SQL

Class notes

By

Mr . **Murali** sir

SRI RAGHAVENDRA
XEROX
Software languages Material Available
Beside Bangalore Iyyager Bakery .Opp. CDAC, Balkampet Road,
Ameerpet,Hyd

9951596199

Introduced in 2007. ← Oracle 11g → greedy Technology.

10-6

d.b → Collection of files. RS = BC

↳ RDBMS s/w

— oracle also d.b. Server. (oracle is a dbms s/w).

— Servers all are works on Port no.

→ data Store permanently in secondary storage devices. (Hard disk).

— These called dbms s/w.

← To store data Permanently in secondary storage device (Hard disk)

Orcle d.b.

SQL (languages) PLSQL. Dynamic SQL.

↳ Structured Query language (procedure language extension for SQL)

→ DBMS s/w → Relation DB management System.

— Data :- Collection of raw facts.

Student marks → 90 80 70 20 0.

Collection of nos.

names — Sham Jyo

Collection strings.

SQL ↓ is ASCII Language

— Information :-

— Data store :- Data store in DBMS s/w.

→

→ Every org'n deals with lot of data

Data:- Collection of raw facts.

Ex:- (1) Student marks

(2) Customer names.

Information:- when we are processing data we r achieving meaningful results.

Ex:- (1) marksheet.

(2) Invoice of a Customer.

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

(1) books & papers.

(2) flat files.

(3) Data base.

Flat files:- This is a traditional mechanism which is used to store data or information in files

Disadvantages:-

(1) data retrival

(2) data redendancy

(3) data integrity

(4) data Security

(5) data indexing

(6)

10 V Data

To

Sequel

data -

is high

D.F

&

→ 'w

- Caley

Interface

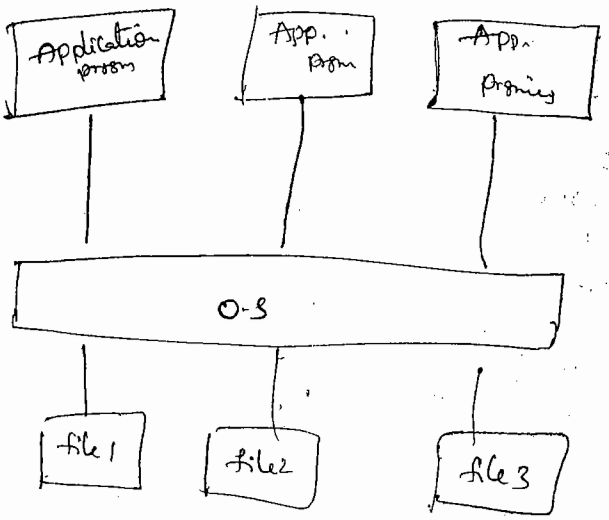
with

Prgmrs

app'n

(1) Data Retrival:-

TO retrive data from data bases we are using Sequel language, where as if we want to retrive data from flat file we must develop Application progm in high level languages.
Java, dot net
C, dot net



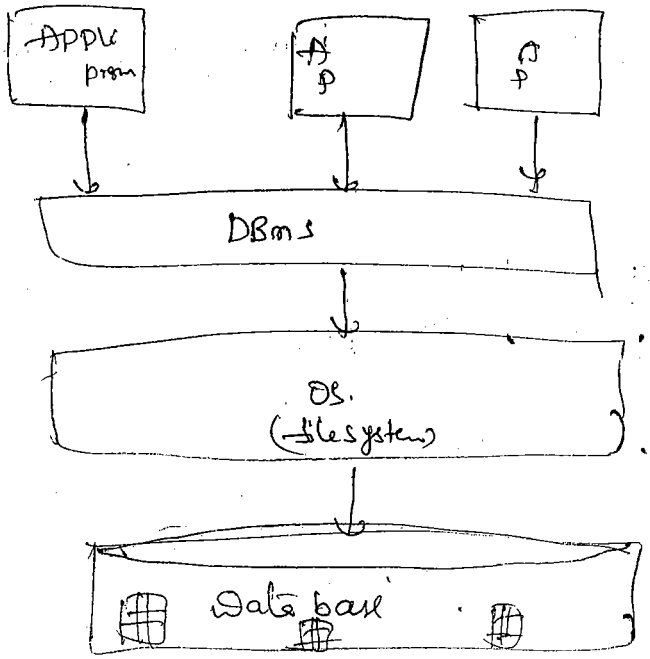
flat file approach.

D.BMS S/wds — oracle, SQL.

D.B - Collection of files.

→ when ever we are loading d.b.m.s s/w's automatically d.b. is created & also automatically end user interface is created that's why end users directly interacting with data base using un interface otherwise application Prgmrs indirectly interacting with a database using appln prgms.

DBMS approach.



Transactions designs all are in Emp no. only.

↓primary key.

— In Dept duplicate not available.

→ unique values contains Dept table.

→ Contains primary key → master table (Dept table).

→ foreign key → child table.
(Emp table)

→ we can't key to delete master table record is it not possible.

Deptno	name	loc
10		
20		
30		

to delete not possible.
bcz we first delete the records in related Emp (childs table record

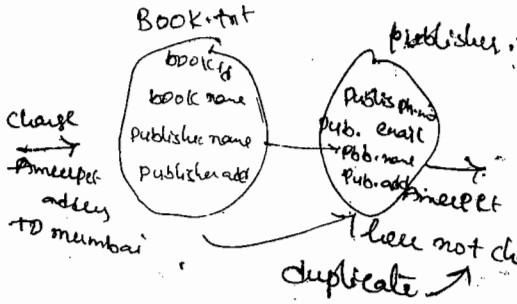
→ On_delete_cascat

when ever delete master table, all deleted in child table

Data redundancy :- (duplicate data) 11-6

Inconsistency is a biggest Problm in flat file.

Consistency data in d.b.m.s s/w



→ Some times we are maintaining multiple copies of the same data in diff. locations.

→ This data is also called as "redundant data".

→ In flat files when we are modifying data in 1 location it is not effected in another location (files) this is called Inconsistency.

→ where as D.B.m.s s/w's (SQL servers) automatically maintains Consistent data through transactions

Ex:- Inconsistency ATM : Transaction 3000, but money not given

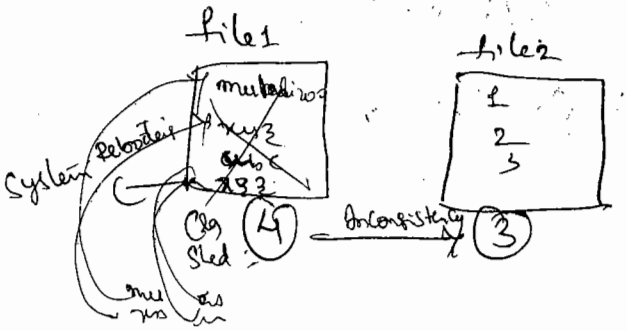
Every transaction have 4 properties

↓

ACID

↓

maintain consistent data



Transaction ⇔ Algorithm.

before Trans + After Tran = Total data remains is same

↓

Consistency Property.

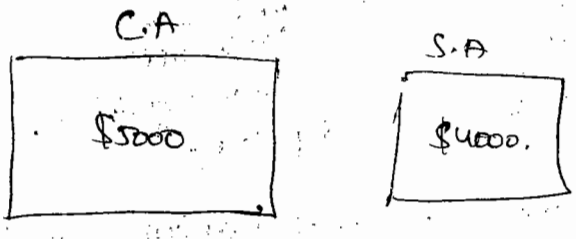
Atomocity → Either all or none.

Roll back:

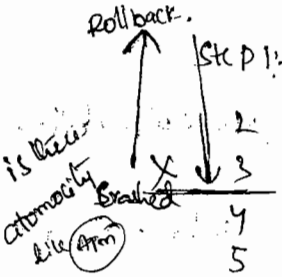
ACID

↓

only Consistency data



Task:- To transfer $1000 from CA → S.A.



Step 1:- Read amount C.A
2. deduct $1000 from C.A
3. Write new amount in C.A
4. Read S.A
5. add $100
6. write new in S.A

→ permanantly store all in d.b. Hard disk is → durability.

↓

(Commit)

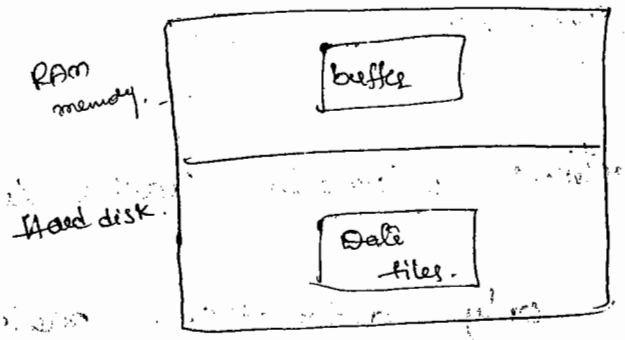
→ Automacity → Rollback. (Cancel)

→ durability → Commit. (OK).

→ Every d.b. server having 2 parts.

(1) Ram memory area
↓
here buffers are there.

(2) Hard disk. (Storage area).
↓
here files are there.
↑ (Automatically saved).



→ Create 1 table. it is stored in H.disk only with out saving also bcz ddl commands automatically saved.

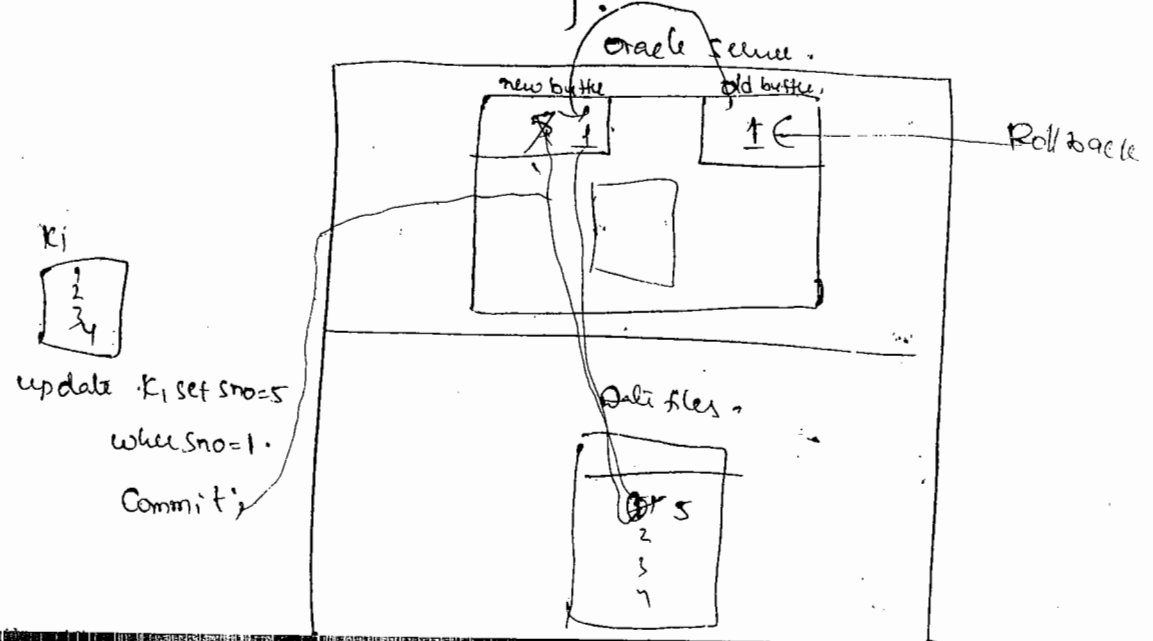
→ Commit:-

is used to save permanently in Hard disk.

→ Save transactions into disk.

→ Rollback:- (Cancel the transaction).

** undo from memory.



→ After Commit, Rollback is not possible.

Commit means → changes in Hard disk permanently.

insert, delete → These possible.

⟹ Every transactions internally having 4 properties These 4 properties only automatically maintains consistent data in databases. These are also called as "ACID" properties.

A → Automacity (either all or none)

C → Consistancy (before, after total data remains same)

I → Isolation (running transactions individually; Trns cannot be overlapped)

D → Durability (Save the transactions permanently in disk).

→ All d.b. products introduced Commit Command based on durability property. and also all d.b. products introduce rollback Command based on automacity property.

→ Reduce the redendancy we having "Normalization"

→ In d.bases, if you want to reduce Redendancy d-b designers uses "normalization process".

(3) Data Integrity:- (Consistency) 12-6

Set of rules are there in projects then set of rules called Business Tools

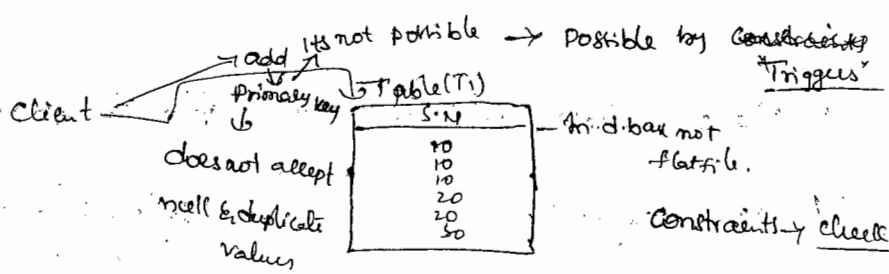
(To maintain proper data) → 2 constraints are there ← triggers, constraints

→ Integrity means to maintain proper data

→ To maintain proper data developers defines set of rules based on client requirement

→ These set of rules are also called as "Business Rules"

→ In d.b's we are defining business rules through "Constraints" (Primary key, foreign key, check) triggers.



→ where as in flat files (Clay) we are defining business rules using Application prgms ('C lay') in high level languages.

(4) Data Security:- It is prblm in flat files (Clay no security)

d.b. provides security internally - role based security

→ Data stored in flat files cannot be secured bcz files does not provides security mechanism.

→ where as d.bases (Oracle) provides "role based security"

Data Indexing :- (B Tree Index, Bitwise Indexing).

↓ very fasting retrival data — Airtel (very fast indexing), BSNL (slow)

→ In databases to retrive data very fastly then we all using indexing mechanism.

→ where as flat-files does not provides indexing mechanism

→ Orgns suffering from flat file approach to store data

→ In 1960 onwards orgns uses dbms s/w's to store data permanantly in Hard disk.

→ DBMS:- (Interrelated data) It is a

→ Collection of prgms (s/w's) internally written to manage database:-

→ It is a place where we can store data in hard disk.

→ i.e. It is an organized Collection of Interrelated data (Emp → dept).

→ once data stored in database it can be integrated and also it can be shared by no. of users.

work in IBM

↑

In "1970" "E.F. Codd" written a paper "Relational model of data for large shared data banks".

→ In this paper E.F. Codd introduced relational data model

i.e, In this model we all store data in the format

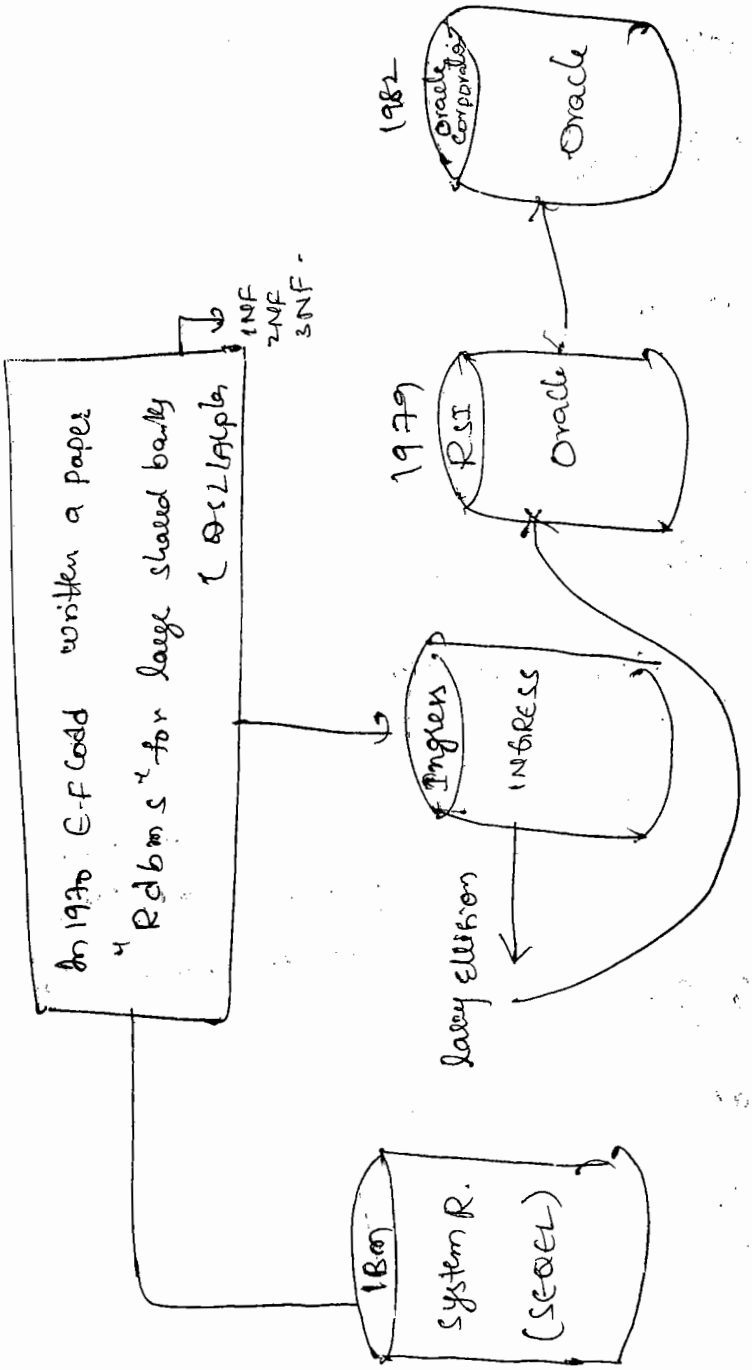
format of table.

