YYYY') having count(*) >

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(*) > ;

output →

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 see using to keywords there are asc, desc by default order by clause having ascending

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 shorting databases on first column specified within order by clause.

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

DEPTNO	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;

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);

13/08/2015

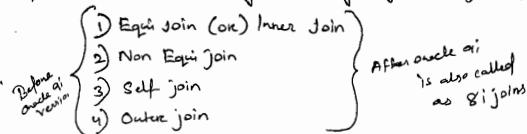
e.x:- SQL> Select ename, sum(sal) from emp group by rollup (ename);

JOINS

→ Joins are used to retrieve data from multiple tables

→ In all databases when we are joining 'n' tables then we must use ' $n-1$ ' joins/conditions.

→ Oracle server having following types of joined joins



→ These 4 joins are also called as 8i joins

8i 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.

→ 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 conditioned columns must belongs to same data type.

→ Whenever tables having common columns then only we are using equal join & also these common columns must belongs to same data type.

Syntax:-

Select col1, col2 from table name 1, table name 2.

where table name 1. common column name = table name 2. common column name.

Solution :-

```
SQL> select e.name, sal, dept.deptno, dname, loc
  from emp, dept
 where emp.deptno = dept.deptno;
```

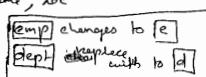
Note :- In all databases for avoiding feature ambiguity then we are specifying tablename along with every column using a (.) dot operator within select list.

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;



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 equi joins ?

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 with in 8i joins but in 9i joins we can also use either logical operator AND or where conditions.

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

Ex:- \Rightarrow 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:- \Rightarrow 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:- \Rightarrow 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 equi join 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 equi join these join is also called as oracle 8i outer join.

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

Condition

Ex:- \Rightarrow 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
 to OPERATIONS BOSTON

Non Equi Join:-

Based on other than equality condition ($<$, $>$, \leq , \geq , $=$, between, \dots) we are

SQL> Insert into t1 values (...);
SQL> Select * from t1;

DEPTNO	ENAME
10	a
20	b

SQL> Create table t2 (deptno number (10), dname varchar2 (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

17/02/2015

Note:- In oracle when table doesn't have common column then also we are using non eqi join but in this case ^{one} table column value lies between another table two columns.

For ex:- 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;

Same data types.

In self join we must create alias names for the table with this from clause these alias names are also called as reference names these alias names must be different names these alias names behaves like a exact table when 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?

\hookrightarrow 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?

\hookrightarrow SQL> Select e1.ename "employee", ~~e2.ename~~ "managers", 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?

\hookrightarrow 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

9) joins (or) ansj joins :-

- 1) Inner join
- 2) Left outer join
- 3) Right outer join

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

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

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

18/08/2015

2) 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.g:- SQL> Select * from t2;

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

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;

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

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

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

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

5) Natural Join:-

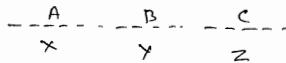
This join also returns matching rows only. In this join, we are not allowed to use joining condition explicitly. Because internally database server only automatically establishes join condition. But here, resource tables must have common column name. This join performance is very high compare to three 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.g.: SQL> select * from t1 natural join t2;



e.g.: 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;

82 Joins

Syntax!:-

Select col1, col2...
from table1, table2, table3
where table1.commonal = table2.commonal
AND
table2.commonal = table3.commonal

92 Joins

Syntax!:-

Select col1, col2
from table1 join table2
on table1.commonal = table2.commonal
join table3
on table2.commonal = table3.commonal;

CONSTRAINTS

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

Generally constraints are created of table columns. All above constraints are defined in two ways

- 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 group 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));

SQL> select * from Z3;

SNO	NAME
1	Minali
1	abc

SQL> set insert into Z3 values (1, 'abc');

error! Unique constraint 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 values.

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

Column level:-

SQL> create table Z4 (sno number (10) primary key, name varchar 2 (10));

Table level:-

SQL> create table Z5 (sno number (10), name varchar 2 (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 are 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

20/08/2015

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

Syntax:-

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

e.x:- `sql> create table hs (sno number (10), name varchar(20), sal number (10),
foreign key (sno, name) references zs);`

whenever we are establishing relationship 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

i) Deletion in master table:-

when we are try to delete a master table record within master table or 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 other wise 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 try to delete
master table record in master table then automatically corressponding master & child
table records are automatically deleted in both the table.

Syntax:-

`sql> create table tablename (col1 datatype (size) references mastertablename
(primarykey column) on delete cascade, ---);`

e.x:- `sql> create table mas (sno number (10) primary key);`

`sql> insert into mas values (....);`

`sql> select * from mas;`

SNO -

1

2

3

4

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

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

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
1
2
3
4

SQL> select * from child;

Sno
1
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 try 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 mastertablename
(primarykeycolumnname) on delete set null);

2) Insertion in child table:-

When we are try 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.x:- 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 emp mas;

SQL> Insert into child values (5);

SQL> select * from child;

Check:-

→ It is used to define logical conditions according to our business rules.

→ It is used to define logical conditions according to our business rules.

```

SQL> create table test (name varchar2(10) check (name = upper(name)));
SQL> insert into test values ('abc');
error: check constraint
(SCOTT.SYS_CO0SS2A) violated.
SQL> insert into test values ('ABC');
SQL> select * from test;

```

21/08/2015

```

e.g:- 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.g:- 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 constraint 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 user defined name constraint type

↳ May be notnull, unique, primarykey, foreignkey, check

```

e.g:- 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_CO0SS4U) violated

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

User defined constraint name

```

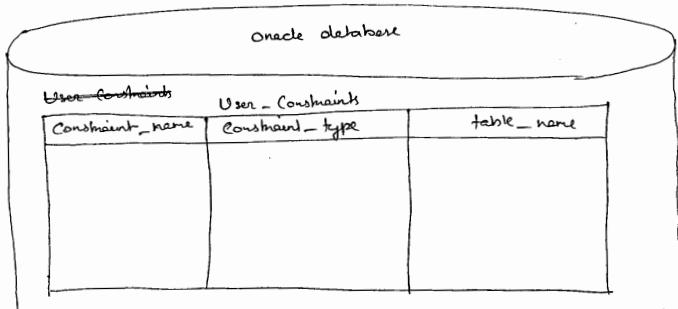
SQL> create table test4 (sno number(10) constraint P_AB primary key);

```

SQL> insert into test4 values (1);
Error: unique constraint
(SCOTT.P_ATB) violated

Data Dictionary !-

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



Note:- In oracle all constraints information stored under user_constraints data dictionary.

e.g:- SQL> desc user_constraints;

SQL> select constraint_name, constraint_type from user_constraints where table_name = 'EMP';
Capitalllette ↴

CONSTRAINT_NAME	CONSTRAINT_TYPE
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:- SQL> desc user_cons_columns;

SQL> select constraint_name, column_name from user_cons_columns;

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;

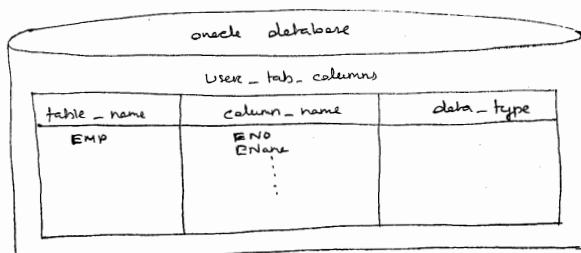
SQL> Select search_condition from user_constraints where table_name = 'TEST';

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_tabs_columns;

SQL> select column_name from user_tabs_columns where table_name = 'EMP';

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

SQL> select count(*) from user_tabs_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_tabs_columns data dictionary.

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.

Primarykey	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;

Creating 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 emp1 add foreign key (deptno)
references dept1 (deptno);

dropping constraints:-

27/08/2015

methods

Alter table tablename drop constraint constraintname

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 we use
using cascade clause along with alter drop

Syntax:-

alter table tablename drop 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='B3';

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 executed.

1) Non-correlated Subquery :-

- Non correlated subquery having two parts
- child query (or) inner query
 - 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

- single row subqueries
- multiple row subqueries
- multiple column subqueries
- 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 use =, >, <, <=, >= 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');

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.

Ex:-

```
SQL> select ename, dname from emp, 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 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 column?

```
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);
```

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 group 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 groups 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	5000
MANAGER	2291.66667
ANALYST	3200

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

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

26/08/2015

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);

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

→ 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.

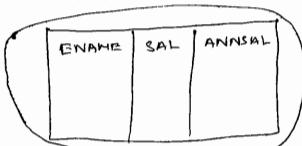
e.g:- SQL> Select ename, sal, sal * 12 annsal from emp where annual > 30000
error "ANNSAL": invalid identifier

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 Annal > 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 these are Rownum, Rownid 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 Rownum having temporary values.

e.x:-

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

ROWNUM	ENAME
1	CLARK
2	KING
3	MILLER

- ③ create a query to display first row from emp table by using Rownum?

sql> Select * from emp where rownum = 1;

27/01/2015

- ④ create a query to display record row from emp table by using rownum?

sql> Select * from emp where rownum = 2;

no rows selected.

- Generally Rownum doesn't work with more than one positive integer i.e it works with less than, <, <= operations.

- ⑤ write a query to display first 5 rows from emp table by using Rownum

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 rows selected.

endif;
end loop;

- (e) 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;
- (f) 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;
- (g) 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;
- (h) 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;
- (i) 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;
- (j) 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~~ >= 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 operators

(k) write a query to display second row from emp table by using rownum alias name?

e.n:-
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

(l) 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);

Q) Create 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) Create a query to display 2nd, 3rd, 4th records from emp table by using rownum alias name? 28/03/2015

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

Q) Create 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) Create 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
JONES	3375

Analytical functions used in inline views:-

Analytical functions:-

→ Oracle 8.2 introduced analytical functions

→ Analytical functions are similar to group functions but group function reduces no. of rows in each group whereas analytical functions doesn't 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	1950

Analytical function:-

DEPTNO	AVG(SAL)
10	3618.66667
10	3618.66667
10	3618.66667

SRI RAGHAVENDRA XEROX

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

→ Oracle having following analytical function they 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 were same & also rank skips next consecutive rank no. where as dense_rank does not skip next consecutive rank number)

e.g:-

row_number()

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

- ② 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

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) n from emp) where n=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) n from emp) where n=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) n from emp) where n=8;

Enter value for n: 2

DEPTNO ENAME

10 KING 5900 1

31/05/2015

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

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)
 r from emp) where r = 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) r from emp) where r = 2;

DEPTNO	ENAME	SAL	R
10	KING	5000	2
20	JONES	2975	2
30	WARD	1250	2

SRI RAGHAVENDRA XEROX

Software Languages Material Available

Beside Bangalore Ayyagar Bakery,

Opp. CDAC, Balkampet Road,

Ameerpet, Hyderabad.

Q) Create 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)
 r from emp) where r <= 2;

DEPTNO	ENAME	SAL	R
10	MILLER	1300	1
20	FORD	3006	2

Deleting Duplicate rows:-

Q) write a query to display duplicate rows from the following table ?

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

SQL> insert into test values (& 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 ore

write a query to delete duplicate rows in a table

SQL> delete from test where sno not in (select min (sno) from test group by sno);

SQL> select * from test;

SNO
10
20
30
40

Multiple Columns Subquery :-

→ In all database we can also compare multiple 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 conditional 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

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 car 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 employees in each job from emp table by using multiple column subquery?

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

Correlated subquery:-

→ Generally in non correlated subqueries child query is executed first then only

- Generally in non-correlated subqueries child query is executed only once per parent query table where as 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 update 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

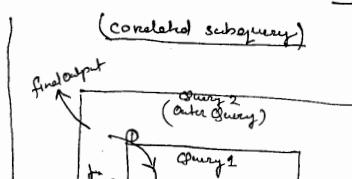
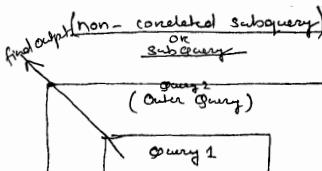
columnname = (Select columnname from tablename2 aliasname2 where aliasname1.
commoncol = aliasname2.commoncolname);

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);
sql> select * from emp;

02/09/2015



(Q) Write a query to display first highest salary employee from emp table by using correlated subquery?

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

KING 5000

(Q) Write a query to display second highest salary employee from the following table by using correlated subquery?

e.xi -
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

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

Execution:-

Phase 1:-

Step1:- get a candidate row (first row) \rightarrow abc 100

Step2:- select count(*) from test e2. where e2.sal >= 100;

4

Step3:- select * from test e1 where 2 = 4; (false)

Phase 2:-

Step1:- get a candidate row (xyz 150)

Step2:- select count(*) from test e2. where e2.sal >= 150;

3

Step3:- select * from test e1 where 2 = 3; (false)

Phase 3:-

Step1:- get a candidate row (pqr 200)

Step2:- select count(*) from test e2. where e2.sal >= 200;

2

Note!- whenever resource table having duplicate data within comparing 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

KING 5000

Exists Operator:-

- In oracle we can also use exist 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 ^{non}empty set returns true. where ever exists operator empty set returns false.

ex1:- exists {1, 2, 3} = true

ex2:- exists {} = false

SRI RAGHAVENDRA XEROX

Software Languages Material Available
Beside Bangalore Ayagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

- 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

- 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 e1ename='SCOTT' and e1.esal=e2.sal);

- 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

Not exists! -

- ②) write a query to display those department doesn't have employees in emp table by using co-related subquery?

Sol) select * from dept d where not exists (select * from emp e where deptno = d.deptno);

DEPTNO	DNAME	LOC
40	OPERATIONS	BOSTON

- ③) write a query to display those department does not have employees in emp table by using non co-related subquery?

Sol) 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. In place of this one we can also use correlated subqueries along with not exists operator.

Ex:-

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

Sol) select * from dept d where deptno not in (select deptno from emp);

no rows selected

Sol) select * from dept d where not exists (select * from emp e where deptno = d.deptno);

DEPTNO	DNAME	LOC
40	OPERATIONS	BOSTON

Subquery special operations used in non-correlated subqueries! -(all, any)

- ④) write a query to display the employees who are getting more salary than the highest paid employee in 20th dept from emp table?

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

- ⑤) write a query to display the employees who are getting more salary than the lowest paid employee in 10th department from emp table?

Sol) 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 table having min function is also a

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 > 10);
SQL> select * from emp

all, any operators used in multiple rows subquery !-

01/09/2015

In \rightarrow It returns same values in the list \rightarrow child query.

\geq All \rightarrow It satisfies all values in the list

\geq 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 \neq any

e.x:- SQL> select * from emp where deptno not in (10, 20);

30

SQL> select * from emp where deptno \neq any (10, 20);

10

20

30

Note:- Not in is also same as \neq all

e.x:- SQL> select * from emp where deptno not in (10, 20);

30

SQL> select * from emp where deptno \neq all (10, 20);

30

SRI RAGHAVENDRA XERC
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 window 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 views to base table based on following restriction:
 - 1) If a simple view contain group(), 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.

exit sql> create or replace view v1

as

select * from emp where deptno=10;

sql> select * from v1;

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 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;`

Note:- In oracle when a view contains rownum also then we must use alias name as.

e.x!- SQL> create or replace view v3
as

select rownum sno, ename from emp;

SQL> select * from v3;

SNO	ENAME
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 there 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 views!

→ 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 try to perform dml operations then some table columns are effected & also some other table columns are not effected if you wants to view effected, uneffected column then use readable - columns data dictionary.

SQL> desc user_updatable_columns;

SQL> select desc column_name, updatable from user_updatable_columns
where table_name = 'VS';

COLUMN_NAME	UPDATABLE
ENAME	YES
SAL	YES
DNAME	NO
LOC	NO

Generally we cannot perform dml operation through complex view 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 trigger code is executed only once per dml statements whereas as in row level trigger trigger code is executed for each row of each dml statements

Syntax:-

create or replace trigger triggername
before/after insert/update/delete on tablename

[for each row] → for row level trigger



Difference between statement level, row level triggers

SQL> create table test (col1 date);

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 depno = 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 depno = 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 all statements. that's why we are using for each row clause in trigger.

These qualifiers are used in either in trigger specification or in trigger body.

Syntax:-

: old. columnname

Syntax:-

: new. columnname

	insert	update	delete
new	✓	✓	✗
old	✗	✓	✓

- ② Create a PL/SQL row level trigger on emp table, whenever we are 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

update dept set
 dname = ? new.dname where
 dname = ? old.dname
 update dept set = ? new loc where loc = ? old.
 end; /
 trigger created

Testing:

SQL> update vs set dname = 'xyz'
 where dname = 'SALES';
 1 row updated.
 SQL> desc were_updatable_columns;
 SQL> select column_name, updatable from were_updatable_columns where
 table_name = 'VS';

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仓库 having application.
- Generally views doesnot store data, whereas materialised views stores data.
- Materialized views are used to improve performance of the joined or aggregateable 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:-

View

- 1) View doesn't store data
- 2) View purpose is security

Materialized view

- 1) Materialized view stores data
- 2) Materialized view purpose is Improve

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.g:- sql> create materialized view m21

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 m21

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.g:- 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;
```