→ And also in this paper E.F. Codd introduced 1st 3 normal forms.

→ and also introduced DSL/ALPHA language to operate relational databases.

Now → Oracle → SQL.

IBM 1970's → SystemR → DSL/Alpha (SEQEL)



In 1979 : Oracle Versions. (1) SQL
(2) PL/SQL.

(1) 1979 1st version → 2.0.

(1) first public release

(2) basic sql functionality, joins.

(2) 1983 → oracle 3.0.

(1) Commit, rollback.

(2) rewritten in c language.

(3) 1984 → oracle 4.0.

(1) read consistency

(4) 1985 → oracle 5.0.

→ introduced Client - Server architecture.

(5) 5.1

→ distributed d.b.

up to 6.0 ↔ PL/SQL there is no
PL/SQL there from 6.0.

(6) 1986 → oracle 6.0.

→ introduced PL/sql.

→ row level locks.

(8) 1992 ⟶ Oracle 7.0.

→ Integrity Constraints (foreign key) introduced.

→ stored procedures, stored-fun's

— triggers are introduced.

— d.t. Varchar change into Varchar2.

— Truncate table.

— View Compilation (snapshot) introduced

upto 7.0.
d.types
↓
Varchar only
available.
In 7.0 it
changes
Varchar_2

(9) 1992 ⟶ oracle 7.1.

→ Introduced dynamic SQL.

→ ansi / iso sql92 standard

(10) oracle 7.2.

→ Inline views.

→ In PL/SQL Introduced Ref Cursors

(11) oracle 7.3.

→ bitmap indexes.

→ (reading files, writing file) (Utl_file package)

(12) oracle 8.0 ⟶ 1996.

→ Object-technology (like C++)

→ nested table.

→ Varray

→ Instead of triggers. oracle 8i

(13) Oracle 8i (Internet) ——→ 1997

→ rollup, cube

→ Columns increased per a table upto 1000.

→ analytical functions

(14) oracle 9i ——→ 2001 (two colleges)

→ renaming a Column

→ ansi joins

→ multi table inserts.

→ flashback queys.

→ merge stmt.

(15) 9.2, 9.3.

(16) oracle 10g (grid technology) —— 2005

→ Introduces administration side

→ Recycle bin Concept

→ flashback table.

→ rename tablespace

→ wm_concat()

→ Indices of Oracle (plsql)

→ regular expressions.

Oracle 11g (2007).

→ Introduced Continue stmt. in PL/SQL loops

→ loops read only tables.

→ Simple_integer datatype introduced in PL/SQL.

→ Virtual Columns

→ follow clause introduced in triggers.

→ using sequences without using dual table in PL/SQL.

→ Compound triggers.

→ enable, disable clauses used in triggers specification.

→ named, mixed notations used in a subprogram executed using select stmt.

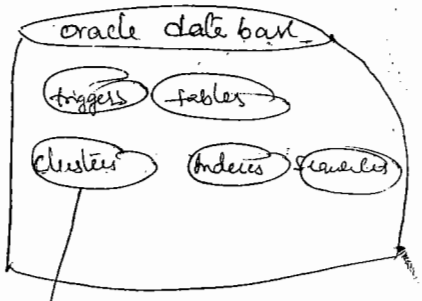
→ regexp_count().

Purpose → load data
modified data
deleted data
retrieve data

Sql is directly not interact with d.b

→ Any d.base

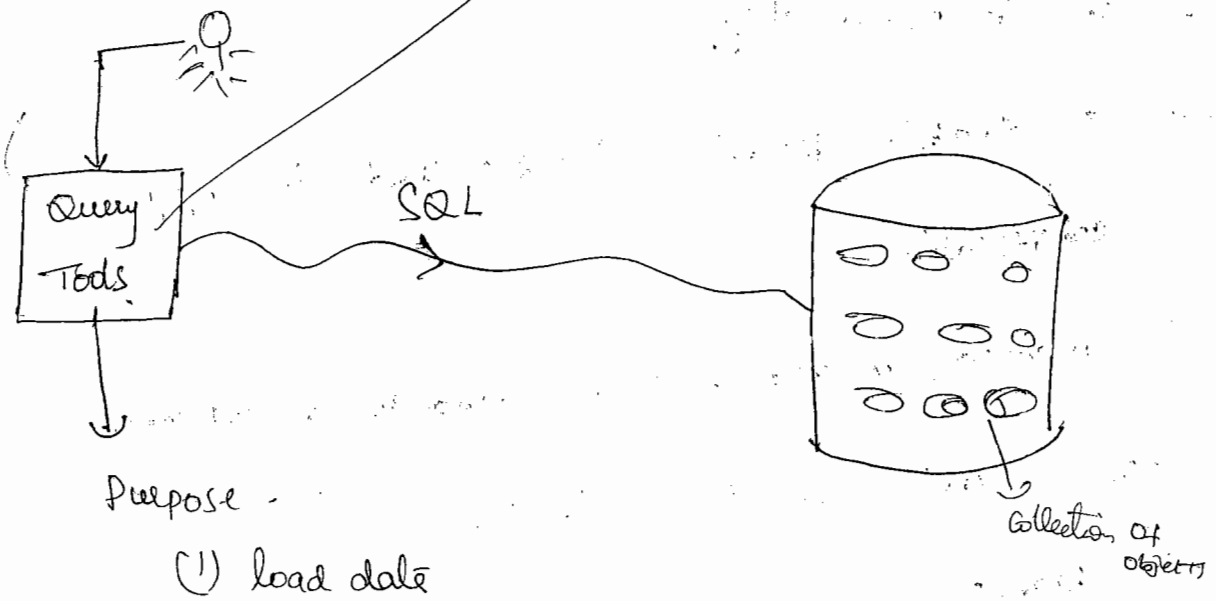
Through the "query tools" SQL server interact with d.b.



Collection of objects.

Tools

(1) Sql plus
(2) isqlplus
(3) Toad
(4) PL/SQL developer (use in projects)
(5) Sql navigator.



Purpose.

(1) load data
(2) modify data
(3) delete data
(4) Retrive data

→ don't load diff. s/w's

good deb. is
↓
Enterprise Edition s/w

→ 10g, 11g, 9i E.P. edition's

oracle.
— Express Edition.
— Standard "

oracle.
→ Each version having many Editions

→ start → programs → oracle application → SQL+ → any oracle sw we have in the "SQLplus Tool" by default

→ Express Edition we use user name: system, developer. P.W:- Edition.
but here no tables.

→ So we use Enterprise edition in workers in production Environment (Real time)

→ previous days. log, ilg → scott, Tiger → not works in Enterprise edition
but 1st time. log onwards security mechanism is there. (locking mechanism)

U.N:- | space sys AS SYSDBA → unlock user.

Oracle log. and

U.N:- scott

P.W:- Tiger ←

Error:- Account locked.

unlock user → U.N:- | sys AS SYSDBA ←

automatically asked the password.

Enter password: sys ←

SQL> Alter user scott account unlock; ←
keyword.

SQL> Conn scott / Tiger ←
Connecting (keyword) P.W.d.

Pwd:- Tiger ←

Confirm Pwd: Tiger ←

→ clear screen:-

Shift + delete (or) cl scr;

→ To view all tables.

→ select * from tab;

→ Predefined tables we most used 4 tables → (1) Dept (m.T)
(2) Emp (child T.) } most powerful tables, these 2 are normalized table.
(3) Bonus
(4) Salgrade.

R – tables
DB – tables.

Dept → master table. primary key → master key.

Emp → child table. foreign key → child table

→ To view particular table:-

→ select * from Dept;

→ Sql is divided into sublanguages:-

(1) DDL (Data Definition language):- (All DDL commands are automatically saved)

(1) Create
(2) Alter
(3) Drop
(4) Truncate } common.
(5) Rename (Oracle 9i)

↓
In universal standard DDL have Comment (save) the table.

② DML (Data manipulation language):-

(4) merge. oracle giving. (oracle 9i)

(1) insert
(2) update } Common commands
(3) delete

③ Data retrival language / Data query language :- (DQL)

(1) Select.

(2)

④ Transaction Control language:- (TCL):-

(1) Commit. (Save) permanently store in disk

(2) rollback. (undo from the memory)
Cancel (Ctrl+Z)

(3) Savepoint.

⑤ Data Control language:-

(1) Grant (giving permissions to another user)
(2) Revoke } Powerful

SQL

DDL, DML
TCL, DCL

PL/SQL

TCL, DML.
~~DCL~~

Data types

PL/SQL — Variable a number(10)

SQL — Column number(10)

C

int a;

SQL:-

→ D-types identifies type of data in a column

Data type:-

1. number(P, S)
2. Char
   - ↳ Varchar2(max size)
3. Date.

① Number(P,S):-

Precision ← P → Total no. of digits.

Scale ← S → floating point no's.

It is used to store fixed, floating pt nos.

Syntax:- Column name number(P,S).

T1

SNO
567.67

sno. number(5,2)

Sno. number(5)

max. limit of 'P' is '38',

↓

precision '1 to 38'.

(2) Char :- Alpha numeric data.

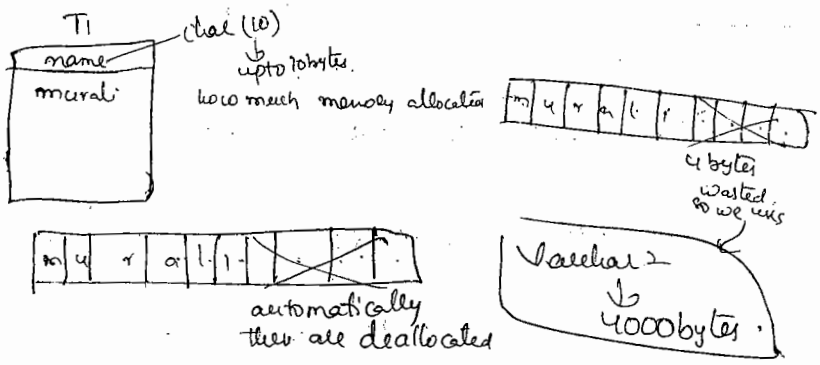
It is used to store fixed length, alpha numeric data in bytes.

→ max. limit upto '2000' bytes.

older 255 bytes
10g, 11g, character 2000 bytes upto → max. limit

→ By default character dty. leaving '1 byte'.

Syntax:- Columnname Char(size).



> Create table t1(name char(10));
1 table create. (Col) name

> Insert into t1 values ('murali');
1 row creation

Sureelli.

> select ~~*~~ from t1;

Name
murali.

→ memory wastage is display by "dump fn"

> Select dump(name) from t1;

dump(Name)

Type=96 len=10; 77 97 114 97 108 105 32 32 32 32

wastage
32-Ascii of 'space'

Varchar2
↓ dump(Name)

Type=1 len=2; 20 30 40 50 60

→ Varchar2(maxsize):-

* It is used to store Variable length, Alpha-numeric data in bytes.

→ max. limit is upto 4000 bytes.

Syntax:- Colname Varchar2(maxsize)

→ DATE:-

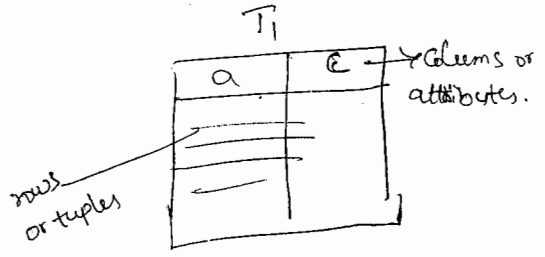
It is used to store dates in oracle date-format

→ DD-MON-YY.→ By default date-format in ord. 13-JUN-13.

> Select SYSDATE from Dual;

Syntax:- Columnname Date 14-6-13.

→ Data Definition language:-



These commands are used to defined structure of the table.

(1) Desc

→ Create:- not only for table any thing like triggers, database.

→ It is used to create database objects like tables, views, Sequences.

→ Creating a table:-

Create table tablename (colname1 datatype(size), colname2 datatype(size), ...);

> Create table d1(sno number(10), name varchar2(10));
table created

To View structure of the table:-

Syntax:- desc tablename;

Ex:-

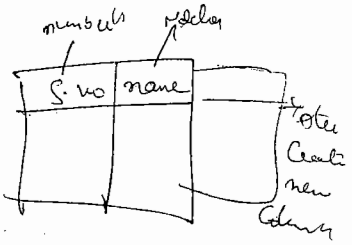
```
sql> desc d1;
SNO                    Type
name                   Number(10)
                       Varchar2(10)
```

→ Alter:- Add or remove Column.

It is used to change existing table structure.

Alter (3 commands in d.b) [how]

add modify drop.



→

Add:- It is used to add columns into the existing table.

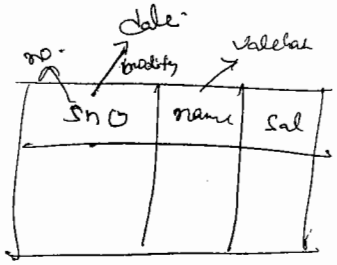
Syntax:- alter table tablename(add ↑ new column already not list column altered Col1 d.ty(s,2), ...);

Ex:- Add sal column.

→ alter table d1 add sal number(10);

→ 1 column is altered

Modify:- It is used to modify column datatype or column datatype size only.



Syntax:- alter table tablename modify(Col1 d.ty(s,3));

Empty table

S.no. is changed to date.

Ex:- alter table d1 modify SnO Varchar2(10);

desc d1;

SnO	Varchar2(10)
name	Varchar2(0)

drop:- Removing the Column.

2 methods.

① using ( ) →

② using Single Column → use Column keyword

→ It is used to remove Columns (Column sal) from the table.

method 1:- If u want to drop a Single Column at a time without using paranthesis then we must use Column key infront of the Column name.

Syntax:- alter table tablename drop Column (K.W.) Columnname;

Ex:- alter table d1 drop Column sal ; (no need to write d type)

method 2:- If you want to drop a Single or multiple Columns with using paranthesis then we are using following Syntax.

Syntax:- alter table tablename drop(col1, col2 ...);

Ex:- alter table d1 drop(sal, name);

Note:- We can not drop all columns in a table.

→ If we have only '1' col. It is not possible.

→ If we have 100 we can drop only '99' but '1' is not dropped error will be occured.

drop:- drop a tables, triggers, sequence, cluster, objects etc. (objects)

→ It is used to remove data base objects from the data base.

Syntax (objects)

*

Eg:- drop object-type objname;

- drop table tablename
- drop view viewname
- drop procedure procedurename

Eg:- drop table d1;

upto 10g → tables palmanantly dropped

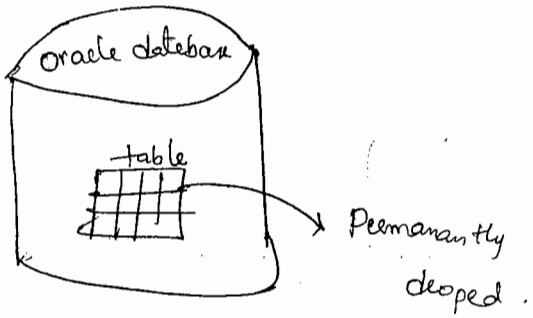
but 10g, 11g → not palmanantly dropped bcz here we have the Recycle bin

→ we remove palmantly we use 'purge' in 10g, 11g Enterprise Edition

droping a table.

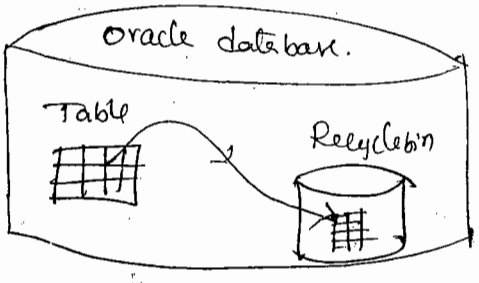
before oracle 10g.

Syntax: drop table tablename;



Oracle 10g (Enterprise edition)

Syntax:- drop table tablename;



get it back from Recyclebin:-

Syntax:-

Flashback table tablename to before drop;

To drop Permanantly:-

Syn:- drop table tablename purge;

ex: drop table d1 purge;

→ drop table d1;

Table dropped.

→ get it back:-

> flashback table d1 to before drop;

> desc d1;

To drop Parminantly:-

> drop table d1 purge;

> again apply flashback command.

> Error:- object not in Recycle bin.

→ Truncate:- (7.0 onwards) (late Command)

don't apply
↳ original tables in projects.

→ d.bak server always delete data parmanantly.

[→ delete Command → is not parently.
→ rollback → it comes back data]

→ It is used to delete data parmanently from table. [do not delete columns]

Syntax:

truncate table tablename;

Copy Table:- data one table to d1 table:-

> Create table d1 as select * from emp;

> Select * from d1; (all columns.)

> truncate table d1;

Table truncated.

Testing:

> Select * from d1;

no rows selected.

> desc d1;

Only column are displayed

Rename:- It is used to rename a table, & renaming a Column also (now a days)

Syntax:- (Rename Primarily learned)

rename oldtablename to newtablename;

Ex:- rename d1 to z1;

→ at a time only 1 table not a no. of tables.

Renaming a Column (oracle 9i):-

Column related to → Alter.

Only 1 Column at a time rename is possible.

Syntax:-

alter table tablename rename Column
i.w.

oldcolname to newcolname;.

→ 2 Colname is same is not possible in oracle.

→ alter table emp rename Column empno to sno;

Note:- By default all DDL Commands are automatically Committed (Saved).

DML (Data manipulation language) :-

This Comments are used to manipulate data in a table there are ① Insert
② update
③ Delete
④ merge (Oracle 9i)

① Insert :-

It is used to insert data into table

3 methods are there.

method 1 :-

Syntax :- insert into tablename values (value1, value2 ---);

Eg :- create table test (sno number (10), name varchar2(10));

> insert into test values (1, 'murali');

1 row created.

> insert into test (2, 'Sachin');

1 row created

> select * from test;

Sno	Name
1	Murali
2	Sachin

method 2:-

Using Substitutional operator (&);

Syntax:- insert into tablename values (&col1, &col2,---)

Eg:- > insert into test values (&sno, '&name');

Enter value for sno: 4

Enter value for name: xyz

> Select * from test;

method 3:- Skipping Columns.

Syntax:- insert into tablename (col1, col2,---) values (value1, value2---);

Eg:- insert into test (name) values ('bbb');

> Select * from test;

sno	name
1	murali
2	sailesh

Update:- It is used to modify data within a table

Syntax:- update tablename.

Set columnname = newvalue where column = old value;

Eg:- update emp set sal = 1000 where ename = 'smith';

> Select * from emp;

If you want to insert a value into particular cell then we also using update statement.

Eg:- > alter table test add address varchar2(10);

> update test set address = 'mumbai' where name = 'Sachin';

> Select * from test;

Sno	name	address
1	murali	
2	Sachin	mumbai
3	abc	

> update test set address = null where address = 'mumbai';

1 row updated.

> Select * from test;

Delete:- It is used to delete all rows all particulars rows from a table.

Syntax:- delete from tablename; ← (to delete all rows)

Syntax:- Delete from tablename where Condition;
↳ (to delete particular rows)

Eg:- > Delete from test (temporarily deleted)
4 rows deleted

> rollback; → (get it back)

Rollback Completed

> [truncate - permanently delete]

> Select * from test

> commit;
  Commit;

> roll back; rollback completed.

> Select * from test;
  no rows selected.

Difference between delete and truncate.

→ whenever we are using delete from table name
like total data deleted in a table and also this
data temporarily stored in a buffer. we can get it
back this data using rollback (without using commit)

→ whenever we are using truncate table table name
total data permanently deleted because by default
are ddl commands are automatically commited
that's why we can get it back deleted data.

Data Query Language (or) data retrival langel

Select: This command is used to retrived data from table.

Syntax: Select Col1, Col2 ..... from tablename where condition group by Columnname having condition order by Columnname [asc/desc];

1. Select all * Columns and all rows
2. Select all Columns and particular rows. → where.
3. Select particular Columns (col1, col2) and all rows
4. Select particular Columns & particular rows.

Creating a new table from another table.

Ex-6

Syntax:-

```
Create table tablename as select statement;
```

Ex:- > Create table x1 as select * from emp;

> Select * from x1;

display as like emp table.

Columns — primary key, foreign key, constraints.

when we are copying table constraints are not copied. Except (NOT NULL)

→ only copy columns

→ data (where condition → set of rows will take only columns will selected)

Creating a new table from another table without copying data:-

Syntax: Create table newtablename as select * from tablename where falsecondition;

Ex:- Create table x3 as Select * from emp

where 1=2; → false → no rows selected → select * from x3;

→ desc x3; → only columns are copied.

like empno='0000000'; } false condition.

1=9;

1=1 } True → It displays data.

→ when ever condition is false data will not copy.

→ Operators used in Select Statement.

→ after select some columns or all (Select - col1, col2)

(1) Arithmetic operators (*, /, +, -) (instead of these we can use mod (m,n))

(2) Comparision op's (=, <, <=, >=, >=, <> or !=)

(3) logical op's (AND, OR, NOT).

(4) Special op's.

} where

used in where clause

(1) Arthmetic op's:- are used in number, date datatype columns.

Q:- display ename, sal, annsal from Emp table.

> Select ename, sal, sal*2 "annsal" from emp;

Column alias.

ename sal annsal

> select sysdate from dual;

SysDate
17-JUN-13.

> select sysdate+1 from dual;

Tomorrows date.

> select sysdate-1 from dual;

Yesterdays date.

→

Query:- display the employees except JOB as clerk from emp table.

> Select * from emp where job != '~~CLERK~~';

After where compulsary write Column

Q> display the employees who are getting more than 2000 sal from emp table.

> Select * from emp where ~~SAL~~ > 2000;

using ~~AND~~ OR.

> select * from emp where JOB='CLERK' ~~AND~~
SAL > 2500;

> Select * from emp where JOB = 'CLERK' OR SAL > 2500.

OR → clerk and all sal above 2500 → display;

OR is nothing but '|' (comma)

→ display the employee who are belongs to deptno 10, 30, 50, 70, 90. from emp

> select * from emp where deptno = 10 OR deptno = 30 OR deptno = 50 OR dept no = 70 OR deptno = 90;

↓

Instead of we use another.

→ Special opr:-

		opp
1	in	not in
2	between	not between
3	is Null	is not null
4	like	not like.

used in where clause

In:- This op'r is used to Pick the values one by one from list of values.

→ generally we are using 'In' op'r inplace of 'OR' operator.

→ Bcz 'In' op'r performance is very high compare to 'OR' op'r

→ In multiple row subqueries we must use In operator.

Syntax [select * from tablename where

Columnname In(list of values);

↓
Seperated with ,.

> Select * from emp where deptno In(10,20,30,40,50);

> Select * from emp where ename In('King', 'Smith');

> NOT In —does not works with→ NULL

↓
No rows selected

> Select * from emp where Deptno
Not in (20,30);

— It displays dept no '10'.

> Select * from emp where Deptno
NOT in (20,30, NULL);

— no rows selected.

Note:- NOT in opr does not works with NULL values.

Null is diff
'0' is diff.

NULL:- NULL is an undefined, unavailable, unknown value

— It is not same as 'zero'

Null ≠ '0'

Any arithmeti opr with NULL = NULL

→ Any arithmetic operations performed on NULL again it will becomes NULL

NULL + 50 = NULL.

Q) display ename, Sal, Comm, Sal + Comm, Of the employee Smith. from emp table.

> select ename, sal, Comm, Sal + Comm
2400 + NULL = NULL.
from emp where ename = 'SMITH';

O.P:-

ename	Sal	Comm	Sal + Comm
Smith	2400	(NULL)	(NULL)

→ Sal + Comm

2400 + NULL = NULL

— If the above ex. any prblm is there to ours Come
prblm
NULL).

→ TO overcome this prblm oracle provided

NVL() function

Ex: NULL + 50

→ NULL

Inplace of NULL we replace '0'

NVL() → 0 + 50
→ 50.

↳ fun automatically substitute '2ero' user defined value.

NVL():- NVL() is an predefined fun. which is used to replace or substitute user defined value in place of 'NULL'

Syntax:- NVL (expr1, expr2).

must be same data type only.

→ These 2 expressions must belongs to same data type.

Ex:- NVL (Null, 0) = '0' (2) NVL(Null, 20) → 20

1st Ex - Null, 2nd Exp value = 2nd Expression.

— If Expr1 is NULL it return expression 2.

— Otherwise it returns expression '1'.

Ex:- NVL (30, 20) = 30.
1

Y select NVL(Null, 20) from Dual;

NVL(Null)
20

Y select ename, sal, comm, sal + NVL(comm, 0)

from emp where ename = 'SMITH';

ename	sal	comm	sal+comm
Smith	2400	(null)	2400.

→ sal + NVL(comm, 0)

2400 + NVL(NULL, 0)

2400 + 0

= 2400.

18-6

BETWEEN:- This opr is used to retrieve range of values. [always 1st write low value, not work 1st high value]

Syntax:- Select * from tablename where colname between lowvalue AND highvalue;

→ use AND opr only

Ex: Select * from emp where sal

between 2000 AND 5000;

↓

If given 5000 AND 2000;

↓

It is no rows selected.

→ BETWEEN — 2000 AND 5000.

↓

Including 2000, 5000 also.

IS NULL & IS NOT NULL :- → Indexes not work.

These op'rs are used in where clause only.

→ These op'rs are used to Test whether a Column having null values or not.

DeptNo
10
20
50
70
90

— Display employees 10th dept.

> Select * from emp where Deptno=10

— who are working NOL dept.

> Select * from emp where deptno=Null. IS NULL;

Syntax:-

① Select * from tablename where Columnname is null;

② Select * from tablename where Columnname is not null;

Employees who are not getting commission

> Select * from emp where Comm IS NULL;

[Comm=NULL no rows selected]

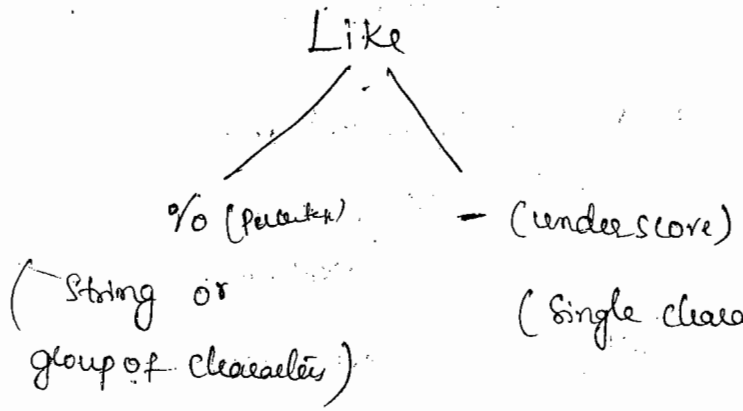
— Display the employees who are getting Comm. from emp table. using special operator.

> Select * from emp where Comm IS NOT NULL;

[Comm values displayed]

Like: This op'r is used to retrive data very fastly based on character pattern.

→ along with like op'r we are using 2 special characters.



Syntax: Select * from table name

where columnname Like 'character pattern'.

dates, no's, times, names all are in single quotes ' '.

Ex:

ends with 'N'

ALLEN — %. MARTIN — %

Display the employees whose Ename start with 'M' from emp table using like op'r.

```
Select * from emp where
        ename like 'M%';
```

— Martin
— Miller

> Display the employees whose ename second letter would be 'L'. from emp use like opr.

> Select * from emp where

ename like '_L%';

4th letter 'L'

'___L%';

- ALLEN
- BLAKE.
- CLARK.

— Concatination opr:- (||) :- (2 pipes).

It is used to Concatinate diff. data types Columns and also used to Concatinate Column values with Strings.

>Select 'my employee name is' || ename from emp.

my employee name is smith
" " " ALLEN

Functions.

→ Functions are used to solve particular task and also fun's return a value.

→ All d.base systems having 2-types of fun's

(1) predefined fun's (in SQL)

(2) user " " . (not possible in SQL).

(1) predefined functions :-

→ (1) Number fun's.

→ (2) Character fun's

→ (3) Date fun's.

→ (4) group fun's / aggregate fun's.

(1) Number functions :- These fun's operate over number data.

(1) abs() :- Convert -ve values to +ve.

Convert -ve values into +ve side we using "absolute fun"

Ex:- Select abs(-50) from dual;

O/P:-

abs(-50)
50

→ Test predefined fun we use 'dual' table;

Virtual table ← → Predefined 1 row
→ generate sequel values.

> Select abs(-50) from emp;

50
50
:
50 — 14 rows selected.

Select abs(-50) from dept;

50
50
50
50 — 4 rows selected

→ Dual:- Dual is an predefined virtual table which contains only 1 row and 1 column.

→ By default Dual table column datatype is varchar2.

→ generally this table is used to test predefined, user defined fun's functionalities. and also

→ This table is used to generates sequence values through sequence object. & also

→ This table is used to perform mathematical operations (in process like calculator)

> Select 40+80 from dual;

40+80
120

> Select 40+80 from dual;

40+80
120

> select * from dual;

Column.
(Dummy)
—
X
row

> desc dual;

Dummy Varchar2(1)

→ Select predefined or create database's
1st check on dual bcz it is proper works or not.

> select nvl(null,50) from dual;

op :- 50

> select nvl(30,50) from dual;

op :- 30.

> Select ename, comm, sal, comm-sal emp
where comm is not null;

abs(comm-sal)

op :- ename comm sal comm-sal
' ' ' —(500)
' ' ' —(500)

abs(comm-sal)
500
500

** mod(m,n):- It returns a reminder after m divided by n. $\frac{m}{n}$.

> Select mod(10,2) from dual;

o/p:-

mod(10,2)
0

> greatest (exp1, exp2 .. expn) ; least (exp1, expr2, .. expn)

→ greatest returns "maximum value" within given expression

→ where as least returns min value among given expressions.

> Select greatest (3,6,9) from dual;

o/p:- 9

— Display gretest sal from emp:

> Select greatest (Sal) from emp;

o/p:
1600
1200 } same values but not greatest value.

- single colmn dont apply

greatest → exp1, expr ..

max → single clm.

ename	com	sal
smith	300	1500

→ row greatest (sal,com)

→ If u want to compare no. of expressions we can greatest.

→ Select ename, comm, sal,
greatest (comm, sal) from emp where
comm is not null;

19-6:

whenever necessary flting pt nos then round.

round (m,n) → It rounds given floated value number 'm' based on 'n'.

↳ Select round (1.8) from dual; (1.4) (1.5)
op '2'. 1. 2.

→ Select round (1.23456, 3) from dual;
m, n
↳ after decimal 3 places.

op: 1.23456
1.234 · above 50%.
add '1'
op: 1.235

1.23410,3
↓
1.234 only

note:- round always checks remaining no. If remaining no. above 50% then '1' added to the rounded number.

→ Select ename, sal from emp;

> Select ename, sal, round (sal/22), round (sal/22, 1)
from emp;

O/P sal | 132.5
192.7

Trunc (m, n) :-

It Truncates (Remove) given floated number 'm', based on 'n'.

> Select trunc (1.7) from dual;
(Right placed Truncates)

O/P :- '1'.

> Select trunc (1.23456, 3) from dual;
(Truncated)

O/P :- 1.234.

→ ceil(), floor() :-

ceil() → returns a nearest greatest integer value

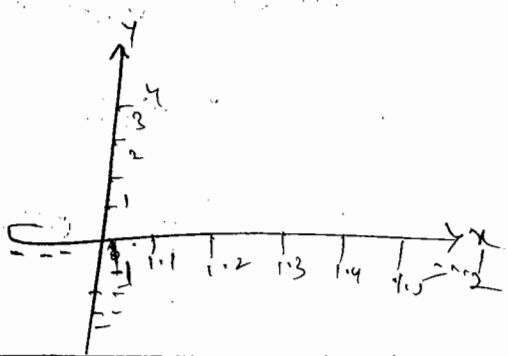
floor() → nearest, lowest integer value

→ Select ceil(1.3) from dual;
O/P :- 2 (nearest greatest integer)

→ Select floor(1.9) from dual;
'1' (nearest lowest integer)

→ ceil() returns nearest, greatest integer.

→ Where as floor() returns nearest lowest integer



Character functions

These fun's Operate over character data.

① upper():-

upper ('abc')

upper ("ename")
↳ colname do not use ' '.

It is used to retrieve column values or string into upper case.

> Select * from emp;

all enames are upper case

> select upper (ename) from emp;
↑ col values.

> Select upper ('abc') from dual;

② lower():-

> Select lower (ename) from emp;

o/p displays lower case.