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 oracle's 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 view by using select statement then oracle server each & everytime executes view definitions in this case each & every time base tables are effected.

Performance is very high compare to view performance i.e

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 table views are also called as 'Virtual tables' are view is also called 'Windows of a table' i.e. Through the View we can Access base table data.

SQL> Select rowid, sno, name, mct;

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 mct;

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

↳ predefined
procedure name

e.x! - sql> exec dbms_mviews.refresh ('mw1');

pl/sql

sql> select * from mw1;

SNO	NANE
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 rowsids 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 mw1;

sql> dbms_mviews.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 rowsids are recreated, where as in fast refresh materialized views rowsids are not changed even if we are refreshing materialized

- 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.

e.g:-

```
SQL> create materialized view log on base;
SQL> create materialized view mw2 refresh fast as select * from base;
SQL> select rowid, sno, name 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

Materialized View:-

Syntax:-

Materialized view viewname

refresh complete / refresh fast on demand / on commit

as

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

→

→ and some other SQL commands.

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 exist.

SQL> conn scott/tiger;

SQL> grant all on emp to murali;

SQL> select * from emp;

error: table or view does not exist

SQL> select * from scott.emp;

SQL> create synonym ab for scott.emp;

SQL> select * from ab;

privilege

10/9/2015

→ 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

i) system 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, murali;

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;

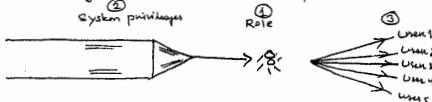
Role!-

- 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

Administrator 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, murel;
```

→ In oracle all system privileges related to role stored under role-sys-prive's data dictionary.

e.g:- SQL> conn sys as sysdba;

Enter password: sys

```
SQL> desc role-sys-prives;
```

```
SQL> select role, privileges from role-sys-prives  
where role = 'R1';
```

ROLE	PRIVILEGE
R1	CREATE TRIGGER
R1	CREATE ANY INDEX
R1	CREATE PROCEDURE

Predefined roles:-

Whenever we are installing oracle server then automatically three predefined roles are created these are

① Connect → used by end users

→ Connect role internally having create session privilege which is used to allow 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) → all 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;

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.g!:-

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 views only used for security.

Syntax!-

grant all on viewname to username1, username2...;

e.g!:- 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 condⁿ value from view to base table.

Syntax!-

Create or replace view viewname

as

Select * from tablename where condition with check option;

e.g!:- SQL> Create or replace view v1

as

Select * from emp where deptno=10 with check option;

SQL> Select * from v1;

sql> insert into v1(empro, ename, depno) values (1, 'abc', 10);
1 row inserted.

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 perform DML operation.

e.x:- sql> create or replace view v2

as
select * from emp where depno=10 with read only;

view created

sql> delete from v2 where depno=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 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> drop 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.

```
SQL> Conn Mnelli / Mnelli;  
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 11i 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 are 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, ...  
when not matched then  
insert (targettablecolumns) values (source table columns);
```

e.x:-

```
SQL> create table deptS as select * from dept;  
SQL> insert into deptS ('1','a','b');  
SQL> merge into dept d  
using deptS s  
on d.deptno = s.deptno  
when matched then  
update set d.dname = s.dname, d.loc = s.loc  
when not matched then  
insert (d.deptno, d.dname, d.loc) values(s.deptno, s.dname, s.loc);  
SQL> select * from dept;
```

15/9/15

Note!:- Through merge statement we cannot update all columns that is we can not update on clause columns.

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 for  
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  
for username. objectname @ database Linkname;
```

```
SQL> scott/H4ger
```

```
SQL> insert all on  
emp to murlali, a1;
```

```
SQL> conn murlali/murlali
```

```
SQL> Select * from scott.emp  
SQL> Create synonym xy for  
scott.emp;
```

```
SQL> select * from xy;  
SQL> Create public synonym xy  
for scott.emp
```

Error: insufficient privileges

```
SQL> Conn sys as sysdba  
pass: -sys
```

```
SQL> grant create public synonym  
to murlali
```

```
SQL> conn murlali/murlali;
```

```
SQL> conn a1/a1
```

```
SQL> select * from scott.emp  
SQL> select * from xy;
```

Error: Tab (or) View does not

- we can also drop synonym by using drop synonym synonymname.
- All synonym information stored under user-synonym data dictionary.
- 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 use using following two pseudo column there are

- 1) current
- 2) nextval

current returns current sequence number where nextval returns nextval sequence number.

Syntax:-

- 1) sequenceName . current
- 2) sequenceName . nextval

- These pseudo columns are used in insert, update, delete, select statements.

→ In order if you want to access sequence value to

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.

```
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;
```

16/9/2015

```
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.g:-

```
sql> create sequence s1
```

start with 5

increment by 1

minvalue 3

maxvalue 100;

```
sql> select s1.nextval from dual;
```

5

```
sql>/
```

c

```
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.g:- sql> create sequence s1

start with 3

increment by 1

minvalue 5;

Note:- Generally sequences are used to generate primary key value automatically

e.g:- sql> create table test (order_no number(10) primary key, itemname varchar2(10));

sql> create sequence s1 start with 1001;

```

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;


| <u>ORDERNO</u> | <u>ITEMNAME</u> | <u>SNO</u> |
|----------------|-----------------|------------|
| 1001           | abc             |            |
| 1002           | xyz             |            |
| 1003           | pqr             |            |


SQL> update test set
      sno = s2.nextval;
SQL> Select * from test;


| <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 requesting 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 finds any value as input to cache memory area from that

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 Oracle user_sequences data dictionary
SQL> desc user_sequences;

→ We can also drop sequence by using drop sequences 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 column then database server very fast retrieving data from 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 three indexes on those columns.

2) Manually:-

We can also create indexes 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

already having indexes also.

→ Oracle having two types of Indexes

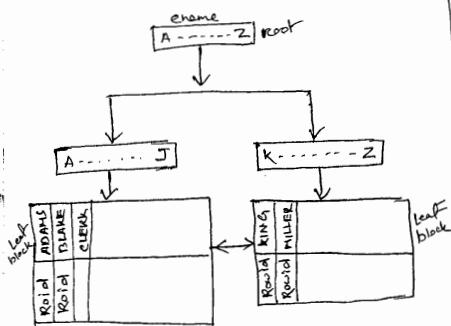
- 1) btree indexes
- 2) bitmap indexes

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 always leaf blocks stores actual data along with rowids. whenever we're requesting data by using where clause or order by clause then oracle server searches those columns having btree indexes. even btree indexes are available then oracle server internally uses index scan on btree structure to retrieve data very fastly from the leaf blocks.

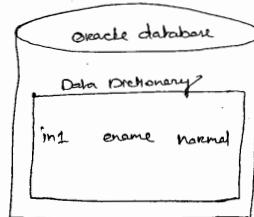
Developer

```
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(ename);
```

```
sql> desc user_indexes;
```

```
sql> select index_name, index_type from user_indexes where  
table_name = 'EMP';
```

e.x:-
sql> desc user_ind_columns;
sql> select index_name, column_name from user_ind_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 index)

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());

10/9/2015

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 functionbased 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 column!:-

- Oracle 11g introduced virtual columns in a table which store stored expression directly into database. Prior to oracle 11g we can also stored stored expressions

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 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
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);

$$\text{Cardinality of a column} = \frac{\text{number of Distinct values}}{\text{Total number of records}}$$

ex:-

1) cardinality of a empno col = $10^7 \approx 10^7 \Rightarrow$ high cardinality
 ↴ bitmap indexed.

2) cardinality of a Job col = $5/14 \Rightarrow 0.357 \dots \Rightarrow$ Low cardinality
 ↴ bitmap indexed.

21/9/2015

- whenever we are creating bitmap 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 bitmap function

SQL) create table bitmap index in1 on emp(job); | Develop
 SQL select * from emp where job = 'CLERK';

Job	Bitmap table												Rowid	
	1	2	3	4	5	6	7	8	9	10	11	12	13	
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.

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 operator always corresponding expression must belongs to same datatype & also set operators always retains 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

→ In oracle ^{where} 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 number into character functions then oracle server automatically converts number type into character type.

e.x! -
SQL> select length ('3156') 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 Paved 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	number	yes	yes
Varchar or char	Date	yes	yes
number	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

e.x!-

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

<u>deptno</u>	<u>decode (deptno)</u>
10	ten
20	twenty
30	others.

Q) write a query to update salary commission of the employee within emp table by using following condition
 1) 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.

select 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;

<u>JOB</u>	<u>deptno 10</u>	<u>deptno 20</u>	<u>deptno 30</u>
CLERK	2000	3000	1450
SALESMAN			2800
PRESIDENT	5300		
MANAGER	2750	3075	2950
ANALYST		6200	

e.x!- select ename, 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 ename

/

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 operator 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.g:- `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`

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 () over columnname in (Value1, Value2));
```

e.g. :-

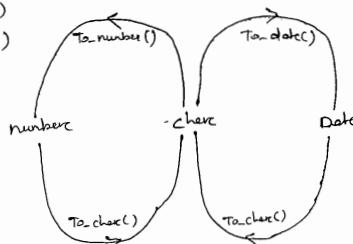
```
sol) select * from (select job, sal, deptno from emp)
pivot (sum(sal) over deptno in (10 as deptno10, 20 as deptno20, 30 as deptno30));
```

JOB	deptno10	deptno20	deptno30
CLERK	2000	3000	1450
SALESMAN			2800
PRESIDENT	5300		
MANAGER	2750	3075	2950
ANALYST		6200	

```
sol) select * from (select deptno, job from emp)
pivot (count(*) over 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()



To_number!:-

e.x!- `select '$35.6' + 3 from dual`

error:

`select to_number('$35.6') + 3 from dual`

error:

Solution:-

`select to_number('$35.6', '$99.9') + 3 from dual;`

Output:-

38.6

e.x!- `select to_number('a35.6') + 3 from dual;`

error:

`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 convert

e.x!- `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!- `select to_char(sysdate, 'j') from dual;`

2457289

Julian date!-

→ Julian date is number of days 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 numbers into Julian date then we are using format 'j' within `to_date()`

e.x!- `select to_date(123, 'j') from dual;`

24/09/2015

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 numbers

G

group separator (,)

d

decimal indicator (.)

\$

dollar indicator

0

Leading Zeros

L

local currency

e.g. - group separator (,)

SQL> select to_char(1234567, '999,999,999') from dual;

output

12,34,567

e.g. - decimal indicator (.)

SQL> select to_char(1234567, '999.999.999.99') 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

SQL> Select to_char(123, '\$999') from dual;

\$123

e.g. - Leading zeros

SQL> select to_char(123, '00999') from dual;

00123

SQL> select to_char(123, '999900') from dual;

0123

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 R(nls_currency) format

Syntax:-

'nls_currency = own currency'

e.g:- `select ename, to_char(sal, 'L99g99g999d99', '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

`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

`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()

Dnvl():-

→ It is used to substitute (or) replace where value in place of null.