> Select * from emp;

: displays original case (caps)

> update emp set ename = lower(ename);

14 rows updated

> Select * from emp;

> displays all enames in Small letters

Initcap():- It returns 1st letter would be Capital & all remaining should be small

> Select initcap(ename) from emp;

o/p Allen
James.

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

In every d.b. when ever space is there next letter will be new letter.

o/p:- Ab Cd Ef.

length():- always returns number data type

— It returns number of characters and also includes spaces also ex:- 'sh-ra-th' → 9 length

> Select length('AB CD') from dual;

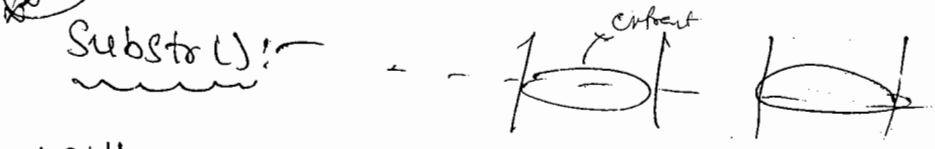
o/p: 5

select ename from emp where length(ename) = 5;

BLAKE
Miller

*** used in every d.b.

Substr():-



It will extract position of the string within given string based on last '2' parameters

Portion

> Substr('ABCDEFGH', 2, 3) → counting.

-8 -7 -6 -5 -4 -3 -2 -1
1 2 3 4 5 6 7 8

BCD

-2 3

~~CDE~~ . ~~GH~~

> Substr('ABCDEFGH', -5, 4)

DEFG.

> Substr('ABCDEFGH', -5)

DEFGH.

> Select Substr('ABCDEFGH', 2, 3) from dual;

o/p: BCD.

(-6, 4)

CDEF.

Syntax:-

must be nos

Substr(Colname (or) 'String name', Stirting Position, no. of characters from position)

> Select * from emp;

→ using substring display the employees whose ename second letter would be 'LA' (substring) from emp table.

[BLARK
CLARK]

(LA)

Powerful

~~> substr (ename, 2, 2)~~ → wrong.

~~Select * from emp where~~

> Select * from emp where

substr(ename, 2, 2) = 'LA';

O.P:- BLARK, CLARK

→ After select in no fun & character fun's where clause is possible.

after select fun
select max

→ select * from emp where max(sal)

after

→ group fun is not accepted

Note:- We are not allowed to use group fun's in where clause. But we are allow to use no. fun's, character fun's, date fun's in where clause.

→ INSTR():- Most powerful. But dead easy

Always this fun returns no. data type

- It returns position of the DELIMITER (special symbols ex: *, #) position of the character, position of the string within given String.

> Select instr ('ABCD', '&') From dual;

O/P:- 0
0
0

'*' → delimiter.

instr ('AB*CD', '*')

O/P:- 3.

1	'A'
2	'B'
2	'BC'
0	'BD'

→ sequence is not true.

instr ('AB*CDEFG*KJH*Nm', '*'); redocurrence no of occurences

('', '*', 2) ('', '*', 3)

O/P:- 3. O/P:- 9. O/P:- 13

instr() is always contains '4' parameters.

→ goes to 2nd star.

('', '*', 5, 2);

O/P:- 13

('', '*', 5)

O/P:- 0.

AB* → counts ('', '*', -5, 2).

o/p:- 3.

AB* → counts ('', '*', -4, 2)

o/p:- 3.

('', '*', -3, 2)

o/p:- ' '.

-3 -* is also counts.

Syntax:-

Instr (Columnname (or) ' String name ' , ' str ' ,

Searching , no. of occurances
Position from Position)

(must be no's)

Note:- Always Instr() returns position of the String based on last '2' parameters But oracle Server Counts no. of characters left side 1st position onwards (AB*)

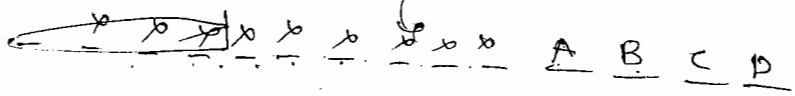
> Selet Substr (Title, 1, Instr(Title, ' ') from movie;

→ lpad():- (used in hierarchical query) 20-6

↳ adding some thing.

LPad ('ABCD', 10, '*')

↳ total length



→ It will fills remaining spaces with specified characters on left side.

→ Always here second parameter refers total length of the string.

> Select lpad ('ABCD', 10, '#') from dual;

o/p:- ######ABCD.

(ENAME, 10, '*') from emp;

o/p:- *****Smith.

→ RPad():-

> Select RPAD ('ABCD', 10, '*') from dual;

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

> Select Rpad (Ename, 40, '-') from Emp;

o/p:- Smith----------

> Select Rpad (Ename, 40, '-') || Sal from emp.

o/p:- Smith-----2000
Allen-----4000.

→ LTRIM():-
↳ removed.

It removes left side specified characters.

> Select LTRim ('SSMISSTHSS', 'S') from dual;

o/p:- MISSTHSS.

> Select JOB, LTrim (JOB, 'C') from emp;

o/p:- CLERK → LERK.

> Select LTrim (JOB, 'CS') from emp;

o/p:- Salesman - alesman.
Clerk - lerk.

→ (JOB, 'CSM') from Emp;

o/p:- Clerk - lerk.
Salesman - alesman.
Manage - anage.

> (JOB, 'CL') from emp.

o/p:- Clerk - erk.

→ RTRIM:-

> Select RTRIM ('SSMISSTHSS') from dual;

O.P:- SSMISSTH

→ TRIM( ):- Oracle 8i introduced trim function.

> select Trim ('SSMISSTHSS', 'S') from dual;

MITH (wrong). wrong syntax

O.P (MISSTH).

→ Trim remove only left & right side characters only.

→ middle characters are not removed

→ And also used to removed 1st, & last places (leading)

Syntax:-

Trim (' character (01) string , From Clause ' given string ' (02) Column name)

> Select Trim ('S' From 'SSMISSTHSS') from dual

O/P: MISSTH

> select Trim (' SmithH ') from dual;

o/p: SmithH for clarity o/p bcz there is spacing

↓

> select length (Trim (' SmithH ')) from dual

o/p: 5 (SmithH).

→ Translate() , Replace():-

> select translate ('JOHN', 'H', 'N') from dual;

(H replace with N)

o/p:- 'JONN'.

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

o/p: JONN'

> select Replace ('JOHN', 'H', 'NHFDHF') from dual;

(H replace with NHFDHF string.)

o/p:- JONHFDHFN.

> select Translate ('JOHN', 'H', 'NHFDHF') from dual;

o/p:- JONN'

> select Translate ('JOHN', 'HNOJ', '*MRQ') from dual;

o/p: QP*M.

→ Translate is used to replaces characters by characters where as

→ Replace is used to replaces characters by string or string by string

> Select translate('ABCD', 'DCBA', '1234') from dual

O.P:- 4321.

> Select Replace('A B C', ' ', 'ORACLE');
(space) (space)

O.P:- AORACLEBORACLEC

→ String with string also Replace.

> Select JOB, Replace(JOB, 'salesman', 'marketeis') from emp;

O.P:- Salesman Connecting into marketeis.

Select Replace('SSMISSTHIS', 'S') from dual;

O.P:- MITH.

deleted 'S' from total string not only left, right, middle also.

DATE Functions:-

[Sysdate-1 → Yesterday
Sysdate+1 → Tommorow]

→ In oracle by default date format is

dd-MON-YY

(1) SYSDATE (pseudo column) → no()

(2) ADD_MONTHS()

(3) LAST_DAY()

(4) NEXT_DAY()

(5) MONTHS_BETWEEN().

① SYSDATE:- It returns Current System Date.

> Select Sysdate From Dual ;

O.P:- 20-Jun-13.

② ADD_MONTHS():- It is used to add or subtract no. of months to the specified date based on 2nd parameter.

> Select add_months (Sysdate, 1) from dual ; ← adding 1 month.

O.P:- 20-JUL-13.

> (sysdate, 15) > (Sysdate, -1) ← previous month.

O.P:- 20-SEP-14. O.P:- 20-MAY-13.

> Select add_months ('15-aug-05', 1) from dual
↳ our date.

15-Sep-05

③ LAST_DAY():- It returns last date of the specified month.

> Select last_day(Sysdate) from dual ← passing only 1 parameter.

O/P: 30-JUN-13.

④ NEXT_DAY():- next Sunday date, next monday date.

→ It returns next occurance day from the specified date based on 2nd parameter.

> Select Next_day(sysdate, 'Sunday') from dual
next sunday

23-JUN-13.

⑤ MONTHS_BETWEEN():- [returns number datatype no only.]

[months_between(Date1, Date2)].

Always this fun. returns no. datatype. i.e, It returns no of months b/w 2 dates. Always Date1 more than Date2. [Date1 > Date2] other-wise It returns -ve sign.

> Select months_between ('16-SEP-04', '16-JUL-04') from dual

> Select months_between ('16-FEB-04', '16-JUL-04');

o/p -5

> Select months_between (abs('16-FEB-04', '16-JUL-04');

o/p 5

→ Always returns floating pt values bcz date is '31'. So we use round().

— how many employees works.

> select months_between(sysdate, HIREDATE) from emp;

→ o/p:- 390.11470
bcz 389.22227. } floating pts.
1st, lowest date, 2nd highest date.

> select round(months_between(sysdate, hiredate)/12) from emp;

o/p: 390/12 → 32.5 years.
38

→ This fun always returns floating pt values bcz internally this fun uses every month have '31' days.

> select ename, round(months_between(sysdate, hiredate)/12) from emp;

o/p Smith - 22
allen - 33.

Date Arithmetic

(1) Date + Number ✓ (Possible)

(2) Date − Number ✓

(3) Date1 + Date2 ✗

(4) Date1 − Date2 ✓ = d

> (1) select sysdate+1 from dual;

Tomorrow.

(2) select sysdate-1 from dual;

Yesterday.

(3) select sysdate- sysdate from dual;

Zero'.

→ 1st day of month

> Way to display 1st date of the current month using predefined fun's.

> Previous month last day +1.

> select ADD_MONTHS (LAST_DAY (sysdate), -1)+1

from dual;

01-JUN-09.

Date Conversion — TO_date.

TO_date.

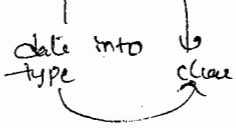
Simple, but more powerful:-

Date Conversion functions :-

① to_char() :-

② to_date() :-

① to_char() :- It is used to convert date type into character type. (date string).



i.e; it is used to convert date type into date string.

> Select TO_char(Sysdate, 'Day month yyyy') from dual

O/P :- Friday JUNE 2013

→ convert any date into client requirement format

> Select TO_char(Sysdate, 'DD/mm/yyyy') from dual.

O/P :- 21/06/2013.

Day
Dy
Month
Mon
year
} return value character

return nos. { D, DD, DDD, MM, YY

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

o/p:- ~~f~~riday FRIDAY 'day' o/p:- friday.

'DY'	'DD'	'D'	'DDD'
o/p:- ~~FR~~I SAT	21	6	172.

Note:- Basically TO_CHAR() is case sensitive fun.

DAY - FRIDAY DY - FRI.

day - friday.

> select to_char (sysdate, 'D') from dual;

o/p:- '6' (21 → date)

Sunday - 1 6 - friday.

mon - 2 ---- Fridy - 6'

> DDD → day of the years (completed days)

> DDth → 20th, 23rd.

> DDSP → Date Spell out. o/p:- Twenty-First

> MON → JUN. > month - June.

> MM → 06.

> YEAR → Twenty thirteen. > YYYY - 2013.

> YY - 13

> HH24:MI:SS → 12:04:46.

hours & mins & secs

FM - Fill mode

> If any space is occur in To_char O.P. we use 'FM'.

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

O.P: 21/JUNE 2013.

> Select to_char (sysdate, 'DD' /Fmmonth/ YYYY') from dual.

O.P: 21/JUNE/2013

31

→ 15-Aug-05 convert into 15/August/05.

> Select to_char('15-Aug-05', 'DD/month/YY') from dual;

string.

Error O.P:- Invalid number.

> In 'TO_char()' we must & should follow rules:-

→ 1st parameter → date-type.

→ when ever we use quotes date is

It is a taken as string

→ string is convert into date using TO_date().

> Select to_char(to_date('15-Aug-05', 'DD/month/YY') from dual

no char no. DAY

O.P: 15/August/05. Monday/August/05.

→ If 1st parameter 'sysdate' no need to_date.

Error Note:- when ever we are using TO_char() always 1st parameter must be date type. other wise oracle server returns an error. ora-1722: invalid number. [in DUAL side]
to over come this problem we are converting date string into date type using TO_date() conversion function..

TO_Date():- It is used to convert date string into date type. (oracle date format) (any string is date)
(date string → into → date type)

> select to_date('15/JUNE/2007') from dual;

O.P:- 15-Jun-07 (oracle format).
dd-mon-yy.

> select to_date('15/06/07') from dual;

O.P:- Error. Not a valid month

→ Follow rules for TO_Date():- (most powerful concept)

TO_Date('15/JUNE/07')
(no. char no.)

when we pass to oracle returns types must match with default date format

DD-mon-YY
(no- char no.)
(don't match 15/06/07 → no no no)
→ invalid month

no char no
DD- mon - YY
15 / 06 / 07
no no invalid
error :- month.

→ when ever get an error in TO-Date in SQL we use a 2nd Palamelēe Same as 1st Palamelēe format

> Select to_date ('15/06/07', 'DD/mm/YY') from dual

o.p :- 15-JUN-07

Note:- whenever we are using TO-date Passd palamete letter values match with the default date format other wise use a 2nd palamelēe as same as 1st Palamelēe format.

Then only oracle Server Internally Converts date string into date type.

↑ character or string

→ any date + no. is not working so it convert in to date type so use to_date

> Select to_Date ('09-Feb-04') + 5 from dual

o.p: 14-Feb-04

Select '09-Feb-04' + 5 from dual

↓

error so us to date.

> Select to_date ('09-02-04', 'dd-mm-yy');

o/p: 14-Feb-04

Q) Display the Employees who are joining in the month december from emp table using To_char()

> Select To_char(Hiredate, 'dd-Dec-yy') from emp;

>
Select To_char(

This is Conditionally query. So use where clause

> Select * from emp where To_char(Hiredate, mon)='Dec';

o/p = Dec

> display the employees who are joining in the year 81 from emp table using to_char()

> Select * from emp where To_char(Hiredate, 'yy')='81';

→ we want to know year is 1981 or 1981 or 1881, 1771.

> Select * from emp where to_char(hiredate, 'yyyy')

> Select hiredate, To_Char(hiredate, 'YYYY')

from emp;

o.p:- 17-Dec-80

To_char
YYYY
YY

27-06

13-6-2014.

Date Conversion Functions():-

(1) to_char() :- In where clause.

(2) to_date() :-

Select to_char(to_date('24-JUN-05'), 'DD/mm/YYYY'

O:P:- 24/06/2005. from dual

WA query to display the employees who are joining in the month december from emp table using to_char function.

~~Select ename to_char(HIREdate, 'dd-DEC-YY')~~
~~from emp;~~

Select * from emp where to_char(HIREdate, 'mon')='DEC';

~~O:P:-~~

WA query to display the employees who are joining in the year '81' from emp table.

Select * from emp where to_char(HIREdate, 'YY') = '81';

a characters in to-char so gaps is there in o.p.

Fill Mode:- (FM).

when ever to_char returns gaps then we must use Fill mode this mode is represented by 'FM'.

Ex:- Select to-char(sysdate, 'DAY/MON/YYYY') from dual

O.P:- FRIDAY---/JUN/2014.

Select to-char(sysdate, 'FMDAY/MON/YYYY')

O.P:- FRIDAY/JUN/2014.

Full length character to-char is a problem. (Internal concept)

> Select * from Emp where to-char(Hiredate, 'month' = December;

O.P:- NO rows Selected.

> ------ 'Fmmonth' = December;

O.P will occurs.

to-char helps in another situation does Year is 1980 or 2080 or not.

> Select Hiredate, to-char(Hiredate, 'YYYY') from Emp;

O.P:-

Hiredate	
1980	1980
1981	1981
1982	1982

Note:- In oracle whenever we are passing date string '05-Jun-06' into date fun's (last_day(), next_month()) then oracle server automatically converts date string into date type.

⇒ But here passed parameter format must be by default date format ('dd-mon-yy') otherwise oracle server returns an error.

⇒ In this case we must use to_date (If error will come)

> Select last_day ('15-JUN-05') from dual;

O/P:- 30-JUN-05.

> Select last_day ('15-06-06') from dual

O/P:- not a valid month.

Solution:-

> Select last_day (to_date ('15-06-05', 'DD-mm-YY'))

from dual;

O/P:- 30-JUN-05.

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

* Every d.b. date data type contains 2 parts

① date part (Visible) ② time part (not visible)

⇒ In oracle when ever we are using round(), trunc() fun's on dates then automatically date part is changed based on time portion.

⇒ whenever we are using a round() fun then oracle Server automatically add 1 day to the date if time portion is ≥ 12 Noon. And also time portion automatically set to 'Zero' ('00:00:00')

Testing at 7:00 PM :-

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

O/P: 13-JUN-14 19:10:45.

Using Round() :-

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

O/P: 14-JUN-14 00:00:00.

trunc():- whenever we are using trunc() oracle server returns same date and also time portion automatically set to 'zero'. (00:00:00). If time portion ≥ 12 Noon also.

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

O.P:- 13-JUN-14 00:00:00.

WA query to display the employees who are joining today from emp table

> Ext Insert into emp (empno, ename, hiredate) values (1, 'murali', sysdate);

> Select * from emp where hiredate = sysdate;

⇒ no rows selected.

Solution:-

> Select * from emp where trunc (hiredate) = trunc (sysdate)

1 murali 13-JUN-14.

14-6-14.

Group functions (or) aggregate fun's:-

Compulsory passes column name not our own

(1) max()
(2) min()
(3) avg()
(4) Sum()
(5) Count(*)
(6) Count (Column name)

All group fun's operate on group of data and returns a single value.

(1) max():- It returns maximum value within a Column.

> Select max(sal) from emp;

O.P:- 5500.

> Select max(HIREDATE) from emp; (freshee)

O.P:- 12-Jan-83.

> Select max(ename) from emp;

WARD (alphabetical order)

(2) min():-

> Select min(sal) from emp;

O.P:- 950.

> Select min(HIREDATE) from emp; (Senior)

17-Dec-80.

Note:- We are not allow to use group fun's in where clause.

(3) avg():-

> Select avg (sal) from emp;

O.P:- 2433.92857..

Note:-

> Select avg (Comm) from emp;

O.P:- 550.

Note:- In oracle by default all group fun's ignores null values exept "Count(*)".

> With null value we are using nvl().

> Select avg (nvl(Comm,0)) from emp;

O.P:- 157.142857

Sum():- It returns total in a no. datatype Column.

> Select Sum (sal) from emp;

34075

Count (*):- It returns counts no. of rows within a table.

> Select Count(*) from emp;

O.P:- 14 (rows).

Count(Column name) :-

It Counts no. of not null values within a Column

> Select Count(Comm) from emp;

O.P:- 4.

> Select Count (distinct(deptno)) from emp;

O.P:- 3 (10,20,30)

group by :- group by clause is used to arrange similar data items into set of logical groups this clause used in Select stmt

Syntax:- Select Columnname from tablename

group by Columnname;

> WAQ to display no. of employees in each deptment from emp table using group by.

> Select deptno, Count(*) from emp group by deptno;

deptno	Count(*)
30	6
20	5
10	3

- WAQ to display no. of employees jobwise from emp table using group by.

> Select job, count(*) from emp group by job;

O.P:-

clerk	4	president	1		
salesman	4	manager	3	Analyst	2

> Select deptno, min(sal), max(sal) from emp group by deptno

O.P:-

Deptno	min(sal)	max(sal)
30	950	3150
20	1500	3600
10	2900	5500.

Note:- In all d.b. systems we can also use group by clause without using group functions.

> Select deptno from emp group by deptno.

Deptno,
30
20
10

Rule:- other than group-fun columns specified after Select those all columns must be used after group by otherwise Oracle Server returns an error "not a group by expression",

Ex:- Select deptno, max(sal), ename from emp

group by deptno.

Error:- "not a group by expression"

> Select deptno, max(sal), ename from emp

group by deptno, ename,

↓

O/P:- will come. here we donot use group-fun

> Select deptno, job from emp

group by deptno, job.

O/P:- ✓

> Select deptno from emp → here less cols is working

group by deptno, job;

↳ bt here after select how many cols here also that no. of cols

Deptno
20
30
20
10
20
10

Note:- when we are try to display group fun Columns with another column then database Server Returns errors to overcome this problem we must use group by clause

Ex:- Step 1:-

> Select Sum(sal) from emp;

3407

> Step2:-

> Select deptno, sum(sal) from emp;

error:- not a single-group group function

Solution:- using group by

Select deptno, sum(sal) from emp

group by deptno;

o/p:-

deptno	sum(sal)
30	9100
20	13625
10	11350.

16-6-14

> WAQ to display those departments having more than 4 employees from emp table using group by clause.

> Select deptno, count(*) from emp

group by deptno

where count(*) > 4;

~~deptno = "20";~~

Error:- after group by we ~~can't~~ use where clause

Sol:-

Select deptno, count(*) from emp group by deptno having count(*) > 4;

o/p:-

Deptno	Count(*)
30	6
20	5.

Having:- After group by clause we are not allow to use "where" clause.

In place of this one ansi/iso SQL Provided another clause "having".

⇒ Generally if you want to restrict rows in a table then we are using "where clause". [where is in table]

Whereas if u want to restrict groups after group by then we are using "Having clause".

⇒ Generally we are not allow to use group fun's in "where clause".

Whereas we can also use group fun's in "having clause".

Order by:- not allow to use in a subquery

⇒ This clause is used to arranged data in sorting Order. along with order by clause we are using 2 key words. There are ASC, DESC. (1) Asc (2) Desc

→ by default Order by clause having 'Asc' order

Syntax: Select * from tablename

order by column name [asc/desc];

> Select Sal from emp Order by Sal desc

O/P: 5600
3500
⋮
950.

> Note: In order by clause we can also use Column Position (no.) in place of column name for improve Performance of the query.

Ex: > Select * from emp order by 2 desc
↓
must use * only when use the column position.

O/P:
eno ename
⋮
x x
⋮

> Select deptno, Count(*) from emp
where Sal > 1000
groups by deptno
group having Count(*) > 2
Order by deptno asc.

deptno	Count(*)
30	3
20	5
30	4

roll up, Cube: along with group by. (Oracle 8i)

⇒ Oracle 8i introduced roll up, Cube clauses. These clauses are optional clauses use along with group by clause only.

⇒ These clauses are used to calculate Sub total, grand total automatically

Syntax:-

Select Col1, Col2 ....

From table name group by rollup (Col1, Col2);

(2) Select Col1, Col2 ....

from table name group by Cube (Col1, Col2);

Rollup is used to calculate Sub-total, grand-total based on a single column.

Whereas if you want to calculate Subtotal, grand total based on no. of columns then we use using 'Cube'.

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

(deptno: only this column total)

> Select deptno, job, sum(sal), count(*) from emp group by Cube (deptno, job);

⇒ WAQ to sum of sal and with ename with grand total

Select ename, Sum(sal) from emp

group by ~~ename~~ rollup(ename);

Joins.

Joins are used to retrive data from multiple tables

Rule:- If u are joining 'n' table then we are using (n-1) join Conditions.

- Oracle Server having following types of joins

8i joins. in oracle d.b.
- (1) Equi join (or) Inner join.
- (2) Non Equi join
- (3) Self join
- (4) Outer join.

left outer, right outer (9i)

These joins are called "8i joins"

9i joins (or) ANSI joins.

1. inner join.
2. left outer join. (left join)
3. right outer join. (right join)
4. full outer join
5. Natural join.

These joins are working any d.b that 8i work only in oracle.

- Internally Oracle d.b. Server uses Cross join by default.

Note:- We can also retrive data from multiple tables without using joining Condition. In that case Oracle Server internally uses "Cross join" (by default)

Cross join returns more Duplicate data because Cross join works based on Cartesian product.

Ex:- > Select ename, sal, dname, loc
from emp, dept.

17-09-19

(1) Equi join (or) INNER JOIN :- common column & same data-type matching rows & data

Based on equality Condition we are retriving data from multiple tables. Here joining Conditional Columns must belongs to same data-type.

- When tables having Common Column names then only using this join.

Syntax:- Select col1, col2, ...
from tablename1, tablename2
where tablename1.Common Columnname = tablename2.Common Columnname;

joining condition.

> Select ename, sal, deptno, dname, loc from emp, dept

where emp.deptno = dept.deptno;

Error:- Column ambiguously defined.

→ always execution goes to first from only.

Note:- In databases if u want to avoid ambiguity then we must specify table name along with common column name using '.' operator.

Ex:-

Select ename, sal, emp.deptno, dept.deptno, dname, loc

from emp, dept

where emp.deptno = dept.deptno.

Note:- Generally to avoid ambiguity feature we can specify every column name along with table using '.' operator

Using alias names:- We can also use alias names for the table in from clause. These alias names also called as reference names created in from clause and also these reference names must be different names.

Syntax: from tablename1 alias name1,
tablename2 alias name2.

Ex: Select ename, sal, d.deptno, dname, loc
from emp e, dept d
where e.deptno = d.deptno;

Note: using equi join we are retriving "matching rows" only.

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

(Here deptno 40 does not display if we are using d.deptno also)

— WAQ to display the employees who are working in the location Chicago using Equi join from emp, dept table.

Select e.ename, d.loc
from emp e, dept d
where e.deptno = d.deptno AND loc = 'CHICAGO';

AND operator

Note:- If we want to use filter Condition (restriction) after joining Condition then we must use 'AND' operator in '8i' joins

Where as in '9i' joins we are using either 'and' (or) 'OR' 'where' clause.

WAQ to display dname, Sum(sal) from emp, dept table using equi join

Select d.dname, Sum(sal)

from ~~group by~~ ~~d.deptno~~ ddname, emp e

where d.deptno = e.deptno;

> Select d.dname, sum(sal), d.deptno

from emp e, dept d

where d.deptno = e.deptno

group by d.dname, d.deptno.

deptno	dname	Sum(sal)
10	Accounting	10247.5

Outer join

~~Using clause:-~~

WA query to display dname, deptno, sum(sal) from emp, dept table of those employees, whose Salary is greater than 15000

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

~~where = on~~

having sum(sal) > 15000

Deptno	Dname	Sum(sal)
10	Accounting	18247.5

WAQ to display deptname & sum of salary of those department in which no. of employees in dept > 3.

Select dname, sum(sal) from emp e, dept d where e.deptno = d.deptno group by dname having count(*) > 3

→ DNAME SUM(SAL)

RESEARCH

SALES.

> WAQ to display deptname & sum of salary of department where no. of employees > 3 & also calculate total of salary automatically

> Select dname, sum(sal) from emp e, dept d where e.deptno = d.deptno group by ~~dname~~ having count(*) > 3 rollup (dname);

Select dname, sum(sal) from emp e, dept d where e.deptno = d.deptno group by rollup (dname) having count(*) > 3

Non-equi join:-

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

→ Generally when tables does not have common columns then only we are using this join.

Eg:- > select ename, sal, losal, hisal from emp, salgrade where sal between losal and hisal;

(or)

> Select ename, sal, losal, hisal from emp, Salgrade where sal >= losal and sal <= hisal;

Self Join:-

- Joining a table to itself is called self join.
- Here joining conditional columns must belong to same data type.
- Generally if you want to compare 2 different column values in a single table then we must use self join. In this case these 2 columns must belong to same data type.

→ If you want to compare multiple values within a single column also then we are use self-join

—whenever we are using self join then we must create table aliases in 'from' clause

from tablename aliasname1, tablename aliasname2 - - - -

Table Alias Names are Reference Name:-

→ In all database, when we representing a same table, twice (or) if we are refering a table using diff name, then we must use table as aliasname. (using clause)- (ANSI) (SQL JOINS)

These are also called referenced for a table.

These aliasnames must be diff. name.

These aliasnames behaves as table where query executes.

Ex: in oracle. table. aliasname are used self-join Correlated Subqueries, merge stmt.

Q) WAQ to display the employees who are getting same salary in emp table. using self join (employee Smith salary)

→ e1

ename	sal
Smith	x

e2

ename	sal
?	x

> Select e2.ename, e1.sal from emp e1, emp e2 where e1.sal = e2.sal and e1.ename = 'SMITH';

ename	Sal
SMITH	1600
ALLEN	1600

WAQ to display employee names & their manager names from emp table using selfjoin

As all employees are not manager but all managers are employees. So server searches each manager number in a column empno.

emp e2

empno	emp	mgr
7369	SMITH	7902
7499	ALLEN	7698

> Select e1.ename "employees", "e2.ename" managers from emp e1. empe2 where e1.mgr = e2.empno;

(or)

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

(It display matches data Compaison.)

Outer join:-

This join is used to retrive all rows from 1 table & matching rows from another table.

Generally equi join returns matching rows only if we want to return non matching rows also then we all using join operator (+) in joining Condition of the equi join this is called as outer join

Note:- This join operator can be used only 1 side at a time in joining Condition.

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

matching rows → All rows

O.P :- 40 operations BOSTON

9i joins (or) ANSI Joins

- Inner join
- left outer join
- right outer join
- full outer join
- Natural join

Inner join:-

This join also returns matching rows only. This join performance is very high compared to 8i equi join

when tables having Common columns then only we are using this join. Here also joining Conditional Columns must belongs to Same data type

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

Q) WAQ to display the employees who are working in the location 'chicago' from emp, dept table using '9i' inner join

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

(or)

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

ENAME	LOC
WARD	CHICAGO
TURNER	CHICAGO
ALLEN	CHICAGO
JAMES	CHICAGO.

22-7-14

Using clause:-

⟶ In ansi joins, we can also use using clause in place of 'on' clause using clause performance is very high compared to on clause. Always using clauses returns common columns one time only

Syntax:-

```
select * from tablename1 join tablename2
using (commoncolname);
```

Ex:- > select * from t1;

A	B	C
x	y	z

> select * from t2;

A	B
x	y

> select * from t1 join t2 on t1.a = t2.a and t1.b = t2.b;

A	B	C	A	B
x	y	z	x	y

> Select * from t1 join t2 using (a,b);

A	B	C
x	y	z

Note:- whenever we are using "using" clause, then we are not allowed to use alias name on joining conditional column.

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

left outer join:- (left side table). 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.

Select * from t_1;

A	B	C
X	Y	Z
P	q	r

> Select * from t_2;

A	B
X	Y
S	t.

> Select * from t_1. left outer join t_2 on $t_1.a = t_2.a$ and $t_1.b = t_2.b$;

A	B	C	A	B
X	Y	Z	X	Y
P	q	r	null	null

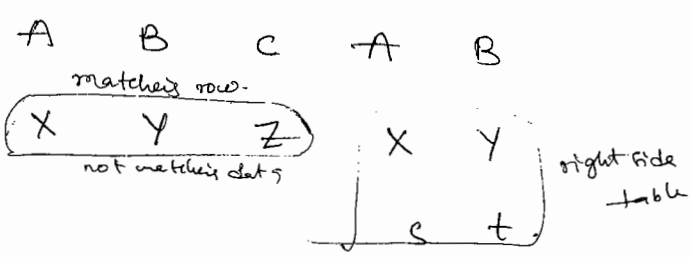
23-6-14.

Right outer join :- (right side table)

This join returns all rows from right side table and matching rows from left side table and also returns 'NULL' values inplace of non matching rows in another table.

Ex:-

Y Select * from t_1 right outer join t_2 on $t_1.a = t_2.a$ and $t_1.b = t_2.b$;



⇒ Full outer join :- It is the combination of left, right outer join i.e, it returns all rows from all the tables and also returns 'NULL' values, in place of non-matching rows, in another table.

A	B	C	A	B
X	Y	Z	X	Y
P	q	r	NULL	NULL
ou	-	-	s	t.

Y Select * from t_1 full outer join t_2 on $t_1.a = t_2.a$ and $t_1.b = t_2.b$;

on clause requires in 8i, 9i.

where not required

Natural join:- this join also returns matching rows only.

① → In this join we are not allow to use joining condition.

② ** when tables having common columns then only we are using this join.

③ This join performance is very high compare to inner join.

④ Natural join internally uses using clause thats why this join also returns common columns 1 time only.

Syntax: select * from

t1 natural join t2;

A	B	C
X	Y	Z

AB one time only display.

Note:- when we are using natural join also we are not allow to use alias name on joining conditional column. Bcz natural join internally uses using clause.

Ex:- > Select d.Dept ✗

ename, sal, deptno, dname, loc

from emp e natural join dept d;

Cross join:- (default join)

> Select ename, sal, dname, loc from emp, dept;

> Select ename, sal, dname, loc ~~(like a)~~ from emp cross join dept;

O.P:- 70 rows (56 rows) 14*4

Joining more than 2 tables.

ex:- Joining 3 tables. (2 join conditions.) 8i join.

Syntax:-

Select col_1, col_2 .....

from table name 1, table name 2, table 3

where

table name 1. Common col = table name 2. Comm column

AND

table 2. Comm col = table 3. Comm col;

9i join. (3 tables) (ANSI joins)

Syntax:

Select col1, col2 ....

from table1 join table2

on

table 1. Comm Col = table2. Comm col

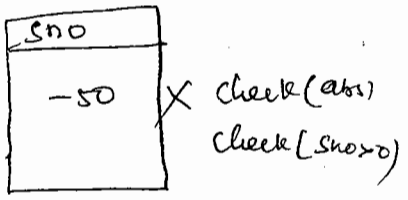
Join table3.

on table 2. Comm col = table 3. Comm col.