Syntax:-

`null(exp1, exp2, exp3)`

→ Hence 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

`sql> select nullif (10, 10) from dual;`

`NULLIF (10,10)`

`sql> 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?

`sql> 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 nonnull value within given expressions

Syntax:-

`coalesce (exp1, exp2, ..., expn)`

`sql> select coalesce (null, null, 30, null, 50) from dual;`

Output

30

`sql> select ename, comm, sal, coalesce (comm, sal) from emp;`

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 coalesce () exp1, exp2 must belongs to same datatype.

e.g:-

sql> select nvl('a', sysdate) from dual;

a
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, updation, deletion problems.

→ In design phase of the SDLC database designer 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 column

(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 (INF 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 eroded table because in this table one column having duplicate data.

Front End

Electronic Shop

Order form

order no: 1

order date:

cust Name:

address:

phone no.:

Item name: Refrigerator
 LED TV
 add another item

Amount:

Foreign 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 atomicicity whereas second normal form deals with relationship between key, non key attribute.
- If a table is in first normal form & also that table 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 putting 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.R. diagram

Child Table

Primary key: Item name

Foreign key: color

Functional dependency: color depends on item name

Partial dependency: price depends on item name

Non-key attribute: tax

2NF

Primary key: Item name

Foreign key: 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	Activity 2	cost
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

28/09/2015

Activity table

Stno	Activity	cost	Activity	cost
101	cricket	\$10	football	\$20
102	cricket	\$10	golf	\$30
103	football	\$20	golf	\$30
104	football	\$20		

Primary key: Stno, Sname

Foreign key: (Stno, Activity) (Activity)

Foreign key: (Activity) (Activity)

Foreign key: (Activity) (Activity)

Stno	Sname	Activity	Activity	Activity
101	abc	101	cricket	Cricket
102	xyz	101	football	Football
103	pqr	102	cricket	
104	zzz	102	golf	Golf
		103	football	
		103	golf	Golf
		104	football	Football

Stno	Sname
101	abc
102	xyz
103	pqr
104	zzz

Primary key: Stno

Functional dependency: activity depends on Stno

Non-key attribute: cost

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, People

Non-key attribute: Dept, Depthead, locno

Ecode	People	Dept	Depthead	locno
E01	P1	System	D1	4
E02	P1	Sales	D2	5
E03	P2	Research	D3	7
E01	P2	System	D1	6
E02	P2	Sales	D2	8
E01	P3	System	D1	9

Phase 1: 2nd Normalization

① E01 → locno

② E01 → P1 (Functional dependency)

③ E01 → D1 (Functional dependency)

Phase 2: 3rd Normalization

① Ecode → Dept (Functional dependency)

② People → Dept (not functional dependency)

Primary key (2nf table) (Master Table)		
Ecode	Dept	DeptHead
E01	System	D1
E02	Sales	D2
E03	Res	D3

Foreign key (Child Table)		
Ecode	Percode	Hours
E01	P1	4
E02	P1	5
E03	P2	7
E01	P2	6
E02	P1	8
E01	P3	9

→ In the above e.g 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 x 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 .

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

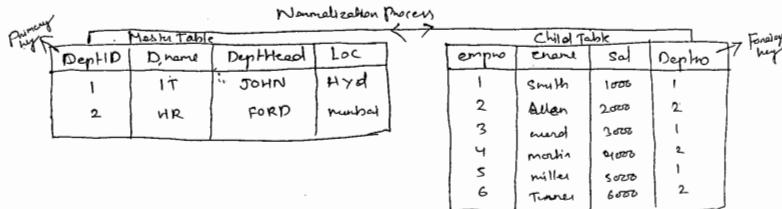
Q) Why Normalization process?

→ Normalization is a scientific process which is used to reduce redundancy & also automatically avoids insertion, updation, deletions problems.

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

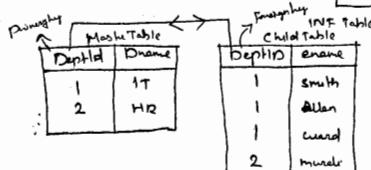
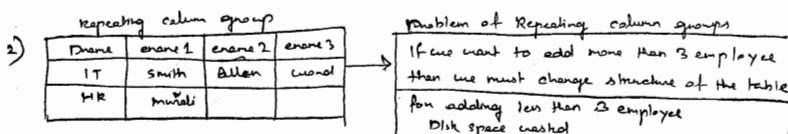
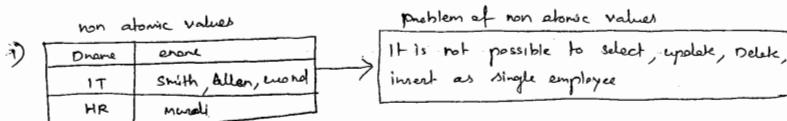
Problems on Data Redundancy

- 1) Disk space wastage
- 2) Data inconsistency
- 3) DML query become slow



A table is in INF

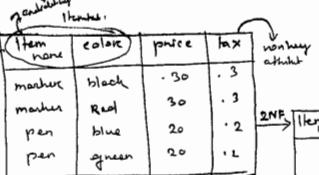
- 1) Data in each column should be atomic. There is no multiple values separated with commas.
- 2) Table doesn't contain repeating column groups.
- 3) Identifying a record uniquely by using primary key.

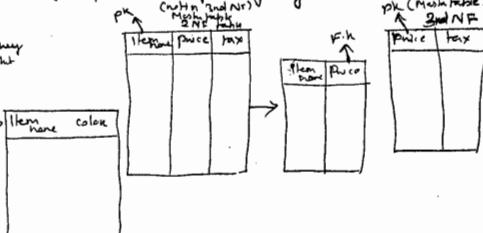


→ If a table is in 2NF & also any non key attributes which depends on another non key attributes then that table not in 3rd NF

Process :-

→ Identify non key attributes which depends on another non key attributes puts 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:- 



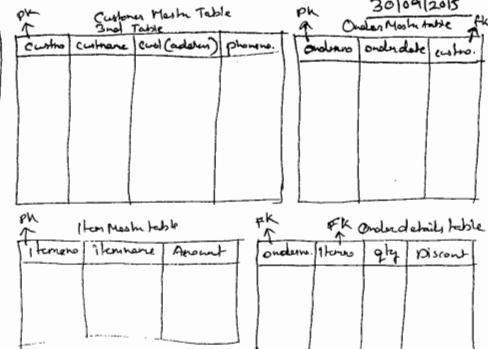
order form

order no _____
order date _____
cust no. _____

Item name	Items	▼
Refrigerator	q'ty	
<input type="checkbox"/> add another item	discount	
Amount	submit	

Employee form

Custno _____
Custname _____
Address _____
Phone no. _____



consolidating Cust in 3NF
Student table

key attribute

studentname	courseid	Grade	Value
abc	CS111	A	4.00
xyz	CS111	B	3.00
pqr	CS222	C	2.00
zzz	CS222	A	4.00

nonkey attribute

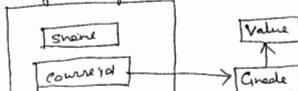
FK ↑

studentname	courseid	Grade
abc	CS111	A
xyz	CS111	B
pqr	CS222	C
zzz	CS222	A

FK ↑ (Master table)
3rd table

Grade	Value
A	4.00
B	3.00
C	2.00

logical Diagram



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 cluster 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.

Step 1:- Create a cluster using common column-name.

Syntax:-

```
create cluster clustername (colcommon columnname datatype (size));
```

Step 2:- Create an index on cluster

Syntax:-

```
create index indexname on cluster clustername;
```

Step 3:- Create cluster tables

Syntax:-

```
create table tablename (col1 datatype (size), col2 datatype (size), ....)  
cluster clustername (commoncolumnname);
```

e.g:-

```
sql> create cluster emp_dept (deptno number (10));
```

```
sql> create index in1 on cluster emp_dept;
```

```
sql> create table el (empno number (10), ename varchar2(10), deptno number(10))  
cluster emp_dept(deptno);
```

```
sql> desc el;
```

```
sql> create table dl (deptno number (10), dname (varchar2(10)), loc varchar2(10))  
cluster emp_dept(deptno);
```

```
sql> desc dl;
```

`SQL> select rowid from emp;`
`SQL> select rowid from dept;`

Note:- In all databases by default cluster table having same rowid.

rule:- we cannot drop cluster 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.g:-
`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.

(or) A superkey which is a subset of another superkey then that total key is not a candidate key.

e.g:-

empno	ename	skillid	skill	votoid
1	manali	1	Oracle	V1
1	manali	2	Teradata	M1
1	manali	3	Bybase	V1
2	abc	4	DB2	V2
2	abc	1	Oracle	M2
3	xyz			V3
4	zzz	5	Information	V4
4	zzz	4	DB2	V4
5	pqr	6	Informix	V5

Superkey

- ① empno + skill
- ② empno + ename + skill
- ③ empno + votoid + skill
- ④ ename + skill
- ⑤ ename + votoid + skill
- ⑥ votoid + skill
- ⑦ empno + ename + votoid + skill

not a Superkey

- ① empno + ename
- ② empno + votoid
- ③ ename + votoid

Candidate key

- ① empno + skill
- ② ename + skill
- ③ votoid + skill

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

- Generally 2nd, 3rd NF deals with relationships b/w key, non-key attribute where as 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. This table also behaves like a master table. This BCNF table does not contain non-key attribute.

02/10/2015

Profcode	Dept	H.O.D	hours
P ₁	physics	murali	5
P ₁	computer	abc	3
P ₂	chemistry	xyz	7
P ₂	botany	pqr	8
P ₂	physics	murali	9

Nonkey attribute

$$\begin{array}{l} \textcircled{1} \text{ profcode + dept } \xrightarrow{\text{FD}} \text{ hours} \\ \textcircled{2} \text{ profcode + H.O.D } \xrightarrow{\text{FD}} \text{ hours} \end{array}$$

$$\begin{array}{l} \textcircled{1} \text{ profcode + Dept } \rightarrow \text{ Candidate key} \\ \textcircled{2} \text{ profcode + H.O.D } \rightarrow \text{ Candidate key} \end{array}$$

$$\begin{array}{l} \textcircled{1} \text{ Dept } \xrightarrow{\text{FD}} \text{ H.O.D} \\ \downarrow \\ \text{determinant} \\ \downarrow \\ \text{Candidate key} \end{array}$$

Dept	H.O.D
physics	murali
computer	abc
chemistry	pqr

$$\begin{array}{l} \textcircled{2} \text{ H.O.D } \xrightarrow{\text{Determinant}} \text{ Profpl.} \\ \downarrow \\ \text{Candidate key} \end{array}$$

Profcode	Dept	hours
P ₁	physics	5
P ₁	Computer	3
P ₂	chemistry	7
P ₂	botany	8
P ₂	physics	9

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.x:-

multivalued dependency

ename

languages
English

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 tables contains more than two attributes of 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 some attributes set's values which depends on another attribute set's value and also some attributes are not logically related then only we are using 4thNF 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 4thNF.

Assumption 1 -

- Each employee having multiple projects
- Each employee having multiple skills

4NF

Ecode	proj code	Skills
111	Railway Res	JSP
111	Railway Res	Oracle
111	Railway Res	Asp.net
111	Library Mgt	SqlServer
111	Library Mgt	Vct+f

Ecode	Proj code
111	Railway Res
111	Library Mgt

Ecode	Skills
111	JSP
111	Oracle
111	Asp.net
111	SqlServer
111	Vct+f

- Before 4thNF 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 his skills these null value problem is automatically avoided in 4thNF process tables & also these tables reduces duplicate data

Fifth Normal Form! -

- When a relation cannot be decomposed into number of relation is called fifthNF.

into no. of tables. Then we must ~~join~~ check ~~when~~ ~~as~~ they are joining these table then result and record must be available in resource table are not.

LOCKS

- Locking is a mechanism which prevents unauthorised access for one 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 these method we use locking set of rows by using (for update) clause these clause is used in select statement only.

Syntax:-

```
Select * from tablename  
where condition for update [nowait];
```

- Whenever we are performed 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/tigress
```

```
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 ~~order~~ clause use along with for update clause. Whenever we are using nowait clause automatically controls returns to the current session if another

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 lock]

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 updated)
3 rows updated.

Table level locks:-

In this method DBA performs locks on a table oracle 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 (working)
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

SQL> Conn murali/murali

SQL> lock table scott.emp in share mode

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 belong 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

y/level:-

→ Level is a pseudo column which automatically assigns numbers to each level within tree structure.

a/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 relationship between hierarchical columns by using prior operator.

Syntax:-

connect by prior parentcolumnname = childcolumnname

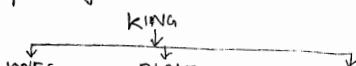
Syntax:-

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

(Q) select level, ename from emp start with mgr is null connect by prior empno = mgr



Level
1

Note:- Oracle 9i introduced sys_connect_by_path() which returns path of the hierarchy within tree structure

Syntax:-

sys_connect_by_path (columnname, 'delimiterchar')

e.g. 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?

e.g. 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

e.g. 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 whenever 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 searches this column value available in another hierarchical table

6/10/2015

Control files:-

- It controls all other files in storage area. These file extension is .ctl
- Control files also stores logically created databases information
- These control files are used by DBA in Background 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$controlfile;  
SQL> select name from v$controlfile;
```

Redolog file:-

- Redolog files store committed information from Redo log buffer. These file extension is ".log".
 - Redolog files also used by database Administrator in backup and recovery process.
 - In oracle all Redolog file information stored under v\$log file data dictionary
- For ex:- 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
 - 1) SGA
 - 2) Process
- 1) SGA:- It is also called as system Global Area (or) Shared Global Area. SGA memory having set of buffers these are
 - 1) Database buffer cache
 - 2) Shared pool
 - 3) Java pool
 - 4) Large pool
 - 5) Redolog buffer.

- Whenever we are submitting SQL, 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.

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 process large amount of data & also Redolog Buffer Store new information for the commit 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 user 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.