2. Constraints.

23-7-14.

t1



← Constraints are used to prevents invalid data entry into our tables.

- Constraints are created on table columns.
- Oracle server having following types of Constraints

(1) not null (3) primary key. (5) check.

(2) unique (4) foreign key.

> In all the above constraints are created in 2 ways. (2 methods)

(1) Column level.

(2) Table level.

(1) Column level:- In this method we are defining (creating) Constraints on individual columns.

In this method i.e, ↑ when ever we are defining the column then immediatly we are defining Constraint type specifying.

Syntax:-

```
Create table tablename (Col1 datatype(size)
constraint-type, Col2 datatype(size) Constraint-type,...);
```

** (2) Table level:-

Create Bank (Accno varchar2) primary key
Acname varchar2 primary key ; XX

In this method we are defining Constraints on group of Columns. i.e, In this method ↑ 1st we are defining all Columns & last only we can specify Constraint type along with group of Columns (primary key (Accno name))

Syntax:-

```
Create table tablename (Col1 dtype(size)
Col2 dtyp(size),.....,Constraint-type (Col1,Col2....));
```

Constraint types or Constraints

	Column	Table
① Not null.	✓	✗
② unique	✓	✓
③ P.K.	✓	✓
④ F.K.	✓	✓
⑤ Check	✓	✓

> Create table bank
(Acc number(10),
name varchar(20),
balance number(10),
Primary key (Ac(no,name)

notnull unique

a	b	c

→ only 1 P.K. in 1 table.

→ uniqueidentification of record in a table

→ If 1 P.K Column not possible programer assumption Combination of Columns will take as P.key

...P.K.

Acno	name	balance
1001	murali	50000
1002	abc	20000

(1) NOT NULL :- (no table level) Not null constraint does not support table level.

It does not accepts "null" values but it will accepts duplicate values.

Column level :-

Ex:- > create table d1 (sno number (10) not null, name varchar2 (10));

O/P:-

Name	Null ?	Type
sno	not-null	Number(10)
name		varchar2)

testing :-

> insert into d1 values (null, 'abc');

Error:- 01400 :- Can not insert null int sno.

> insert into d1 values (20, 'abc');
insert into d1 values (20, 'xyz'); } duplicate allows.

Table level :-

Create table d2 (sno number(10), name varchar2(10), not null (sno));

Error:- invalid identifier

bcz not null not supported Table level.

② **Unique**:- This Constraint is Created on Column-level, table level.

It does not accepts duplicate values but it will accepts null values (oppo. not null).

unique & P.K. Column using where clause bcz (Internally d-b server creates indexes) bcz improving performance.

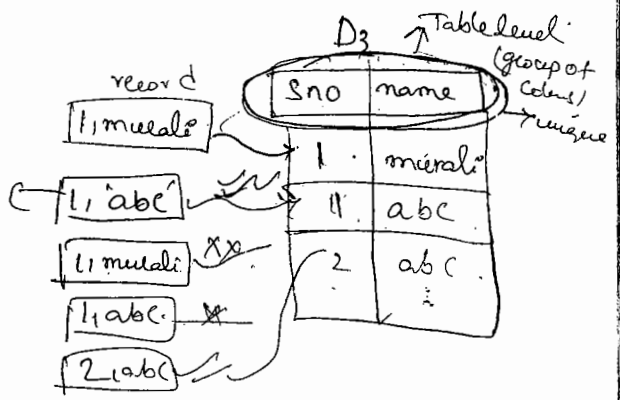
Note:- when ever we are creating unique Constraint oracle Server internally automatically creates 'btree' indexes on those Columns.

Column level:-

> Create table d2 (sno number(10) unique, name varchar2(10));

Table level:- record

> Create table d3(sno number(10), name varchar2(10), unique (sno, name));



> Select * from d3;

Sno	Name
1	murali
1	abc

unique

> Insert into d3 values (1, 'abc');

error:- ORA-00001 : unique constraint violated.

(3) Primary key :- P.key. uniquely identifies record in a table

- In all d.b. systems there can be only 1 primary key in a table. and also primary key does not accepts - Duplicate, null values. and also

- when we are creating P.K. oracle server automatically creates "btree" index on p.k. columns.

Column level :-

> Create table d4 (sno number(10) primary key, name varchar2(10));

Table level :-

> Create table d5 (sno number(10), name varchar2(10), primary key (sno, name));

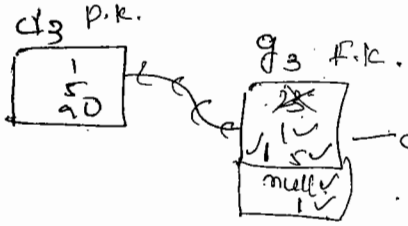
(4) Foreign key :- 24-7-14

- If u want to establishes relationship b/w tables then we must use referential integrity constraint "Foreign key".

→ Here 1 table foreign key must belongs to another table primary key.

→ Here then 2 Columns must belongs to same datatype

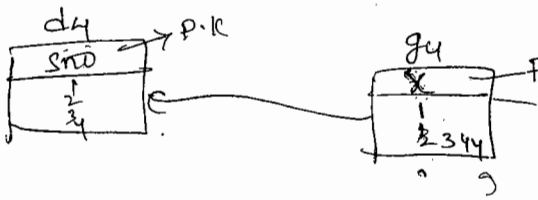
→ Always F.K values are based on P.K. values only.



→ Generally P.K. does not accepts duplicate, null values where as F.K. accepts duplicate, null values.

Column level:-

```
Create table g4
(Sno number(10) references d4(Sno));
```




```
Create table g4
(x number(10) references d4(Sno));
```

→ Column level :- (References) :-

Syntax :-

```
Create table tablename (Col1 datatype (Size)
References mastertablename (Primary key Colname),
Col2 datatype (Size), . . . );
```

> Create table g4 (sno number(10) references d4);

> Create table g4 (x number(10) references d4(sno));

Table level:- (F.K, References):-

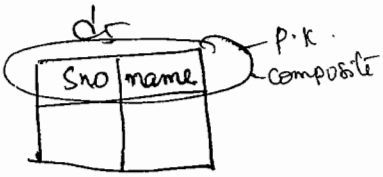
Syntax:-

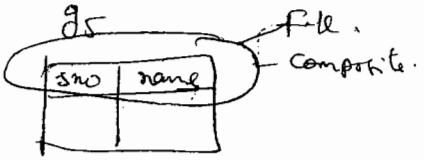
Create table tablename (col1 datatype (size), col2 datatype (size), ... ,

foreignkey (col1, col2, ...)

references

mastertablename (primary col(names));





> Create table g5 (sno number(10), name varchar2(10), foreign key (sno, name) references d5);

→ when ever we are establishing relationship b/w tables then oracle server automatically follows 2 rules.

(1) Deletion in master table.

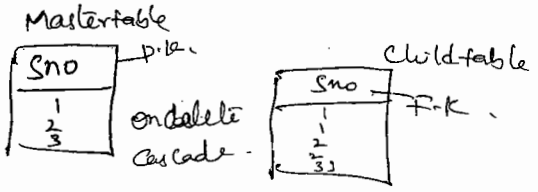
(2) Insertion in child table.

① Deletion in Master table :- In oracle if u are try to delete a master table record in master table then oracle Server returns an error ora-2292 if child table record exist.

— To overcome this problem 1st we are deleting child table record in child table then only we are deleting appropriate master table record in master table.

— otherwise we an "on delete cascade clause"

→ On delete Cascade clause :-



when ever we are using "on delete cascade" clause in child table then if u are deleting a master table record in master table. Then automatically master, child table records are deleted in both the tables. (not all records)

Syntax :-

```
Create table tablename (col1 datatype(size) references
master tablename (P.K. colname) on delete cascade);
```

Ex :-

> Create table mas (sno number(10) primary key);

> Insert into mas values (....);

> Select * from mas;

Sno
1
2
3

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

> insert into tablename) child values (----);

> Select * from child;

Sno
1
1
2
2
3
3

Testing:-

deletion in master table:-

> delete from mas where sno = 1;

1 row deleted.

> Select * from mas;

2
3

> Select * from child;

2
2
3
3

On delete set null:-

Oracle also supports another clause on delete set null in child table.

- when ever we are using this clause if we are deleting P.K. values in master table then automatically appropriate foreign key values are set to null. in child table.

Syntax:- Create table tablename (col1 datatype(size) references master table (primary key colname) on delete set null;

Insertion in child table :- In oracle if we are try to insert other than p.k. values into foreign key then oracle server returns an error ora-2291.

→ Bcz in all d.b. system f.k values are based on p.k. values only.

25-7-14.

⑤ **Check :-** This constraint is used to define logical conditions according to our business rules

Column level :-

Syntax :-

```
Create table tablename (col1 datatype (size)
check (logical condition), ..... );
```

> Create table test (sal number(10) check (sal > 5000));

> Insert into test values (2000);

Error :- check constraint violated (SCOTT.SYS_C005335)

> insert into test values (8000); // 1 row created.

> Select * from test;

SAL
8000

Note :- check constraint does not work with "Sysdate" It allows user defined dates.

> Insert into

> Create table test1(name varchar2(10)
check(name = upper(name)));

> insert into test1 values('murali');

error: ORA-02290: check constraint (SCOTT.SYS_C005336) violated

> insert into test1 values('MURALI');

— 1 row created

Table level :- (AND) operator

> Create table test2 (name varchar2(10), Salnumber(10),
check (name = upper(name) and Sal > 5000));

Assigns user define names to Constraints :-

Generally when ever we are creating Constraint Oracle Server automatically generates an unique identification number for identifying a Constraint uniquely in the formate of SYS_C00056789 (or) SYS_Cn (n = 8)

this is called "Predefined Constraint name".

→ In plate of this one we can also create our own name using Constraint keyword. This is called user defined Constraint name.

Syntax :-

Constraint name

P.K not null
F.K
check
unique

Constraint	user defined name	Constraint type

Predefined Constraint name :-

> Create table test3 (sno number(10) primary key);

> insert into test3 values(1);

> insert into test3 values(1);

error :- unique Constraint.

(SCOTT . SYS_C005338) violated

user defined Constraint name :-

> Create table test4 (sno number(10) Constraint P_fan primary key);

> insert into test4 values(1);

> insert into test4 values(1);

error :- unique Constraint

(SCOTT. ~~P_SAL~~ (SCOTT. P_FAN) violated.

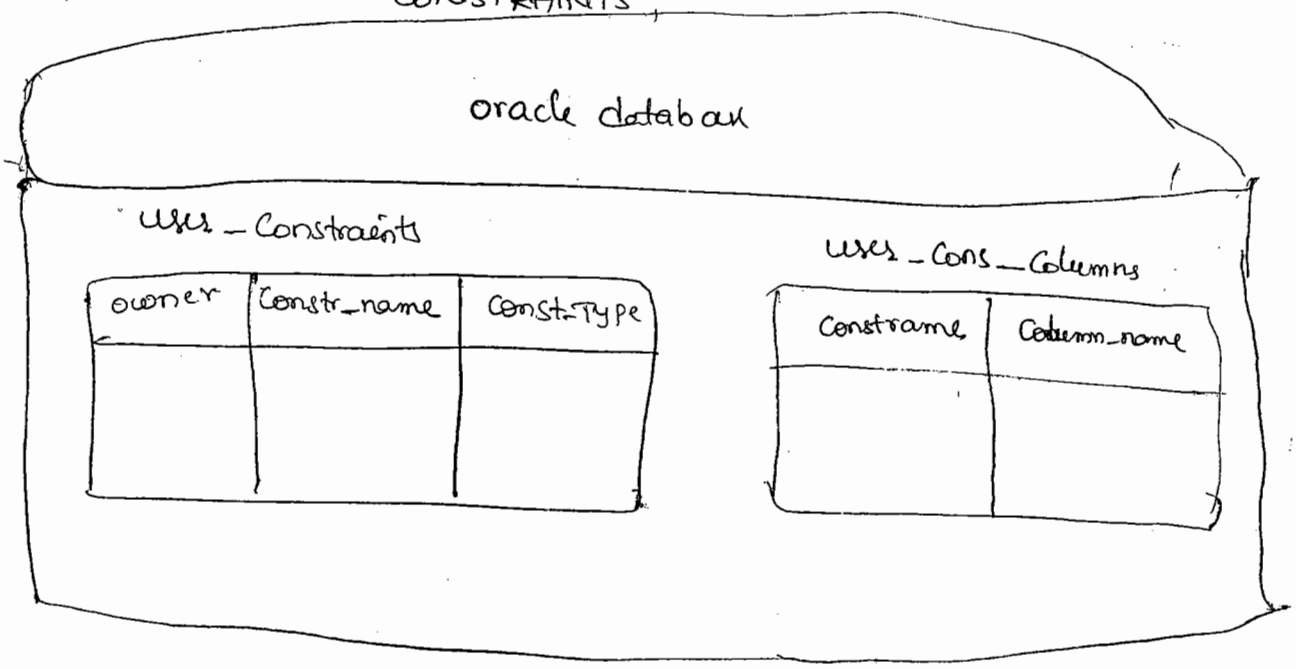
Data Dictionaries :-

when ever we are Installing oracle Server then automatically read only tables are created

— These read only tables are also called as Data Dictionaries

all constraints are stored in

> desc USER_CONSTRAINTS;



Note 1:- In oracle if u want to view Constraint names, Constraint types then we are using "USER_CONSTRAINTS" table

Data dictionary

SQL> desc USER_CONSTRAINTS;

> Select Constraint_name, Constraint_type from User_Constraints where table_name = 'Emp';
(table = caps only)

Constraint_Name	Constraint_type
PK_EMP.	P.
FK_EMP_DEPT.	R.

26-7-14.

Note 2:- If u want to view Column names along with Constraint names then we are using USER_CONS_Columns 'USER_Cons_Columns' data dictionary

Note:- In Oracle if u want to View logical condition of the check constraint then we are using SEARCH_CONDITION property from "user_constraint" data dictionary.

> desc user_constraints;

> Select SEARCH_CONDITION from user_constraints where TABLE_NAME = 'TEST1';

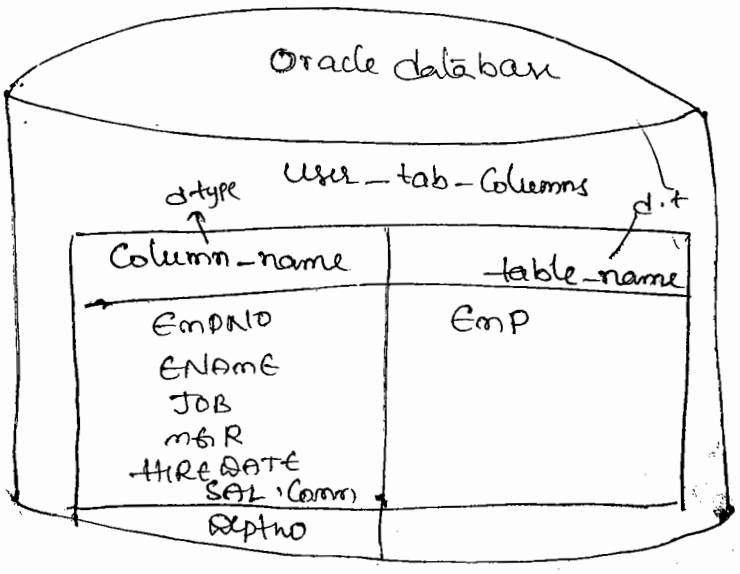
SEARCH_CONDITION

name = upper(name)

→ Select count * emp; no. of rows → o.p. → count(*) = 14

⇒ no. of columns in table.

Note:- In oracle all columns information stored under "user_tab_columns" data dictionary.



Ex:> desc user_tab_columns;

> Select Column_Name from user_tab_columns where table_name = 'Emp';

Column_Name

EMPNO
ENAME
JOB
MGR
HIRE DATE
SAL
Comm
DeptnO

no. of Columns no.

** Interview question:

NOQ to display no. of columns no. from emp table.

> Select Count(*) from user_tab_columns
where table_name = 'EMP';

O.P

Count(*)
8

Default Constraint :-

In oracle we can also create default values for a column using default Constraint.

Syntax :-

Columnname datatype (size)
default actualvalue

O.P:-

> Create table test (name Varchar2(10), Sal number(6) default 2000);

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

> Select * from test;

name	sal
abc	2000 - default

> Creat table test (n

> insert into test (name) values ('abc'), sal = 5000;

name	sal
abc =	5000

- If u want to view default value for a column then we are using DATA_DEFAULT property from user_tab_Columns data dictionary.

> desc user_tab_Columns;

> Select Column_name, DATA_DEFAULT from user_tab_Columns where table_name = 'TEST';

COLUMN_NAME	DATA_DEFAULT
Sal	2000

adding or droping Constraints on existing table:-

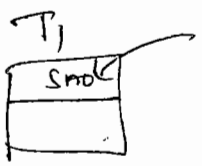
we can also add and droping constraints on existing table using 'alter' Command.

Note:-

new Constraint :- Primary key:-

(1) If u want to add Constraints on existing table existing column then we are using table level Syntax method". (1st Constraint then Colname & Primary (cols))

>

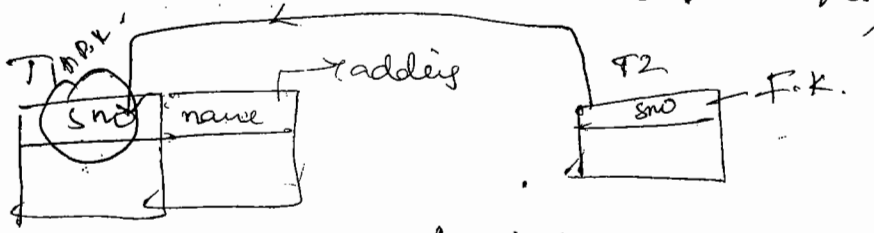


> Alter Table T1. add primary key (Sno);

new Column Unique

(2) If u want to add a new column along with Constraint then we are using "Column level Syntax method"

> alter table t1 add name varchar2(10) unique;



> adding foreign key <- foreign key references } add 2

> Create table t2 (sno) number(10));

<- Table Created

> alter table2 add foreign key (sno) references t1;

NOT NULL:-

Note:- If u want to add not null constraint to the existing table & existing column then we are using alter with modify.

problem to we are modify.

Syntax:-

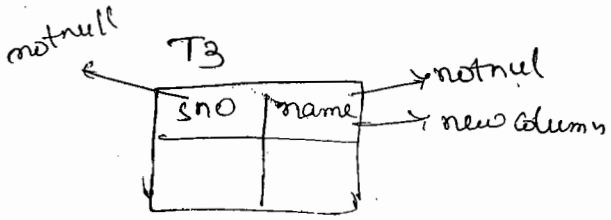
alter table tablename modify Columnname not null;

> Create table t3(sno number(10));

> alter table t3 modify sno not null;

→ new Column:-

> alter table t3 add name varchar2(10) not null;



→ Copy 1 table to another table constraints are not (new) copied.

> Create newemp as ~~comp~~ Select * from emp;

Note

Note:- Generally when we are copying 1 table from another table Con'ts are not copied.

In this case we must use alter cmd which is used to add constnt on existing table (or existing Columns)

> Create table newemp as Select * from emp;

> Create table newdept as select * from dept;

> alter table newdept add Constraint P_kh Primary key (deptno);

> alter table newemp add Constraint P_ab Primary key (empno);

> F.key

> alter table newemp add foreign key (deptno) references newdept (deptno) on delete cascade;

Abs 28-9-14 monday.

dropping Constraints:-

Method 1:-

Syntax:- Alter table tablename drop Constraint Constraint nam

method 2:-

Syntax:- (1) Alter table tablename drop primary key;

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

Eg:- > Create table test (sno number(10) Constraint P-jj primary key);

(or)

> alter table test drop primary key;

Note:- If you want to drop primary key along with referenced foreign key then we are using Cascade clause with alter-drop.

Syntax:- Alter table tablename drop primarykey Cascade;

Eg:- > Create table t1 (sno neember(10) primary key);

> Create table t2 (sno numbes(10) reference t1);

> alter table t1 drop primary key;

— error:- thes unique/Primary key is referenced by some foreign keys.

Solution:-

> alter table t1 drop primary key Cascade;

→ Table alterd

Eg:- > Create table Y1 (sno number(10) Constraint ab notnull);

> alter table Y1 drop Constraint ab;

--- Search constraint name from user-Con-Columns table

M.Imp

Sub-queries

→ Query within another query is called "Subquery" (or) "nested query"

→ Sub queries are used to retrieve data from single or multiple tables. based on more than 1 step process.

→ All database systems have a 2 types of Subqueries

(1) Non co-related subqueries, 1st → Child

(2) Corelated subqueries. 1st → Parent

⟹ In non corelated Sub queries, child query is executed 1st then only parent query is executed. where as in corelated subqueries, parent query is executed 1st, then only child query is executed.

Non Corelated Subqueries:-

- Child Query:- A query which provides value to another query is called child query.

- Parent Query:- A query which receives value from another query is called parent query.

Non-Corelated Subqueries:-

- (1) Single row Sub Query
- (2) Multiple row Sub Query
- (3) Multiple Column Sub Query
- (4) Inline views (or) Subqueries used in from clause

WAQ to display the employees who are getting more than the average salary from emp table.

> Select * from emp where sal >= (select avg(sal) from emp);
2000.50

⇒ Here, child query returns single value. Thats why, this type of subqueries are also called as Single row Sub-query. In single row subqueries we are using =, >, >=, <, <=, <>, (!=) operators.

Execution process:-

Step 1:- > select avg(sal) from emp;

→ 2101.78571

Step 2:- > Select * from emp where (sal >= 2101.78571).

WAQ to display the employees who are working in 'SALES' department using Emp, Dept table;

> Select * from emp where deptno=(Select deptno from dept where dname = 'SALES'); dept = 20.

- 90-1. Same columns
- 5-1. Group functions
- 2-1. Expression
- 3-1. diff. Columns, but of same data type

WAQ to display senior most employee from emp table

Select * from emp where HIREDATE = (Select min(HIREDATE) from emp);

(Hiredate = 14-2-1987)

30-7-14.

WAQ query to display the employees who are working in smith dept from emp table.

> Select * from emp where

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

child query

- WAQ to display the employees who are working under BLAKE was manages from emp table using employee_no, mgr columns.

- 1st write (find out) child query. result in parent query.

- Select * from emp where

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

child claus.

WAQ to display lowest avg salary job from emp table.

Select job, avg(sal) from emp groupby job having

where avgsal = (select low(avgsal) from emp);

sal = select avgsal from

- where clause is a parent claus
- parent clas in having

either where or having is parent query

> Select job, avg(sal) from group by job having avg(sal) = (Select min(avg(sal)) from emp)

Parent query

Child query

nested group fun.

⇒

Error:- ORA – nested group fun without GROUP BY

Note:- In all data base Systems when child query Contains nested group fun's then we must use group by clause in child query.

> Select job, avg(sal) from emp group by job having avg(sal) = (Select min(avg(sal)) from emp group by job)

JOB	AVG(SAL)
CLERK	1137.5

WAQ to display the jobs whose avg salary more than the managers avg salary from emp table.

⇒ Select job, avg(sal) from emp group by job having avg(sal) > (Select avg(sal) from emp where job='MANAGER')

JOB	AVG(SAL)
PRESIDENT.	5000.
ANALYST.	3000.

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

Deptno	Min(sal)
30	1050
10	1400.

Each dept wise max salary,

Select deptno, max(sal) from emp group by deptno.

Deptno	max(sal)
30	2850
20	3000.
10	5000.

KIA@ to display the employee details who are getting max-salary in each department from emp table

— select * from emp where Sal = (select max(sal) from emp group by deptno).

— after where 1 colum so we write sub query 1 Column only

Error: Single-row sub query returns more than 1 row.

Child query.

Allen 1600
Smith 900

900 = (2850, 3000, 5000)

900 = 2850
= 3000
5000

900 in (2850, 3000, 5000) ✓ 900 ≠ ( )

This is a multiple row Subquery bcz here child query returns multiple values,

* In multiple row sub queries we are using "in, all, any" operators

Note: We can also use "in" operator in single row Sub queries.

> Select * from emp
where Sal in (Select max(Sal) from emp group by deptno).

> 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	SMITH
20	3000	FORD
10	5000	KING.

31-07-14.

⇒ WAQ to display the employees who are work in either Sales or Research dept. from emp, dept table.

> Select * from emp where deptno IN (select deptno from dept where dname = 'SALES' OR dname = 'RESEARCH');

{Here = is not allowed, bcz, child query return multiple values
So, we cannot assign multi values to single variable}

Top-N-Anaylis:-

(1) Inline View.

(2) rownum.

(1) Inline View:-

⇒ Oracle 7.2 introduced inline views.

⇒ Generally, we are not allowed to use 'order-by' clause in child query.

⇒ To overcome this problem, oracle 7.2 introduced Subqueries in from clause of the parent query this type of queries are also called as 'inline views'.

Syntax: Select * from (select statement);

(2) rownum:-

→ rownum is pseudo-column. It behaves like a table column

→

- rownum automatically assigns numbers (1,2,3....) to each row in a table at the time of selection.

Eg:- > Select rownum, ename from emp;

ROWNUM	ENAME
1	SMITH
2	ALLEN
3	WARD
4	JONES.

⟹ rownum having temporary values.

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

ROWNUM	ENAME
1	CLERK
2	KING
3	MILLER.

WAQ a query to display 1st record from emp table using rownum,

→ Select * from emp where rownum = 1;

→ WAQ to display 2nd record from emp table using rownum.

✗ Select * from emp where rownum = 2;

O/P:- no rows selected.

rownum does not work with more than 1 positive integer but it work with <, <= operator.

Q:- write a query to display 1st 5 rows from emp table using rownum.

Select * from emp where rownum <= 5;

→ WAQ to display first 5 highest salary employees from emp table using rownum.

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

→ WAQ to display 5th highest salary of employee from emp table using rownum.

> Select * from (select * from emp order by sal desc) where rownum <=5

minus

Select * from (select * from emp order by desc where rownum <=4);

⇒ WAQ to display 2nd row from emp table using rownum,

> Select * from emp where rownum <=2

minus

Select * from emp where rownum <=1;

Eg: > Select * from emp where rownum between 1 and 5.

{ between 1 and 5
rownum <=1 and >=5 } O.P wrt both

> Select * from emp where rownum >=1;

→ rows are selected.

> Select * from emp where rownum >1

→ no rows selected.

emp

Q) WAQ to display last 2 records from emp table using rownum?

> Select * from emp

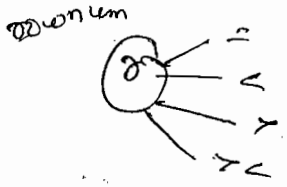
minus

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

Note :- Whenever we are creating a respect a reference name from Rownum in Inne view that alias names works with all SQL operators.

WAQ to display 5th row from emp table using rownum alias name (1-8-14)

```
Select * from (select rownum r, ename,
    Sal from emp) where r=5;
```



R	ENAME	SAL
5	MARTIN	1250.

(or)

```
Select * from (select rownum r, e.* from emp e)
where r=5
```

WAQ to display rows b/w 2 to 9 from emp table using rownum alias name.

```
Select * from ( select rownum r , ename, sal from
emp) where r between 2 and 9;
```

O.P

r	ename	sal
2		
3		
4		
5		
6		
7		
8		
9		

WAQ to display 2nd, 5th, 7th, 9th rows from emp table using rownum alias name.

```
Select * from (select rownum r, ename, sal from emp)
where r in(2,5,7,9).
```

WAQ to display 1st, last row from emp table using rownum alias name. Count(*-1) = 13th record

Select * from (Select rownum r, ename, sal from emp)

where r=1 or r= (select Count(*) from emp)

(or) r in(1,14)..

R	ENAME	SAL.
1	SMITH	800
14	MILLER	1300

— WAQ to display even no. of records from emp table using rownum alias name.

— Select * from (select rownum r, ename, sal from emp)

~~where~~ ~~r%2=0~~ mod(m,n) ~~mod(r,2)=0~~ 2 4 6 8 10 12 14 (%) mod r ∈ r%2=0

where mod(r,2)=0. even

mod(r,2)=1. odd 3 5 1

WAQ to display 5th highest salary employee from emp table using rownum alias name..

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

R	ENAME	SAL.
5	BLAKE	2850.

r ∈ sal

Analytical Functions used in Oracle views :- (All 4.6)

Oracle 8i introduced analytical fun's there are

(1) row_number()

(2) rank()

(3) dense_rank()

These 3 analytical fun's automatically assigns no's to the rows either group wise or rows wise in a table.

Syntax :-

```
Analytical-function name() over
(partition by Columnname order by Columnname [asc/desc])
```

denserank	rank
3000 — 1	1 — 3000
2000 — 2	1 — 2000
2000 — 2	3 — 2000
1000 — 3	4 — 1000

row_number() :- This analytical fun automatically assigns diff. rank no's when values are same where as rank, denserank analytical fun's automatically assigns same rank numbers when values are same. (2000 - 1, 2000 - 1

→ Rank skips next consequelive rank no's

→ where as dense rank doesnot skip next consequetive rank numbers.

Ex:-

deptno	ename	sal	r
20	SCOTT	3000	1
20	FORD	3000	2
20	JONES	2975	3

Rank():-

deptno	ename	sal	r
20	SCOTT	3000	1
20	FORD	3000	1
20	JONES	2975	3.

} 2 skips i.e rank().

denserank():-

deptno	ename	sal	r
20	scott	3000	1
20	ford	3000	1
20	jones	2975	2

Select * from (Select deptno, ename, sal, row_number() over (patition by deptno order by sal desc) r from emp) where r <= 10;

row_number - rank dept

Interview question to display 2nd highest salary employee in each department from emp table using analytical fun.

> Select * from(Select deptno, ename, sal, dense_rank() over (partition by deptno order by sal desc) r from emp)

where r=2;

O.P :-

Deptno	ename	Sal	r
10	Clark	2450	2
20	Jones	2975	2
30	Allen	1600	2.

WAQ to display 5th highest sal employ from emp Table using analytical fun.

- Select * from (Select deptno, ename, sal, dense_rank() over (partition by order by Sal asc)r from emp) where r = 5; — optional - bcz we want 5th highest over all table bt this query 3 depts 5th highest

O.P
10 — 5
20 — 5
30 — 5

but we want only 5th highest over all table.

> Select * from (select deptno, ename, sal, dense_rank() over (order by sal desc)r from emp) where r=5

deptno	ename	sal	r
10	Clark	2450	5

Note:-
In analytical fun's partition by clause is an optional clause.

5-8-11. AAB MIXAAA AB. row add row id

AAB
AAD

rowid :- row address. (deleting duplicate rows)

row id is a pseudo column it behaves like a table column. row id uniquely identifying a record in a table

Table ⇒ Generally when we are inserting data into table then oracle Server automatically creates an unique identification no. for identifying a record uniquely within the block.

Memory (highest block bt meets 5th t

⇒ Using row id we are retriving data very fastly from the database.

emp table lo 1 no. vaste nen o/p lo rownum 3 .... vasthi so changes in rownum

we all ⇒ "Generally 'rownum' having temporary values where as 'rowid' having "fixed" values.

> Select rownum, rowid from emp;

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

⇒ we can also use max, min functions in rowid's

Ex: (1) Select min(rowid) from emp;

(2) Select max(rowid) from emp;

optional ⇒ Generally row id's also used to "delete" duplicate rows in a table.

WAQ to delete duplicate rows except 1 row in each group from a table.

> Select * from test

SNO
10
10
10
20
20
30
30
40
50.

min(rowid)
(10)

> delete from test where rowid not in (select max(rowid) from test group by sno);

not correlated

> Select * from test;

SNO
10
20
30
40
50.

Multiple Column Sub queries:-

In all d.b. Systems we can also Compare multiple Column values of the child query with the multiple Column values of the parent query.

These type of queries are also called as "M.C. subqueries"

multiple.

Parent where clause compulsory exists '()'. Parent query ⇒ where ()

—when we are using multiple Column Subqueries where we must Specify parent query where Conditional

Columns within "paranthesis ()"

```
Select * from tablename
where (col1, col2 ...) in (Select col1, col2 from tablename
                           where condition);
```

WAQ to display the employees whose job, mgr match with the job, mgr of the employees 'scott' from emp-table

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

ename	job	mgr
Ford	Analyst	
Scott	Analyst	

WAQ to display the employees who are getting max. Salary in each deptment from emp table using meltiple row Subquery.

> 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
20	2850	FORD
10	5000	King

> WAQ to display the employees who are getting max. Salary in each department from emp table using multiple Column Subquery.

> 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

WAQ to display Senior most employees in each job from emp table using multiple Column Subquery.

> Select job, hiredate, ename from emp ~~where (job, (job, hiredate)~~ where (job, hiredate) in

(Select job, min(hiredate) from emp group by job).

job	hire date	ename.

WAQ to display ename, dname, Sal of the employees whose Salary, comm match with the Sal, comm of the employees working in the location 'DALLAS'.

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

6-8-11.

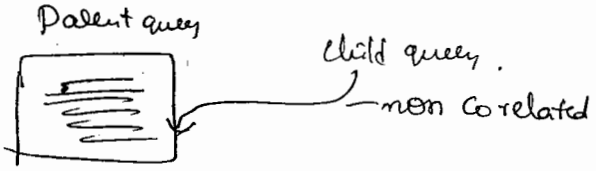
Corelated Subquery:- Generally in non-Corelated subqueries child query is executed first. Then only parent query is executed. where as in Corelated Subqueries parent query is executed 1st then only child query is executed.