```

client
Sql plus / tool
username : scott
password : tiger
(ok)

SQL> select * from Test;
  Test
  Sno
  1 5
  2 4
  3 3
  4 2
  5 1

```

```

SQL> update Test set sno = 5
where sno = 1;
SQL> commit;

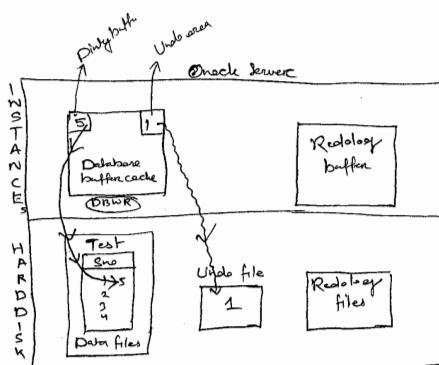
```

Flashback query!-

→ Oracle 9i introduced flashback queries. These queries are handled by database administrator. Flash

→ 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.



) Using Time Stamp:-

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.

Ex:-

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:-

Systimestamp + 'interval 'number' minute (or) hour (or) sec'

Ex:-

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.

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:-

DBwriter processes always fetches data from database buffer cache and store the data permanently into data file.

LGR:-

→ Log writer fetches data from Redolog buffer and stores the data into Redolog file.

→ Generally whenever we are using DML transaction new value for the transaction also stored in redolog buffer. Whenever we are using Commit or 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.

CWTR :-

→ 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 number also we can retrieve accidented data by using flashback query.

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;

SQL> show parameter current_scn

SQL> insert into b1 values (70);
SQL> commit;

Fleshback 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 recovering any other background process if problem is occurred.

pmon:- (process monitor)

- pmon process automatically de-allocate were process when clients not disconnected from this server also

pga!:-

→ pga is also called as private global area.

→ pga memory areas 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 execute collections in pl/sql

pga

logon	session	sql work area
-------	---------	---------------

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

→ In oracle if you want to store multiple data item in oracle memory area then we are using index by table in pl/sql but these tables are not allowed to store permanently into oracle database to overcome this problem oracle 8.0 introduced extension of the index by table called nested table which is used to store permanently into oracle database by using sql

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> /

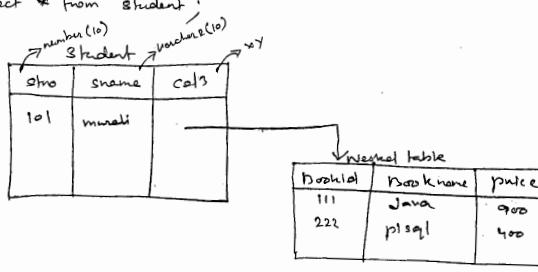
sql> create or replace type xy as table of obj1;

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

e.g.) 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 these operation then we must use table operator along with sub-query in there subquery we must specify nested table column this table operator we replace of relational tablename 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

- partition tables are treated as very large databases. partition table are used to improved performance for the application. In background recover process in partition 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 partition2 values less than (max_value));
-----
```

To view particular partitions:-

Syntax:-

```
Select * from tablename partition (partitionname1, partitionname2, ...);
```

E.X:-

```
sql> create table test (eno number (10), name varchar2 (10), sal number (10))
partition by range (sal) (partition p1 values less than (1000),
partition p2 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);
```

ENO	NAME	SAL
1	a	500

Value (?)

2) List partition :-

→ Oracle 9i introduced List partition.

→ Generally wing host partition database administration create partitions based on character datatype column these partitions is ~~for eg.~~ events based on list of values.

Syntax:-

create table tablenane (col1 datatype (size),....) partition by list (keycolumn)

partition partitionName1 values (Value1, Value2, ...)

— 10 —

partition partitionname values (default))

P. xii.

!-
981> create table test1 (eno number (10), name varchar2 (10))

partition by 1st(name) (partition pt1 values ('india'), 'pakistan'))

partition p2 values ('ws', 'uk', 'canada'), partition others ~~more~~ values (details))

\Leftrightarrow dere test1;

```
SQL> insert into table test1 values (...);
```

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

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

portion (others);

SNO	NAME
1	aus
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 (key column name) 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-tab_partitions

data dictionary example:-

Oracle Storage Structure :-

→ Oracle having following storage structure

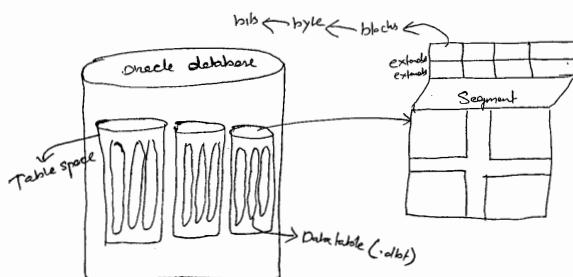
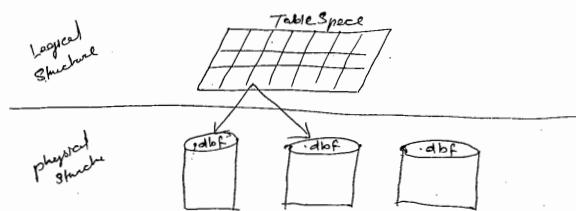
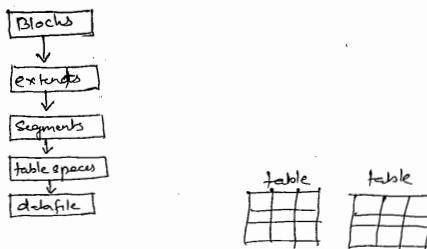


Table Space :-

→ Table space is a logical storage unit which contains collection of datafiles physically. one datafile belongs to one tablespace only. whenever we are installing oracle server then automatically 6 tablespaces are created. If you want to run Oracle minimum 2 tablespace are required these are

System tablespace, sysaux tablespace

e.g:-

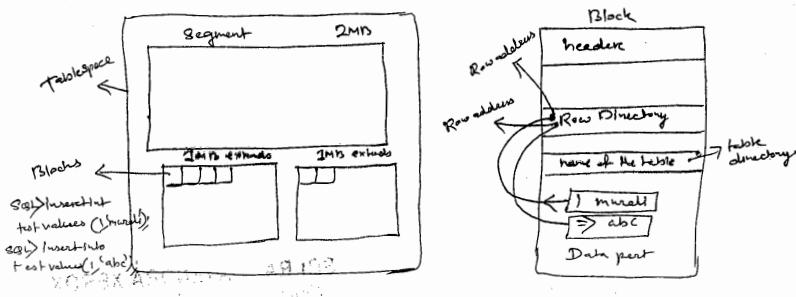
```
SQL> create tablespace tab1 'C:\ORACLE\PRODUCT\10.2.0\ORADATA\ORCL\SYSTEM01.DBF';
```

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 this is called Segment.

```
SQL> create table test (eno number(10), name varchar2(10));
```



→ Oracle having pseudo column rowid which uniquely identifying rows on a table using rowid we can retrieve data very fastly from the database. Oracle rowid having 14 parts rowid 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';
```

AAAMgZAAEAAAAAG AAA

Here

AAAMgZ → data object number

AAE → Tablespace (Relative datafile number)

AAAAAG → data block number

AAA → ROW number within a block

CODD Rules

1) Information rule:-

→ All information in a database must be represented within tabular form i.e all data should be present to the user 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 = 7782;

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 a relational tables

5) All databases having unique方言language

e.x:- SQL

6) View updatable rule.

7) High level insert, update, delete

e.x:- union, union all, intersect, minus

8) physical data independence

9) Logical data independence

10) Integrity independence

11) Subdivision distribution

12) e.x:- distributed databases

13) 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.

O

MURALI & JR
ORACLE 12C

Pranit khade

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

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
- w/m-concat ()

5) Triggers

- row level triggers
- .. apply
- auto increment
- stmt level triggers
- triggers execution order
- Follow clause (11g)
- command triggers (11g)
- mutating error
- system level triggers

c) Packages

- global variables
- overloading procedures

Fundamentals - Forward declaration

viimp:

↓ 7) Types Used in packages

- pl/sql record

8) Bulk Bind

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

- Date _____
- 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 RAHUVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

O.

P2
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

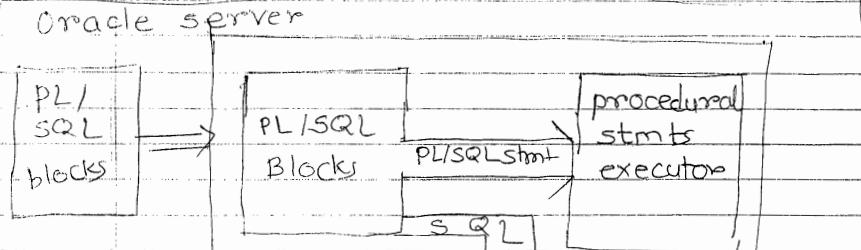
Oracle 7.0 → PL/SQL 2.0

Oracle 8.0 → PL/SQL 8.0

Oracle 12c → 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.



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

var name datatype(size);

Storing a value into variable

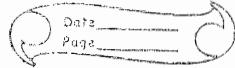
using assignment operator

(:=) we are storing a value
into variable.

Syntax:

env
mer
sett
For
sel

~~atiedt. k~~



Date
Page

```
begin
a := 50;
end;
```

SRI RAGHAVENDRA XEROX

Software Material Available

Bakery

Opp. CDAC, Banjampet 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.

environment setting for SQL

```
> set serveroutput on;
> begin
  dbms_output.put_line('Welcome');
end;
/
Welcome
```

> declare

* Select ... into clause :-

It is used to recover data from table & store it into PL/SQL variables. It always ~~record~~ 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

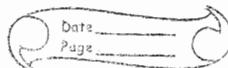
v_ename varchar2(10);

v_sal number(10);

begin

select ename, sal

into v_ename, v_sal



dbms_output.put_line(v_ename1)
' ' || v_sal);
end;
/

Enter value For Ind: 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.

syntax:-

varname datatype(size) not null
:= value;

varname ~~not constraint~~ 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);

Date _____
Page _____

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;
```

/ where

* (not using ^ since max return single value & select ... into ... clause required one record at a time)

➤ declare

a ~~o~~ number(10);

b number(10);

c number(10);

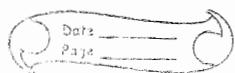
begin

a := 70;

b := 30;

c := greatest(a, b);

i → [c := max(a, b)]
↑ wrong



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.

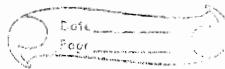
e.g.

```
> 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 ~~an~~ variable attribute then oracle server automatically allocates memory for the variable as same as corresponding column.



PL/SQL having 2 types of variable attributes. This 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



ble
dbms_output.put_line(v_ename||' '|'
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



```
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 Ayagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

declare

i emp%rowtype i.job

i	empno	ename	mgr	job	hiredate	... dept
	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 ~~is~~ 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 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

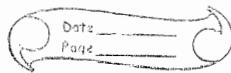
500

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) elif :-

To check more no. of condition
then we are using elif.

Syntax :

```
if condition1 then  
  stmts;  
elif condition2 then  
  stmts;  
elif condition3 then  
  stmts;  
else  
  stmts;  
end if;
```

e.g.

>declare

v-deptno number(10);

begin

Date _____
Page _____

```
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.

* note1:- When PL/SQL block contains "select ... into clause" & also through that if requested data not available in a table then

ORA-1403: no data Found.

Note 2 :-

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;  
 /
```

PL/SQL procedure successfully completed

Note 3 :-

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



e.g.

declare

i emp%rowtype;

begin

select * into i from emp

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

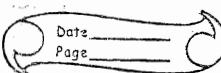
This loop is also called as infinite loop. Here body of loop stmts are executed repeatedly.

syntax:

loop

stmts;

end loop;



```
>begin  
loop  
dbms_output.put_line ('Welcome');  
end loop;  
end;
```

1

* 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;

1

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

e.g.

> declare
n number(10) := 1;
begin
loop
dbms_output.put_line(n);
if n >= 10 then
exit;
end if;

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

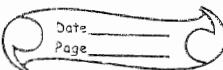
loop

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
```

⇒ 1. 2 3 ... 4

> declare

```
n number(10);
```

begin

For n in reverse 1 .. 10

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



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;
```

→ PL/SQL having 2 types of block

anonymous

- ↗ these block

does not have name

Named

block

- have name

- not allowed to store permanently in database

- not allowed to call a final

- automatically permanently stored in database

- allowed to call a final

2. CURSOR

Cursor is a private SQL memory area which is used to process multiple records & also ~~is~~ this ~~a~~ record by record process.

All database system has 2 types of static Cursor.

- i) Implicit Cursor
- iii) 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 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



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';

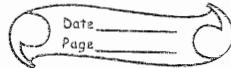
Open

Whenever we are opening then only oracle server retriever 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



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);



Fetch c1 into v_ename, v_sal;
dopl ('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-enameli
' || ' having low salary');

close c1;
end;

1
o/p

SMITH 900

my second employee salary is 600.
WARD having low salary

Explicit Cursor Attribute

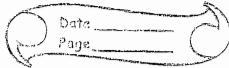
Every explicit cursor having
4 attr. these are

1) %notFound

2) %Found

3) %isopen

4) %rowcount



When we are using these attribute then we must specify cursorname along with these attr.

syntax:

cursorname%attributename ;

Except %rowcount all other cursor attr. returns boolean value either 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. ~~set~~ 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



```
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 1st 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 ;
vENAME varchar2(10);
vSal number(10);
begin
open C1;
fetch C1 into vENAME, vSal
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;
```



Whenever we are creating a cursor then oracle server automatically allocate 4 memory allocation along with cursor memory area. These memory location are identified through cursor attr. These memory location behaves like a variable i.e. these also store only one value at a time.

sql> declare

a number(10);

b boolean;

begin

a := 10;

b := true

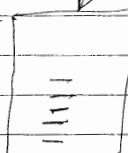
end;

a
10

b
true

emp

c1



c1.notFound

True

c1.Found

true

c1%isopen

true

c1.rowcount

2

* 1. rowcount :

This attribute always return ~~no~~ number datatype i.e. it returns number of record's number fetched from 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:' || '
' || %rowcount);

close c1;

end;

/

SMITH

Q. Write a PL/SQL cursor program which display 5th 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

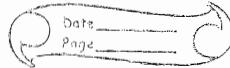
Beside Bangalore Ayyagar Bakery,

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 varchar2(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;

1
> select * from target
sno ename sal

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 without using sum fn.

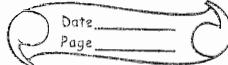
⇒

```
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 + nvl(i.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;

$\%rowtype$

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

Date _____
Page _____

```
begin
  For i in c1
    loop
      dbms_output.put_line(i.enamll);
      || 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.enamll);
      || i.sal);
```

Q. Write PL/SQL to display 5th record from emp by using cursor for loop.

Ans. \Rightarrow 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.

\Rightarrow declare
 cursor c1 is select * from emp;
 n number(10,2);
 begin
 for i in c1
 loop
 n := n + nvl(i.sal, 0);
 end loop;

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;

*%Found :

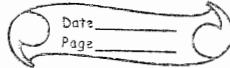
This attr. returns boolean value either true or False.

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;



emp;

```
iF C1%Found then
dbms_output.put_line ('u r emp
does not exist !!' || i.ename);

else iF C1%NotFound then
db-O/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

37
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) % . Found → True : Fetch stmt returns at least one row

→ False : Fetch stmt does not return any row.

2) % . notfound → True : Fetch stmt does not

4) -

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

Syntax:

Formal



cursor cursorname (parameter
datatype) is select columnname
or * from tablename;

open cursorname (actual parameter);

e.g. > declare

cursor c1 (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'

Employee Working as Clerk

SMITH

ALLEN

JAMES

MILLER

Employee Working as Analyst

declare

cursor c1(p_job varchar2) is

select * From emp;

in 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');

dbms_output.put_line ('Employee working as
Analyst');

loop

Fetch c1 into i;

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 as a parameter then display passed deptno details from emp by using cursor for loop.

⇒ declare

cursor c1 (deptno number) is
select * from emp where
deptno = p_deptno;

begin

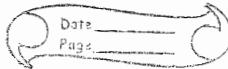
for i in c1(10)

loop

dbms_output.put_line (i.enamell...);

end loop;

Not



```
eg. 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.

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.ename||'
'|| j.deptno);
end loop;
end loop;
end;

/

my dept table deptno is 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;
```



- 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:

No. of employee :

Max sal :

Min " :

Total " :

2011

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;
```

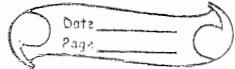
```
i c1%rowtype;
```

```
begin
```

```
open c1 (&deptno);
```

```
Fetch c1 into i;
```

```
dbms_output.put_line('Number of Employee are:' || ' ' || i.a);
```



20/10
o
ng

```
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 || ' ' ||
```

* 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 * 1000~~ 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;

* 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.

PL/SQL

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



>begin
delete from emp where ename
= 'welcome';
end;

ne,

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.

P 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.

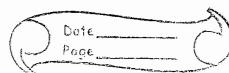
Oracle having three types of Exception.

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 in to 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.

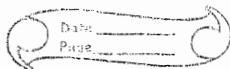
e.g.

begin

insert into emp(empno, ename, sal)

values (1, 'murali', 'abc');

exception



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;

e.g.

'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

e.g. declare
'no';
me';
';
begin
 z := '&x' + '&y';
 dbms_output.put_line (z);
exception
);
 when value_error then
 dbms_output.put_line ('enter proper
data only');
end;
/ * ~~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

3.12
Page

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.

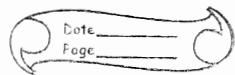
e.g.

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;
/

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



* 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



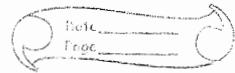
e.g.

```
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;
/
0/p => . , handled through
outer block.
```

—oxo—

User Defined Exception

In oracle we can also create our own exception name & also raise that exception whenever necessary



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.

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.

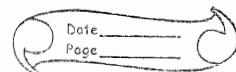
Q Write a PL/SQL program raising userdefined exception today.

⇒ declare

a exception;

begin

if to_char(sysdate, 'DAY')
= 'WEDNESDAY'



exception

when a then

```
:d  
red  
ndler  
SQL
```

dbms_output.put_line ('my exception
raised on wednesday');
end;

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

ising

exception

when a then

dbms_output.put_line ('salary already
high');

end;



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 emename 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
char2(10);
a exception;

begin

select en

if v-empno > 10 OR

v-empno < 1 then

raise a;

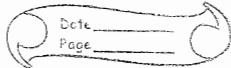
else

select ename into v-ename ~~where empno = &empno;~~

From test

where empno = ~~v-empno;~~ v-empno;

dbms_output.put_line(v-ename);



using
'or
>10
option
terminated

```
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 ~~non~~ executable
Whereas when we / when condition is true then display message & also control goes to remaining executable section

Date 29/10
Page No.

Testing Exception Propagation through user define exception

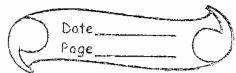
~~Exception raised in executable section :~~

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;
```



)

rough

ble

rk)

using outer blocks ') ;
end;

/

Exception raised in exception section

In PLISQL 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

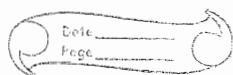
when z2 then

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



Note:- In oracle we can also raise predefined exception explicitly by using raise stmts.

syntax :-

raise predefined_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) Un-Named Exception

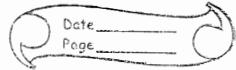
In oracle if you want to handle other than oracle-20 pre-define exception name error then we are using un-named exception.

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 () Fⁿ. This Fⁿ accepts 2 parameter
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 () Fⁿ declared in declare section of PL/SQL block.



)
declare

> begin

it to
> insert into emp (empno, ename)
values (null, 'murali');
end;

/

ora-01400 | cannot insert NULL into
empno

e
by
se
b
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~~ cannot
insert null values in not null
column');

end;

/

npiler
sing
r
option

F'n

e.g.

> begin
 delete from dept where deptno=10;
end;

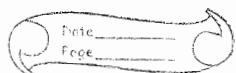
error : ora - 02292

> 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;
 /
 ————— oxo —————

30/10

Q. Write a PL/SQL program by using exception_init function handled ora - 2291 based on emp, dept tables.

> declare
 a exception;



insert into emp (empno, deptno)
no=10;
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;
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:



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

'her

FRI';

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'n 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 :



```
> 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;
```

1

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



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 functn.

```
>declare
  i emp%rowtype
begin
select * from emp where deptno
= &deptno;
dbms_output.put_line(i.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



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;
```

1

Note:- Generally we are not allowed



can
27
variable declaration section of PL/SQL block and then assign these function return value into variable and then only these variable are used in SQL stmt.

tno 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.

at
> 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_errmsg);
end;

31/10
Page

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 sqlerrm function within executable section of PL/SQL block.

e.g.

begin

```
dbms_output.put_line (sqlerrm (sqlerrm (sqlcode)));
```

```
dbms_output.put_line (sqlerrm (100));
```

```
" " " (sqlerrm (1));
```

```
" " " (sqlerrm (-1722));
```

```
end;
```

Ora-0000: normal successful completion

Ora-01403: no data found

~~0000~~ 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



Always one time compilation program units improve performance of application.

Using a Editor write a Procedure For Solving particular task.

show ↑
errors Using SQL plus tool loading
Procedure into Oracle database.

↓
Compiled

↓
Parsed

↓
Execute

Every procedure has ↗ two parts

- 1) Procedure specification
- 2) Procedure body.

In Procedure specification we are specifying name of procedure & type of parameters. Whereas



syntax :-

Procedure : { create [or replace] procedure
use specification }
procedurename (formal parameter)
is / as
variable declaration, cursor declaration,
user defined exception ;
begin
—
—
procedure
dume
body
—
—
end [procedurename];

→ Formal parameter syntax :

parametername [mode] datatype
| → in
| → out
| → in out

→ View Errors

Syntax : → show errors ;

→ Executing Procedure

Syntax :

- r) 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);`
- e) 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
vENAME varchar2(10);`



```
dbms_output.put_line('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;
```

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 (Deptno 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 the
2020.
Procedure parameters are
used to pass the value into proced-



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 :

Modes specifies purpose of parameter. Oracle procedure having 3 types of modes

- i) in
- ii) out

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('a record
inserted through');

end;



** Out Mode :

Out mode is used to return values from the procedure. Out parameter behaves like a 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)
④ 2 (using anonymous block)

Method 1 (using bind Variable)

Date _____
Page _____

Z
64

Method 2 (Using anonymous block)

```
> declare  
x number(10);  
begin  
p1(7,x);  
dbms.out.put_line(x);  
end;  
/
```

Q. Write a pl/sql proc for passing emp name 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 ('SMITH', :x);
> print x

x
2100

method 2 (using anonymous block)

```
declare  
  x number(10);  
 begin  
   p('FORD', >);  
   p('FORD', x);  
   dbms... (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 pl (

p_deptno in number

p_dname out varchar2

p_loc out varchar2)

is

begin

select dname, loc into



execution

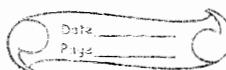
1) using bind variables

```
> variable a varchar2(10);
> variable b varchar2(10);
> exec (10,:a,:b);
> print a b;
```

2) using anonymous block.

```
> declare
  x varchar2(10);
  y varchar2(10);
begin
  p1 p1(20,x,y);
  dbms_output(x||y);
```

2. 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 epl
(p_deptno in number,
p_count out number)
is



select count(*) 1 into p_count
From emp
where deptno = p_deptno;
end;
/

- i) 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

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 recreated. To overcome this

Date _____
Page _____

>create or replace procedure
pl(p_ename in varchar2, p_sal out
nocopy 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 'inout' keyword.

syntax:

parametername in out datatype(size)

create or replace pl procedure
pl(a in out number)
is
begin
a := a * a;
end;
1

execution

method 1) (using bind variable)

>variable z number;

>exec :z := 8;

>exec p1(z);

z

64

method 2 (using anonymous block)

>declare

a number(10) := & a

begin

p1(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.

> recreate or replace procedure p1
(a in out number)



method 1

```
>variable z number;  
> exec :z := 7.902  
> exec pl(z);  
z  
2.000
```

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.

→ 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
  p1(p-deptno, $ in number,
      p-dname in varchar2,
      p-loc in varchar2 default 'hyd')
  is
    begin
      insert into dept values (p-deptno,
      p-dname, p-loc);
    end;
  /
>exec p1 (5,'z');
```

Autonomous Transactions

are independent transaction



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

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

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.

Session 1

Main transaction

```
>insert into test values(1);  
>insert into test values(2);
```

child Transaction (using autonomous)

>declare

```
pragma autonomous transaction  
begin
```

```
loop for i in 3..10
```

```
insert into test values(i);
```

```
end loop;
```

```
commit;
```

```
end;
```

procedure

;

then

re

be

base.

e

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

```
1...10
```

```
> rollback;
```

```
>select * from test
```

```
3...10 ;
```

Session 2

```
>select * from test
```

```
no rows selected
```

```
(before child trnx )
```

* authid current user :

When a procedure having authid current user clause then owner of procedure only allowed to execute that procedure.

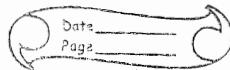
Another user not allowed to execute that procedure if they are giving permission also.

Generally in oracle database whenever we are reading data from table & also perform some table operation then only data security point of view some users 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-empno
number)

authid current_user

is

v-ename varchar2(10);

v-sal number(10);

begin

select ename, sal into v-ename,
v-sal from emp

where empno = p-empno;

dbms_output.put_line(v-ename || '
|| v-sal);

end;

/

> grant execute on pl to manali;

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.

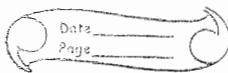
inner procedure

```
create or replace procedure  
pl(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 end;
```

1

outer procedure

```
create procedure p2
```



Date _____

Page _____

exception

when others then
d

→ We can also drop procedure by
using
drop procedure procedurename;

-c
ero)

no. 1

Function

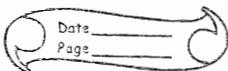
Function is a named PL/SQL block which is used to solve particular task & also function must return value.

Function also having 2 parts
1) Function specification
2) Function body

Note:-

In Fⁿ specif we are specifying name of function & type of parameter whereas in Function body we are solving actual task.

specification } create or replace Function
 Functionname (Formal parameter)
 return datatype
 is/as
 variable declaration, cursors,
 userdefined exception
 begin
 =>
 return Expression ;
 [Exception]
 =>
 =>



Executing a Function

PL/SQL

live
action

2
ob

Method 1 : (using select stmt)

> select Functionname(actual parameter)
From dual;

Note: When a Function does not have or
are
n&
s
solving.

Method 2 : (using anonymous block)

> begin
varname := Functionname(actual
parameter);

s,

e.g.

> create or replace Function

F1(a varchar2)

is → return varchar2

begin

return a;

end;

Q711



Method 1

> select F('hi') From dual ;

method (Anonymous block)

> declare

a varchar2(10);

begin

a := F1('hi');

end dbms_output.put_line(a);

end;

1

Q. Write a PL/SQL stored Function

For passing no. as a parameter

that return a message either

even or odd



> create or replace Function

F1(a in number)

return varchar2

is

begin

if mod(a, 2) = 0

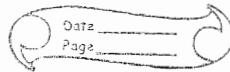
then

return 'even' ;

else

return 'odd' ;

end if;



Method 1 (using select)

```
> select f(4) from dual;  
even
```

Method 2

```
> declare  
(a);      x varchar2(10);  
begin  
  x := f(7);  
  dbms_output.put_line (x);  
end;
```

Method 3 (using bind variable)

```
> variable z varchar2(10);  
> begin  
  :z := f(8);  
end;
```

```
> print z;
```

z

even

SRI RAGHAVENDRA XEROX

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

Method 5

```
>begin  
dbms_output.put_line(f1(8));  
end;  
/
```

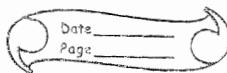
We can also use user defined function in DML stmts.

```
>create table test(msg varchar2(10));  
>insert into test values(f1(8));
```

DML statements are used in 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_empho number)  
return number  
is
```



```
delete from emp where empno =  
      p.empno;  
      v.count := sql%rowcount;  
      dbms_output.put_line (v.count);  
end;  
/
```

ned

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 opr  
inside a query (select stmt).
```

tion
meter
in
er.
on
o

```
> declare  
  a number(10);  
 begin  
   a := f1()  
   dbms_output.put_line ('@ a');  
 end;  
/  
0
```

Q. Write a PL/SQL stored function which returns max(sal) from

autonomous transaction within user defined Functions.

7C
E

e.g.

> create or replace Function

F1 (p_empro number)

return number

is

v_count number(10);

pragma autonomous_transaction;

begin

delete from emp where

empno = p_empro ;

v_count := sql%rowcount;

commit;

return v_count;

end;

/

Ir
Fu

&

F

b;

>

7x

> ~~se~~ select F1(1) From dual;

F1(1)

0

311

Q. W.

Fc

js

F

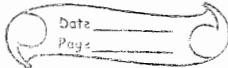
C

F

* select into clause used function →

Q. Write a PL/SQL stored Function which returns max(sal) From

i:



h1n > create or replace function f1
ft return number

is

v\$al number(10);

begin ^{more}
select (sal) into v\$al 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;

911

Q. Write a PL/SQL stored Function For passing emp-name & return job of employee based on passed parameter.

→ create or replace function f1(p_ename)



select job into v-job From
emp where ename = p-ename;
return
job; exception
when no-data-found then
dbms_output.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 f1('abc') From dual;
u r employee does not exist.

Q. Write a PL/SQL stored F1
For passing empno as

301
2012

then 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-sal max(sal)

select v-sal, v-deptno

select sal, deptno ~~from~~

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 ;

F

create or replace function Q. In

F1 (p-empno number) d

return number In

is G

v-sal number(10); →

v-avgSal number(10);

v-deptno number(10);

begin

select sal, deptno into v-sal, P

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

return 1;

else

return 0;

end if ;

exception

when no data found then

return -1;

end;

!

select F1(7002) from dual;



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;

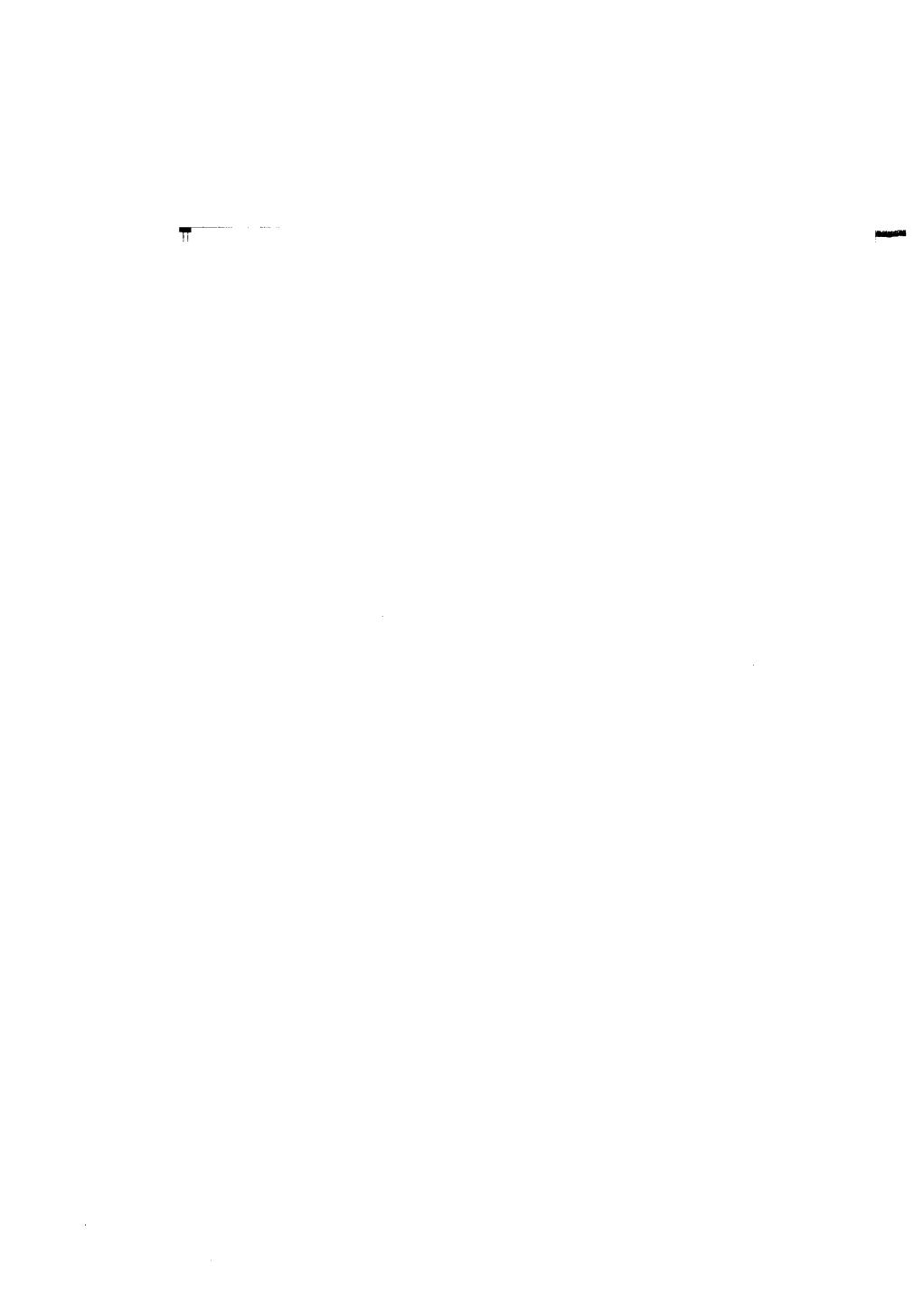
Q. Write a PL/SQL stored function for passing empno, date as parameter then return no. of years that employee is working from emp based on passed date

→
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);



Date _____
Page No. _____

```
> x
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



> create or replace function tax(p.empno)
return number
is

v_sal number(10);

annsal number(10);

x number(10);

begin

select sal into v_sal from emp

where empno = p.empno;

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 Fⁿ if u want to

16/11



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;

vt t1;

v boolean;

begin

vt(1) := 10;

vt(2) := 20;

vt(3) := 30;

in
 z := v-t.exists(3);
 if z - true then
 dbms_output.put_line('u, n
 requested index exists || ');
 || v-t(3));
 else
 dbms_output.put_line('u n
 requested index does not exists ');
 endif;
 dbms_output.put_line(v-t.First);
 → 1
 for i in v-t.First .. v-t.last
 loop
 dbms_output.put_line(v-t(i));
 end loop;
end;

on next
→ (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.delete(3);



O/P

SMITH

ALLEN

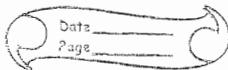
ora-1403 no data

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
```



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

Date: _____
Topic: _____

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.

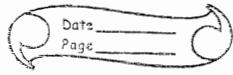
This is an user defined type so we are creating a two step process i.e.

1st 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();
→ constructor name

> declare



```
s begin
into v_t(500) := 30;
also dbms_output.put_line(v_t(500));
end; /
30
```

Index by table are basically sparse i.e. no need to allocate memory explicitly upto those indexes.

```
> declare
  type t1 is table of number(10);
  v_t t1 := t1();
begin
  v_t(500) := 30;
  dbms_output.put_line(v_t(500));
end;
/
```

Error : Subscript beyond count.

Nested table are basically dense i.e. we must reserve memory explicitly before we are storing actual data.

```
> declare
  type t1 is table of number(10);
```

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 ..
```

```
loop
```

```
v_t.last
```

```
dbms_output.put_line(v_t(i));
```

```
end loop;
```

```
end;
```

e.g.

```
> declare
```

```
type t1 is table of number(10);
```

```
v_t1 t1;
```

```
v_t2 t1 := t1();
```

```
begin
```

```
if v_t1 is null then
```

2)

3)

500);
else
dbms_output.put_line ('v_t1 is
not null');
end if;
if v_t2 is null then
dbms_output.put_line ('v_t2 is
null');
else
dbms_output.put_line ('v_t2 is
not null');
end if;
end;
end(10);
,50);
o/p

v_t1 is null
v_t2 is not null

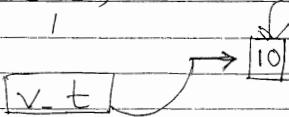
(i) 1) v_t1 t1;

[v_t1] → null

2) v_t2 t1 := t1();


[v_t2] → [] --- . . .

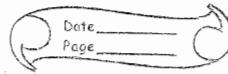
begin
v.t.extend ;
v.t(1) := 10 ;
dbms_output.put_line(v.t);
end;



e.g. > declare
type t1 is table of number(10);
v.t : t1(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



```
v-t.delete;  
dbms_output.put_line(v-t.count);  
end;  
1
```

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));

18/11



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;

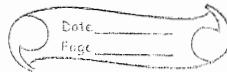
vt(n) := i.ename;

n := n + 1;

end loop;

For i in vt.first

When we are transferring data from table into nested table



because bulk collect clause internally reserved memory in nested table.

declare

type t1 is table of varchar2(10);

v.t t1 := t1();

begin

select ename bulk collect into
v.t from emp;

for i in v.t.first .. v.t.last

loop

dbms_output.put_line (v.t(i));

end loop;

end;

/

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



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.

→ declare

type t1 is varray of varchar2(10);
v.t t1 := t1();

begin

select ename bulk collect into
v.t from emp
where rownum <= 10;

loop → For i in v.t.first .. v.t.last
dbms_output.put_line(v.t(i));
end loop;
end;



Diff betn index by table
nested table, varray.

index by table	nested table	varray
1) This is unbound table having keyvalue pairs	This is unbound table h.	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 numbers & 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, index2)	5) Nested table having exist, delete, extend, trim	5) Varray has exists, limit, extend, trim, first, last, prior,
---	---	--

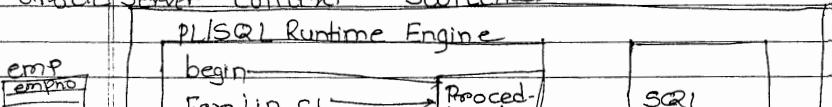
Date _____
Page _____

3 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 these 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 1st 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

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.last  
update emp set sal =  
sal+100 where  
empno = v_t(i);  
end;
```

Procedural Stmt Executor

SQL

Engine

Bulk bind is a 2 step process

- 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

Date _____
Page _____

3) DML... returning ... into clause. 2)

1) 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;

vt t1;

begin

select * bulk collect into vt
From emp;

for i in vt.first .. vt.last

loop

dbms_output.put_line(vt(i).ename);

end loop;

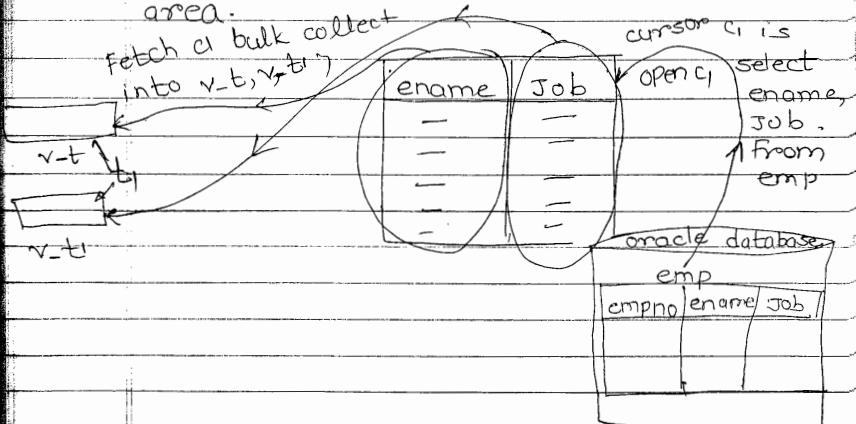
end;

/

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 emp%rowtype(10)

Fetch c1
into v-t
v-tl

```

cursor c1 is select ename,job
from emp;
begin
open c1;
Fetch c1 bulk collect
into v-t, v-tl;
close c1;
for i in v-t.first .. v-t.last
loop
dbms_output.put_line(v-t(i) || ' '
|| v-tl(i));
end loop;
end;
/

```

v-t
v-tl

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

into v-t,
v-t1)

Cursor Cl
Owner Object-
name

Cursor Cl is select
open cl owner, object-
name from
[all_objects];

data
dictionary

1st

.. ..
v-t t1
...
v-t1

all-objects
owner | object-name

e.g.

>declare

type t1 is table of all-objects.

object-name% type

index by binary-integer ;

v-t t1;

v-t1 t1;

cursor Cl is select owner,
object-name from all-objects ;

z1 number(10);

for r in cl loop

sing

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

20/11/15

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.

11'> declare

type t1 is table of number(10);
index by binary integer;

v-t t1;

begin

update emp set sal = sal*100

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 sqlengine
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



ed
'11
2st

Forall indexvarname collectionvarname.
First .. collectionvarname.last
DML stmt where columnname =
collectionvarname (indexvarname);

) ; Using Forall stmts we can perform
bulk updates, bulk delete, bulk inserts.

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;

;

with
bulk
F
all
is
1

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



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;

empno

736°

756°

790°

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

after Oracle 10g (indices of)

empno	First	update all	empno	(1):-
7369			7369	
7566		rows in	indices	7369



update all rows within a array
inorder of array variables
(or) indexes.

solt

declare

type t1 is table of number(10)
index by binary_integer;
vt t1;

begin

select empno bulk collect into
vt from emp;

forall i in indices of vt
update emp set sal = sal + 100
where empno = vt(i);
end;

/

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.

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
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. GDAC, 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

=

d

ress

l



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 stmt typically executes multiple DML stmts whenever an exception occurs in one of those DML

- 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

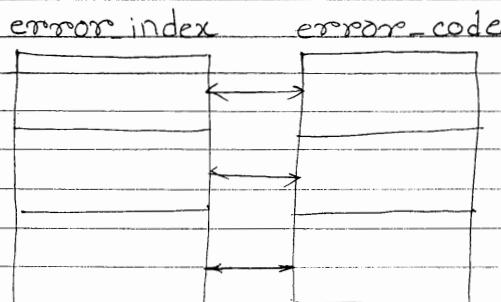
sql%bulk_exception index by table. This index by table having two fields these are error index , error code.

error_index :-

error_index stores
index no. of errors.

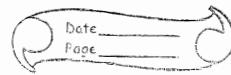
error_code :- It stores error
number (only positive)
of error occurred in bulk bind
process.

sql%bulk-exception



If you want to handle
bulk exception then we must
following 2 steps.

Step 1) Store ~~but~~ no. of exceptions in a variable by using count collection method sql% -



by using following syntax.

syntax

varname := sql%bulk_exceptions.count;

Step2

Before we are using this process
we must use save exception clause
in forall stmts.

e syntax :

forall indexvarname in
collectionvarname.first ..
collectionvarname.last save exception

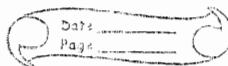
DML stmts where columnname =
collectionvarname(indexvarname);

e
g.

>create table target (sno number(10)
not null);

ept-
>declare

type t1 is table of number(10);



Forall i in vt.first .. vt.last
insert into target values
(vt(i));
end;

/

error: ora-1400: cannot insert →
null into not null.

solution

```
>declare
  type tl is table of number(10);
  vt tl := tl(10, 20, 30, 40, 50);
  z number(10);
begin
  vt(2) := null;
  vt(3) := null;
```

Forall i in vt.First .. vt.last
save exception
insert into target values
(vt(i));

exception

when others then

```
z := sql%bulk_exceptions.count;
dbms_output.put_line(z);
```

Last

7 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);

2);

sql%bulk_exceptions(i%indexvarname).
error_index

sql%bulk_exceptions
(i%indexvarname).error_code

Last

7 declare

type t1 is table of number(10);
l vt t1 := t1(10, 20, 30, 40, 50);

z number(10);

begin

v_t(2) := null;

v_t(3) := null;

unt

Date _____
Page _____

```
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;
```

Date 23/11

Page

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, binary_integer, index_by_table)
varname varname varname

table_to_comma :-

It is used to convert index by table values into comma separated strings. This procedure also accepts



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);`

end loop;
end ;
1

Q. Write a PL/SQL program to retrieve all dept names From dept table display those dept name int 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

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

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 create 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;



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 cursor varname for
select stmt;

24/11
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.ename||' '||

rs;

Q. Write a PL/SQL program by using ref cursor whenever user entered dept no. 10, then display 10th dept details
 From emp dept no. 20 → display 20th .. "
 From dept.

→ declare

```

type t1 is ref cursor;
v_t t1;
v_deptno%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;

Date: _____
Page: _____

```
>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 (i.ename||' '
      || 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.deptno||' '
      || j.dname || ' ' || j.loc);
  end loop;
else
  dbms_output.put_line ('bad input');
end;
```

Date 25/11
Page _____

Oracle 9i introduced sys_refcursor predefined type in place of weak ref cursor.

refcursor varname sys_refcursor;

e.g.

declare

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.empno' || i.sal);

end loop;

end;

1

- * Passing ref cursors as parameter to sub program.

Passing sys_refcursors as in parameter to stored procedure

Q. Write a PLSQL stored procedure



>create or replace procedure pl
(pt.inrefcursor sys_refcursor) *

is

i emp%.%avtype;
begin

for i in pt

loop

dbms_output.put_line(i.empname||
11 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



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

ve

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;



Q. Write a PL/SQL stored function by using sys_refcursor as return type which returns emp details.

→ create or replace function f1
~~return sys_refcursor;~~
return sys_refcursor;

is

→ v.t sys_refcursor;
begin

open v.t for select * from
emp;

return v.t;

end;

Execution

>select f1 from dual;

Q. using pkg

>create or replace package pj1
is

type t1 is ref cursor;

return emp%rowtype;

tion

```
procedure p2 (v_t2 out t2);
end;
/
> create or replace package body p1
is
procedure p1 (v_t1 out t1)
is
begin
open v_t1 for select * from emp;
end p1;
procedure p2(v_t2 out t2)
is
begin
open v_t2 for select * from dept;
end p2;
end;
```

/

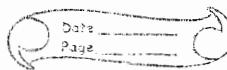
Execution (using bind variable)

> variable a refcursor;

> variable b refcursor;

> exec p1.p1(:a);

> exec p2.p2(:b);



* In oracle we are not allowed to create ref cursor variable in packages.

e.g.

```
> create or replace package p1  
is  
type tl is ref cursor;  
vt tl;  
end;  
'
```

Package created with compilation error.

```
> show errors;  
cursor variable cannot be  
declared as part of package.
```

Date 26/11
Page

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

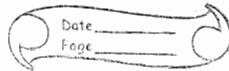
—

Exception

—

end (Function name);

begin



> declare
procedure P1
is
begin
dbms.output.put_line('Local Proc');
end P1;
begin
P1;
end;
/
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;

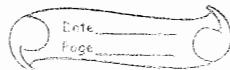
> exec P2;
local procedure



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;
```



* Passing index by table as in parameter
to local procedure.

```
> declare
  type t1 is table of emp%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 emp%rowtype;
```

Date _____
Page _____

```
begin
select * bulk collect into p-t
From emp;
end pl;
begin
p1(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 p1( 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;
```

Date _____
Page _____

Function F1 returns t1
is

v_t t1;

begin

select * bulk collect into v_t from
emp;

return v_t;

end F1;

begin

? pl(F1);

1

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

Date 27/11
Page

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;
```

```
1
```

e.g.

```
> create or replace package body pi1
```

Date _____
Page _____

```
> create or replace package body pj1
is
procedure p1
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 p1;
end;
> exec pj1.p1;
```

--- o ---

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 syntax

```
create or replace directory directoryname
as 'path';
```

privilege to user otherwise oracle server returns error.

Syntax :

>grant create any directory to username;

e.g. > conn sys as sysdba;

Enter password : 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;
password: sys

> grant read, write on directory XYZ
to scott;

> conn scott/tiger;

6 also if you want to read from file then we are using get_line procedure from utl package.

step 4

syntax

writing 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.

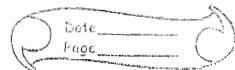
Filepointer variablename utl_file.File_type;

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 : filepointervariable := utl_file.Fopen ('aliasdirname', 'filename', 'mode');
w - write, r - 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(filepointervariable,



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
```

```
    FD := utl_file.Fopen('XYZ', 'FILE1.BRF',  
                          '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

Date 28/11

Page

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 and nope



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.file_type fopen ('xyz',
 'File2', 'w');

 for i in c1

 loop

 utl_file.putf (fp, '%s\n',
 i.ename);

 end loop;

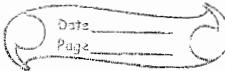
 utl_file.Fclose (fp);

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 ?



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, iename || ' ' ||  
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



within Fopen Function From
utl file package

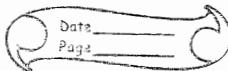
syntax

utl_file.get_line(filepointer varname,
buffervariblename);

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
File1.txt



```
>declare
  fp utl_file.file_type;
  z varchar2(200);
begin
  fp := utl_file.Fopen('XYZ', 'file2.txt',
    'r');
loop
  exit when fp is null;
  utl_file.get_line(fp, z);
  dbms_output.put_line(z);
end loop;
utl_file.Fclose(fp);
end;
/
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 reaches end of file . To overcome this problem we

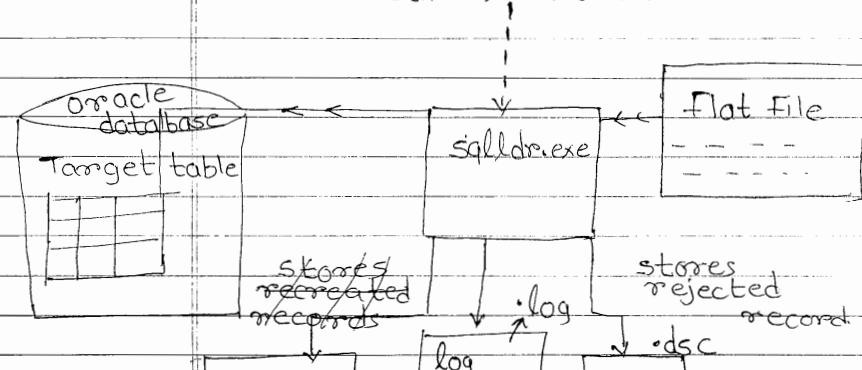
SQL * Loader

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, error 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 ~~ctr~~ 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.



Variable length record Flat file

A Flat File which having a delimiters is called variable length record.

e.g.

- 1 101, abc, 7000
- 102, xyz, 4000
- 103, jjj, 9000.

Fixed Length record Flat file

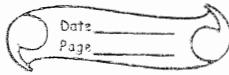
A Flat File which does not have delimiter is called fixed length record Flat file.

e.g.

- 101abc7000
- 102xyz4000
- 103jjj9000

Control File

Always sqlldr executes



control file 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 ^{for}

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'



within infile clause & also use "begin data" clause in the above flat file data within control file itself.

e.g.

load ~~is~~ 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 ~~is~~ insert.



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'

// file1.txt

101, abc, 2000
102, xyz, 3000
103, kkk, 8000
104, jjj, 9000

no. of

> create table target(empno number(10),
ename varchar2(10), sal number(10));

→ control file

load data
infile 'C:\file1.txt'

insert
into table target
Fields terminated by ','
(empno, ename, sal)

save murali.ctl

→ Execution

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.

eg

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:

2 Date
Page

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

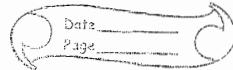
>create table target (empno number(1),
loc varchar2(10));

control File

load data
infile 'C:\File1.txt'
insert

into table target

Filler is replaced by ''



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

Page No. 1
Date: 10/10/2023

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

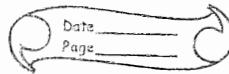
101, abc, 1000

102, xyz, 4000

103, kkk, 2000

104, nnn, 9000

>create table target (empno
number(10), ename, varchar(10),
sal number(10) check(sal > 2000));



insert

into table target

Fields terminated by ','
(empno, ename, sal)

~~File1.dat~~ → ~~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 ~~at~~ number(10),
ename varchar2(10), sal number(10));

insert

into table target

Fields terminated by ','

trailing nullcols

(empno, 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

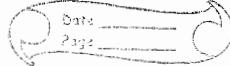
101, abc

'102', xyz

'103', klc

104, nnn

> create table target (empno 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)

o/p

empno ename rno

101 xyz 1

104 mnj 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 is created automatically as same name as flat

Discarded file extension is .dsc

Syntax :

discardfile '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'

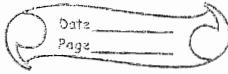
discardfile 'c:\file3.dsc'

insert

into table target when deptno = '10'

Fields terminated by ','

(empno, ename, deptno)



When clause condition value must be specified within single quotes.

In when clause we are not allowed to use other than =, <, > != relational operators.

In when clause we are not allowed to use logical operator 'or' but we ^{are allowed} 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 functions Funcionality within " " & also we must use ':' colon operator in front of the columnname within function Funcionality.

Syntax :

=10'

columnname "Functionname (:columnname)"

File1.txt

101 abc.m

Page No. 1

```
> 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"

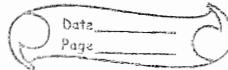
e.g.

File1.txt

101, abc, 130507

102, xyz, 250809

103, kkk, 171203



ber(10),
am(10);

```
>create table target (empno number(10),  
ename varchar(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



File1.txt

101

102

103

104

105

>create sequence S1;

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

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^n then we are not allowed to use SQL Loader datatype in place of SQL Loader

31/12/15

e.g:-

File1.txt

101 abc 3000

102 xyz 2000

103 zzz 9000

104 yyy 7000

number

sn

control

```
> create table target (empno  
number(10) , ename varchar2(10),  
sal number(10) );
```

control file

load data

infile 'C:\File1.txt'

insert

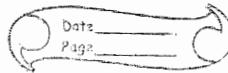
into table target

(empno position(01:03) integer external,

ename position(04:06) char,

sal position(07:10) integer

external)



number(10) → varchar²

sno	value	Time	col1	col2	col3
	default value	HH:MM:SS	col1/100	upper()	to_date()
50					

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) ~~ex integer external~~
~~/100~~, ":col1/100",
col2 position(06:11) "upper(:col2)",
col3 position(12:17) "to_date(:col3,
'DDMMYY'))"

begin data

10000aaaaaaaa050103

20000bbbbbbbbb180905

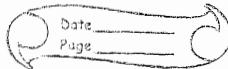
In oracle we are not allowed to use :colon & also we are not allowed to

P Page

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.



Triggers

Trigger is also same as stored procedure & also it will automatically invoked whenever DML opn 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

Trigger timing

Trigger Event

create / or replace trigger triggername
specification
before / after insert / update / delete
on table

Page No. 4/12

[declare]

variable declaration, cursor,
user defined exception
begin

[exception]

end;

Trigger
body

DIFF b/w statement level &
row level.

statement level trigger

> create or replace trigger t1
after update on emp
begin
dbms_output.line('statement
level');
end;

/

ex-1

> update emp set sal = sal + 100
where deptno = 10;

> update emp set sal = sal + 100
where deptno = 20;
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.

Date _____
Page _____

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;

:old → | 1 | murali | 2000 | :old.sal
:new → | 1 | murali | 5000 | :new.sal.

Deletion

>delete From emp where empno=1;

:old → | 1 | murali | 5000 |
:old.sal

QUESTION

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 t1
before insert on emp
for each row

```
begin
if :new.sal < 5000 then
raise application_error (-20123,
'salary should be above 5000');
end if;
end;
```

1

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.
```

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 new.sno &

then

raise application error (-20789,

'we cannot insert duplicate data');

end if;

Date 5/12

Type

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

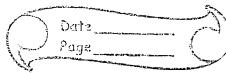
raise application_error (-20123,

'We cannot insert duplicate data');

elsif v_count = 0 then

dbms_output.put_line('ur

ssed is good')



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.
or.

Company does not allow any bonus to job as analyst.

→ create or replace trigger t_wz
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

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;

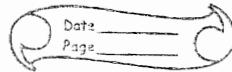
/

Testing

update emp set sal = sal + 100
where ename = 'KING';

Q.

→



Q. Write a PL/SQL row level trigger on emp^{dept} 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;
```

1

Testing

```
>update dept set deptno = 1
where deptno = 10;
```

Q. Write PL/SQL row level trigger on emp^{dept} table whenever user deleting data from emp table then automatically deleted data stored in another table.

→ create table tempt (empno number(10)

Page 07/12

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 true 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 trigger

Example:-

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;
/
>create table target (empno number(10),
oldsalary number(10), newsalary number(10),
col4 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. To create if


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));

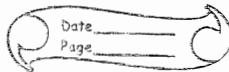
> create sequence s1
starts with 1;

> 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



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;

>create or replace trigger tv2



```
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

```
> /
```

" " " :ABCpqr

```
> select * From test;
```

sno	name
ABC0000000001	murali
ABC0000000002	xyz
ABC0000000003	pqr

```
>create or replace trigger tr2
```

```
[after] insert on test
```

For each row

begin

```
select s1.nextval into :new.sno
```

```
From dual;
```

```
end;
```

'Q)

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;

8/12

A Whenever we are using after timing DML transactional values are 1st 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 \$ cell values at in a table, then 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 doesn't 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') = 'SAT' or
to_char(sysdate,'DY') = 'SUN' then
raise application_error('cannot perform
DML operation on '|| to_char(sysdate,'DAY'));
end;

Date _____
Page _____

```
not perform DML oprn');  
end;  
/
```

→ In all databases we are not allowed to use when clause in statement level trigger.

```
create or replace trigger tw1  
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 if last day of month

begin

```
if sysdate = last_day(sysdate) then
    raise application_error (-20123, 'cannot
    perform DML open on last day of month');
end if;
end;
```

/

Triggering Events

In oracle if you want to define different opns 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
```

Q. Write a PL/SQL 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');

endif;

} end;

/

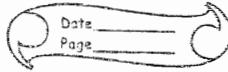
O/p →

e.g.

>create table target (msg varchar2(20);)

>create or replace trigger tr1
after insert or update or delete
on emp

Q



```
1. if inserting then  
2.   z := 'rows inserted';  
3. elseif deleting then  
4.   z := 'rows deleted';  
5. elseif updating then  
6.   z := 'rows updated';  
7. end if  
8. insert into target values(z);  
9. 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

Q Write PL/SQL row level trigger on emp table when no. of rows is greater than

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;`

Execution order of trigger

- 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

>create or replace trigger th4
before insert on test
for each row
begin
dbms.output.put_line('before
row level');
end;
/
Testing

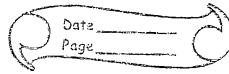
>insert into test values (10);
0/p

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

Piter

Q. Write PL/SQL compound trigger
to display default execution
order of table *

→ >create or replace trigger tkl
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



* Follows clause (11g)

KL
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

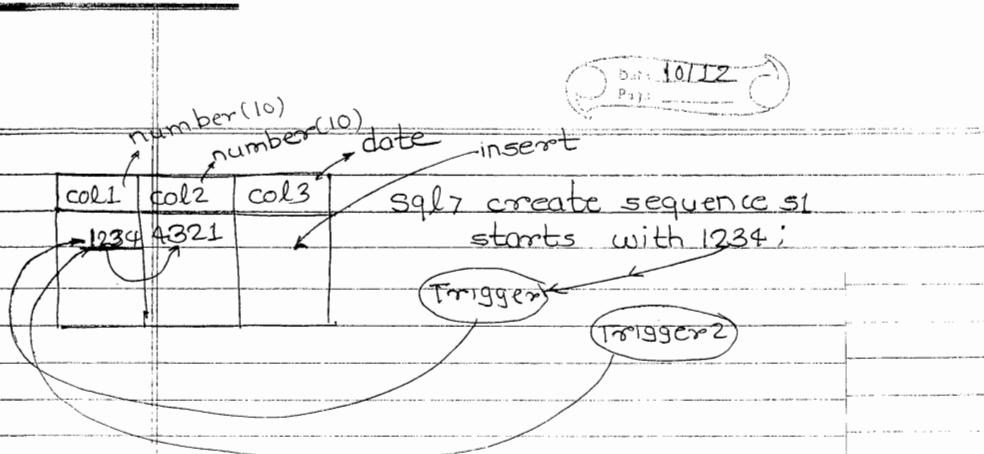
[For each row]

[When condition]

[Follows anothertriggername]

[declare]

begin



>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;

/

```
s1 select reverse (to_char (:new.col1))  
; into :new.col2 From dual;  
dbms_output.put_line ('Trigger 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  
insert into  
before insert on test  
For each row  
begin  
:new.col1 := s1.nextval();  
dbms_output.put_line ('Trigger 1  
executed');
```

Before 11g same level trigger are executed in reverse order of creation.



> create or replace trigger tk2 before insert on test

For each row

cell trans-

Formation in
trigger
requires select
into clause
From dual.

Follows tk1

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.

begin
select count(*) into z
from emp;
dbms_output.put_line('z || ' ||
number of remaining rows');
end;

1

e.g.

>create or replace trigger tk1
after delete on emp

For each row

declare

v_count number(10);

begin

select count(*) into v_count

from emp

dbms_output.put_line(v_count);

end;

1

Trigger Created

Testing

>delete from emp where empno=2;

[ora- 04091]: table SCOTT.EMP is

mutating.

Date: 15/12

PAGE

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.

That's why in stating session connected

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;  
1
```

* When Condition

When condition are allowed to use a ID level triggers only



When condition must be
a SQL exprn.

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 tg1
after insert on emp
For each row
when (new.empno = 1)
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 tv1

Q 67
P 13

```
>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 triggers are created by database administrator.

In oracle we are creating trigger in two level

- 1) Schema level
- 2) Database level

Syntax :

```
>create or replace trigger triggernam  
before/after create/drop/alter/  
truncate/ rename
```

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 triggers 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
```



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 table_name



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;
```

Date: 16/12
Page: 1

~~QUESTION~~ 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 ~~emp~~ emp set sal = sal + 100

where empno = i.empno;

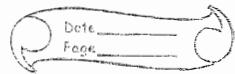
else if i.job = 'SALESMAN' then

update emp set sal = sal - 200;

end if

end loop;

close c1;



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 ^{cursor} then only oracle server establishes locks by default these locks exclusive locks.

where current of

clause uniquely identifying a record because it uses internally around where current of clause is



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 ~~as~~ 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 ^{varchar2} number(10),
sal number(10));

1 rename sal
a 1000

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.



```
> declare
  cursor c1 is select * from test
  for update;
begin
  for i in c1
  loop
    update test set sal = sal + 100
    where current of c1;
  end loop;
  commit;
end;
```

ename	sal
q	2000
b	3000
a	4000
b	5000
c	6000

SRI RAGHAVENDRA XEROX
Software Languages Materials Available
Opp. CDAC, Balkampet Bakery,
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

)
st
if c1%rowcount = 5 then
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 condn 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);

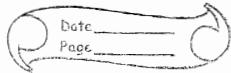
sm
> commit

> select * from test

a	b	c
5	4	

Date _____
Page _____

```
> 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;
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.

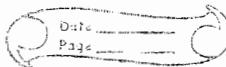
In oracle if you want to store more than 2000 bytes alpha numeric data then we are using varchar2 datatype. This datatype stores upto 4000 byte.

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 'blob' datatype.

Syntax
columnname long

e.g.

```
>create table test (col1 long, col2 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)

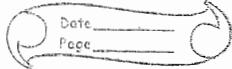
If you want to store more than 2000 bytes of binary data then we are using long raw datatype.
It stores upto 2gb data but there can be only long raw column for table.

To overcome this problem oracle 8.0 introduced blob datatype

syntax

)(columnname long raw

>create table test (col1 raw(2000));



)

any

>create table test1 (col1 long raw);
>desc test1;

x

All database system having 2 types
of LOBS

any

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)

2

are

re

cle

syntax
columnname blob

columnname blob

External large object

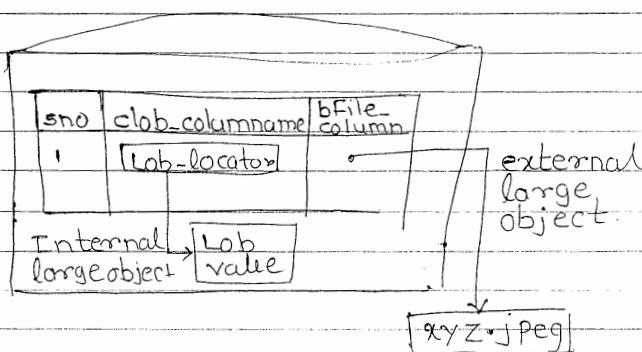
are stored in outside of
database i.e. these objs are stored in
OS file



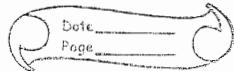
DIFF b/w long, lob datatype
Long Lob

- 1) can contain upto 2gb data 1) can contain upto 4gb data.
- 2) A table does not contain 2) A table can contain more than one long column more than one lob column.
- 3) Subquery cannot select 3) subquery can select a long datatype column. 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



In internal LOB data is stored in within database. In internal lob this pointer is also called as lob locator

data

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),  
d           col2 clob);
```

```
>insert into test values(1, 'abc');  
> "           " (2, empty_clob());  
> , , , , (3, null);
```

```
>select * from test
```

Date 18/12
Page

> select * from test where col2
is not null ;

sno	col2
1	abc
2	

> select * from test where col2 is null ;

sno	col2
3	

Storing large amount of data into
blob column / storing an image
into blob columns.

Step1) create an alias directory
related to physical directory by
using Following syntax.

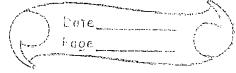
> create or replace directory
directoryname as 'path' ;

> conn sys as sysdba ;
sys

> grant create any directory to scott ;

> conn scott/tiger ;

> create or replace directory xyz as



Step 2 :- create a table by using LOB column

>create table tablename (col1 clob,
col2 blob);

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.

ii) 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 pointers into appropriate lob variable.



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 fⁿ accepts two parameters &
returns bfile datatype.

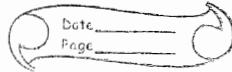


syntax

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.



- v) After opening the file we can load data into lob column by using `fol "loadFromFile"` procedure from `"dbms_lob"` package.
This proc accepts 3 parameters.

syntax

`dbms_lob.LoadFromFile (lobvarname ,
bfilevarname, length of bFile);`

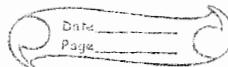
→ IF you want to return length of bfile then we are using `getlength()` function from `dbms_lob` pkg.

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 );
```

> decree

v= clo b clo b i

v_bfile blob;

3 begin

```
insert into test values(1,empty_clob)  
returning col2 into v_clob;
```

```
v_bfile := bfilename('xyz', 'filez.txt');
```

dbmslob.Fileopen(kbFile);

```
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.



1, `> select * from test;`
error: column or attribute type can
not be displayed.

— o/o —

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 ~~ex~~ 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



e.g.

begin

i execute immediate 'create table
k1(sno number(10));'

end;



Q Write PL/SQL program to create
roll.

→ begin

execute immediat 'create roll*er1*';
end;



→ conn sys as sysdba
sys

Retriev Data From dynamic SQL Stmt

Through into clauses we can
retriev data From dynamic SQL
stmt.

Q Write dynamic sql program which is
used to display max sal from emp
declare

→ declare

```
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

tmt
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 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

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;

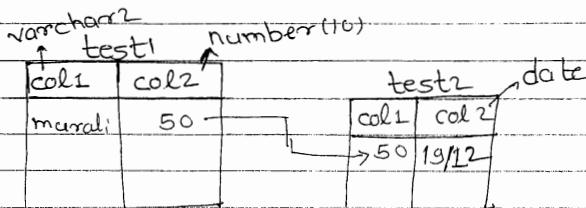
dbms_output.put_line(v-dname || ' ' ||
v-loc);

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 tr1  
after insert on test1
```

For each row

declare

id number(10);

begin

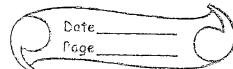
select col2 into id From test1

where col2 = :new.col2;

insert into test2 values(id, sysdate);

end;

1



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 triggers
- 3) Create statement level trigger.

22(10),

solution

```
>create or replace package pc1
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
```



oracle 11g onwards we can also avoid mutating error by using compound triggers. Because compound trigger having global declarations & 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 tri

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 ;

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

>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;

/

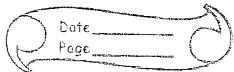
Oracle having two types of
member subprogram.

- 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



Syntax

(10))
create or replace type typename
as object (attrname datatype, ...,
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
(x number, member procedure p1,
member function f1 return number);
,

>create or replace type body circle
as
member procedure p1
is
begin
dbms_output.put_line('my func');



```
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.f1);  
end;  
/
```



With

```
sql> with function F1(a number)
      return number
    is
      begin
        return a*a;
      end;
    > select F1(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;



> 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.

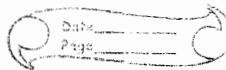
syntax:

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 thots why we must assign value when we ~~declare~~ declare variable in declare section.



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

```
> begin
    For i in 1..10
    loop
        if i=5 then
            continue;
        end;
        dbms_output.put_line(i);
    end loop;
```

ol P

1

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

4

6

7

8

9

10

1

- 5) Oracle 11g introduced pivot() function which is used to display aggregate function value in tabular format & also convert rows into columns.
- 6) Oracle 11g introduced Follows clause in triggers,  compound triggers.
- 7) Variable assignment concept when we use sequences in PLSQL. In this case we are not allowed use dual table.
- 8) Oracle 11g introduced enable/disable clauses within trigger specification.

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