⇒ when ever we are Submitting Co-related Sub-query then database Server get a Candidate row from parent query table and then Control passed into child query where clause and based on the evaluation value of the where clause it Compare values in parent query.

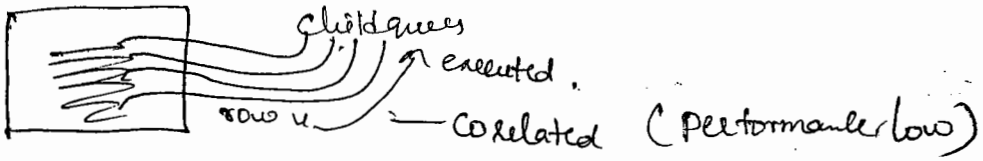
⇒ In Corelated Subqueries we must Create alias name for the parent query table in parent query and use this alias name in child query 'where' Condition.

Syntax :-

```
Select * from tablename aliasname1
Where Columnname operator (select * from tablename
      where columnname = aliasname1.Columnname)
```



In non Corelated Subqueries child query is executed only 1ce per parent query table.



→ where as in Co-related Subqueries child query is executed for each and every row for parent query table.

⇒ Generally Corelated Subqueries are used in de-normalization process. (combined)

In this process they we are using Corelated updates.

→ Generally if we want to modify 1 table Column values based on another table Columns then we are using Corelated update.

Syn:-

```
Update tablename1 aliasname1 Set
Column name = (Select Columnname from tablename2
aliasname1 where aliasname1.Common Colname =
aliasname2.Common Colname);
```

Ex:-

Sol) alter table emp add dname varchar2(10);

> update emp e Set dname = (Select dname from dept d where e.deptno = d.deptno);

> Select * from emp;

O/P:-

ename	eno	Sal	mgr	.....	dname

↑ added Column.

HOQ to display the employees who are getting more than the avg salaries of their jobs from emp table using Corelated Subquery.

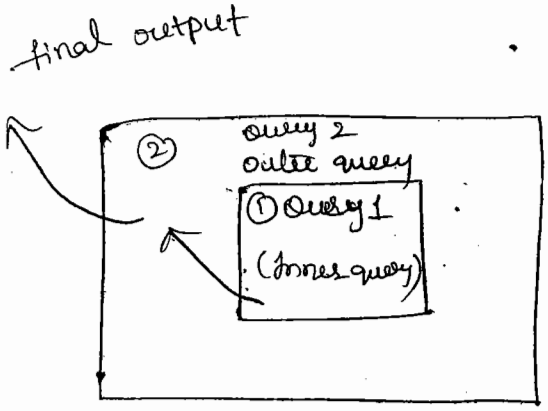
> Select * from emp e (P.Q (outer)) where Sal > (Select avg(Sal) from emp where job = e.job) (C.Query (or) inner query)

empno ename JOB MGR Hiredate Sal comm Deptno.

→ In non Corelated Subqueries always inner query is independent from the outer query, where as in

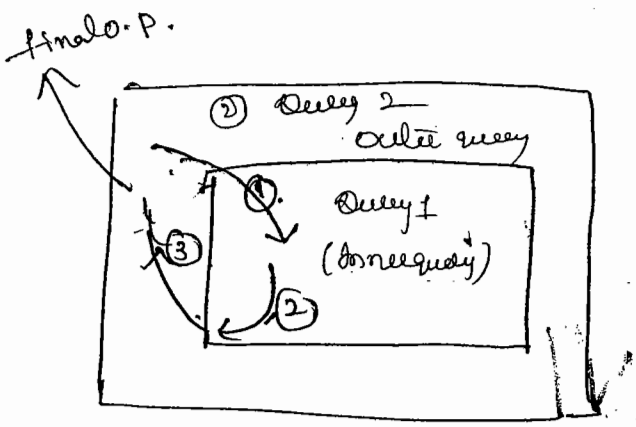
→ In Corelated Subquery always inner query referencing outer query. i.e, Inner query, outer queries are Coreated

non Corelated Subquery



uni direction

Corelated Subquery



Bi-direction

WAQ to display 1st highest salary employee from using Corelated Sub query.

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

o/p:

KING	5100

WAQ to display 2nd highest Salary employee from the following table using Coselated Subquery.

> Select * from test;

o/p

ENAME	SAL
abc	100
xyz	150.
bbb	200
ccc	300.

```
Select * from test e1 where 2= (select Count(*) from test e2 where e2.Sal >= e1.Sal);
```

o/p:-

ENAME	SAL
bbb	200.

Execution proces.

Phase 1:-

Step1:- get a Candidate row (1st row) → abc 100

Step2:- Child → Select Count(*) from test e2 where e2.Sal >= e1.Sal

(4) (100)

with 100
100 = 1
150 = 2
200 = 3
300 = (4)

Step3:- P.Q → 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 >= e1.Sal

(3) (150)

with 150. 150 - 1
200 - 2
300 - (3)

Step 3:- Select * from test e1 where 3=4; (False).

Phase 3:-

Step1:- (bbb 200)

Step2:- e2.sal >= 200.
2

Step3:- 2=2 (true)

returns o/p is 200 bbb 7-8-14.

Note:- when ours resource table contains duplicate data then above query doesnot return values to overcome this problem we must use "distinct clause".

Ex:- > insert into test values ('zzz', 200);

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

— no rows selected. (bcz of duplicate values)

Solution:-

Select * from test e1 where ~~from test~~

2 = (select count (distinct (sal)) from test e2 where e2.sal >= e1.sal

↳ duplicate values will display.

ename	sal
bbb	200
zzz	200

WAQ to display 2nd highest salary employee from the above table using Corelated Subquery (n-1) method

> Select * from test e1 where (2-1) = (select count (distinct (sal) from test e2 where e2.sal > e1.sal)

O.P bbb 200.
zzz 200.

(*) donot use '=' in (n-1)

WAQ to display nth highest salary employee from emp table using Corelated Subquery.

> Select * from emp e1 where &n = (Select Count (distinct (sal) from emp e2 where e2.sal >= e1.sal.

(*) = is like in n-1

> Enter value for n: 1.

KING 5000.

EXISTS operator :- (In Corelated only)

- Exists operator used in Corelated Subqueries.
- Exists operator always returns boolean value either true or false.
- Exists operator performance is very high compare to 'in' operator.
- Exist operator used in 'where' clause only.

(*) When we are using exists operator then we are 'not' allow to 'use Colname'

⇒ We are not allow to use Colname.

Syntax:

```
Select * From tablename aliasname
where exists (select * From tablename
              where Colname = aliasname.Colname);
```

→ exists { } ⇒ False. { } - set means
↳ empty

Child query gives values.

→ exists {1,2,3} ⇒ True.
↳ non empty

⇒ Exist operator is used to test whether a given set is empty or non empty

⇒ Exist operator on empty set returns "false" where as Exist operator on non-empty set returns "True". (child query test true or False)

Ex:

```
exists {1,2,3} ⇒ True
exists { } = False.
```

⇒ Generally if we want to test 1 table Col values are available in another table then we are using Exists operator.

WAQ to display those departments from dept table those departments (employees) are available in emp table using Correlated Subquery Exists

Parent query alias name mandidatory, child query alias name optional

> Select * from dept d where exists (select * from emp where deptno = d.deptno);

O.P:

10	Accounting	N.Y
20	Research	Dallas
30	Sales	Chicago.

⟶ exists always Test whether a given set is empty or non empty in child query. (*, const, 1 or ..., char datatypes)

WAQ to display those departments from dept table does not have employees in emp table using Correlated Subquery. (not exists)

> Select * from dept d where not exists (select * from emp where deptno = d.deptno);

Deptno	Dname	Loc
40	Operations	Boston

WAQ to display those departments from dept table does not have employees in emp table using non-Correlated Subquery.

> Select * from emp;

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

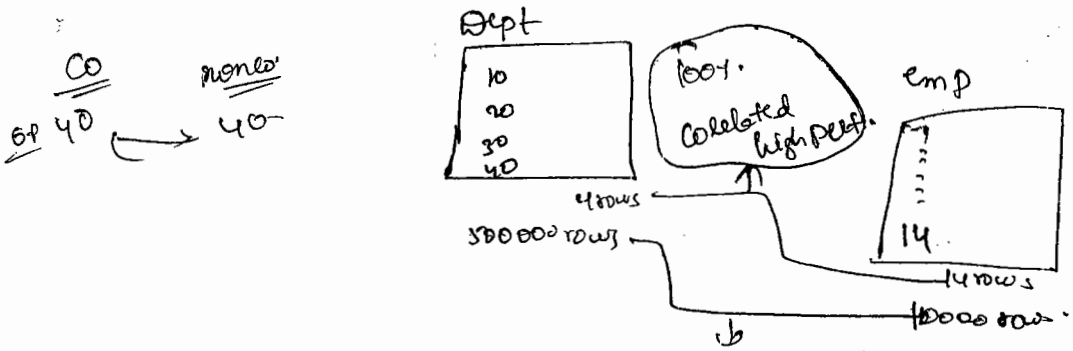
(non Correlated)

This query in corelated also

o/p :- 40 | operations | Boston

o/p :- 40. bt Corelated is very high perfm

⟹



non Correlated high performance.

(100rs)

not in operator does not work with Null values.

Note:- Generally not in operator does not work with null values to overcome this problem we must use not exists operator using Corelated Subqueries.

Ex:- > insert into

emp (empno, ename, deptno) values

(1, 'murali', null);

> Select * from dept where deptno not in

Select deptno from emp);

→ no rows selected.

> Select * from dept d where not exists

(select * from emp where deptno = d.deptno);

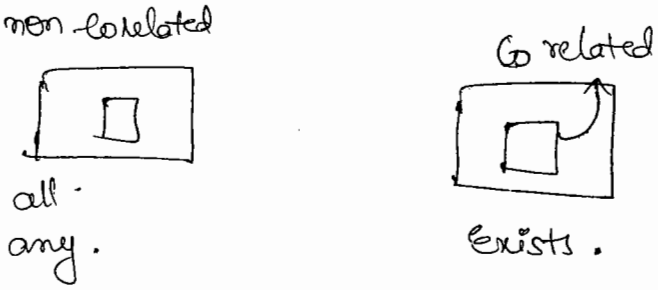
Dno	Dname	Loc
40	operations	Boston

.

In where condition is null, is not null.
Indexes does not Search scenes. workers

8-8-14.

Corelated Sub Query :-



all, any Subquery Special operators used in non corelated Subquery :-

> WAQ to display the employees who are getting more than the highest paid employee in 20th deptno from emp table.

10	20	30
5000	2000	2000
3000	4000	1000
	↓	
	max(sal) > 4000	

> Select * from emp where Sal > (Select max(sal) from emp where deptno = 20) (O.P)

7839	KING	PRESIDENT	Deptno 10

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

> Select * from emp where Sal > (Select min(Sal) from emp where deptno = 10) (O.P) ≡ 6 rows

— when ever resource table having large amount of data, & also child query contains max or min functions & also we are comparing values using relational operators then those types of queries may slowly retrive data from d.b.

This process automatically degrades performce of the application.

TO overcome this problem for improve prfnce of a query, ansi/iso sql introduced subquery special operators. these are "all, any".

These operators are used along with relational opr's (<, >, =, ...)

min or max = all.
max/min = any

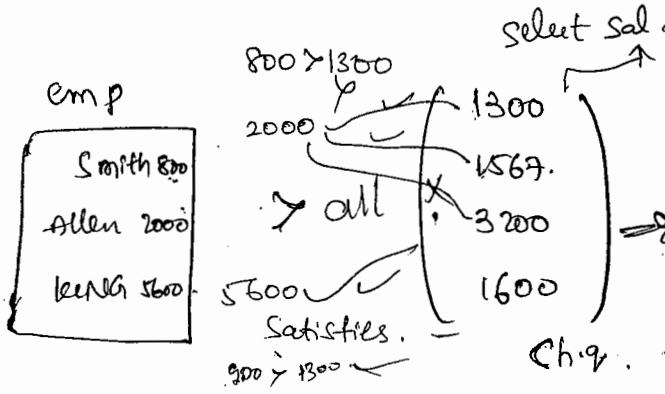
> Select * from emp where

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

> Select * from emp where

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

all, any operators used in multiple row Sub queries :-



In ⇒ It returns same values in the list. → Chi. query

All (><) :- It satisfies all values in the list

Any (><) :- It satisfies any value in the list

more performance than 'In' ← =Any is nothing but internally In op'r

Q) To display the employees who are getting more than the all salaries of the 'clerks' from emp table using Subquery special operator (All)

orderby clause ⟹ not in child query it won't work
↓ thats y we have views.
after child query possible
closing

> Select * from emp where sal >all(Select sal from emp where job = 'clerk') order by sal desc.

O.P :- 3200
3200
5600

without
O.P 3200
5600
3200.

Note:- when we are using Subquery Special opr "All" then optimizer internally uses logical opr "AND"

⟹ when ever we are using Subquery special opr "Any" then internally optimizer uses logical op's "OR"

All = AND
Any = OR .

Ex:- Select * from emp where

dept no > all (10,20);

o/p:- 30.

(or)

deptno > 10 and deptno > 20.

> Select * from emp where

deptno > any (10,20);

o/p:- 20.
30.

(or)

deptno > 10 or deptno > 20

Note:- "=any" is also same as "in" operator.

Bt "<>any" not same as not in operator.

=any = in
<>any ≠ not in

> Select * from emp where deptno
not in (10,20); <> any

O.P :- 30.

> Select * from emp where deptno
<> any (10,20);

O.P :- 10,20,30

<> any
not equal or
≠10 means - 20,30
≠20 & - 10,30
so 10,20,30

Views.

Table, cluster, synonyms — d.b. objects.

Views:- is database object which is used to provides authority level of security.

→ Generally views are created from "base tables"

→ View does not store data View is also called as "Virtual table" & also "Window of table"

user 1
emp

a	b	c	d

user 2

Particular col. only give to another user not all cols. Then restrict cols.
If user to another user we use Views

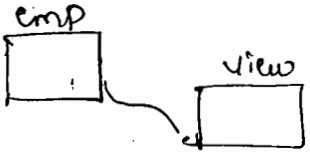
⇒ Generally If u want restrict table colms from 1 user to another user also we are using "Views".

⇒ Based on the Base tables Views are categorised into 2 Views.

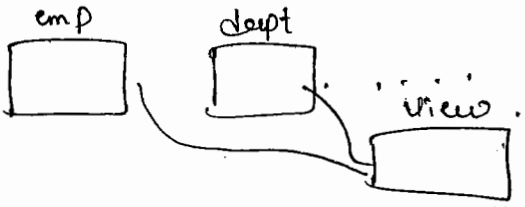
(1) Simple View

(2) Complex View. (or) Join View.

⇒ Simple Views:- Is a view which is created from only 1 base table.



⇒ Complex Views:- Is a view which is created from no. of base tables.



Simple Views:-

Syntax:-

```
Create or replace view viewname as
        Select stmt;
```

(Select stmts are internally stored in user_views)

9-8-14

— In all d.b. Systems when we are creating Views then automatically View definitions are permanently store in database.

⟶ In oracle if you want to view, view definitions then we are using "user_views" data dictionary.

Ex: > desc user_views;

> Select TEXT from user_views where

VIEW_NAME = 'V1';

TEXT

Select "SNO", "NAME" from bar.

> Create a view.

> Create or replace view V1

as

Select * from emp where

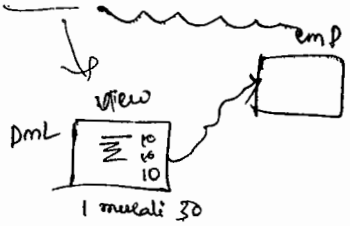
deptno=10;

> Select * from V1;

dml Operations on Simple View (to base-table):-

Rule 1: If a simple view contains group fun's, group by clause, rownum, distinct, set operators, joins then we cannot perform DML Ops through simple view to base-table.

Rule 2:- We must include base table "not null" column into the view then only we can perform "insertion" operation through view to base table.



Ex:- ① Create or replace view V_1

as

select * from emp where deptno = 10;

> Select * from V_1;

> insert into V_1 (empno, ename, deptno)

values (1, 'murali', 30);

1 row created.

> select * from emp;

This 1 murali 30 is available.

Effect in base table

Here in V_1 we can not see this record bcz in where cond'n deptno is '10' so we can see in base table (emp) only.

Example 2:-

> Create or replace view V_2

as

select ename, sal, deptno

from where deptno = 10;

> Select * from V_2;

> insert into V_2 (ename, sal, deptno)

values ('abc', 2000, 20);

Error:- Cannot insert NULL into EMPNO.

There is no empno.

Note

or e

an o

Ex:-

Error:

>

>

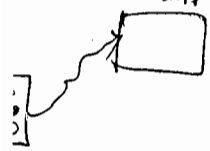
⇒

f null"

n perform

e table.

emp



: 30

Effect in base table

ee for V1

- Can not see record bcz ee Cond'n deptno ' 10

- See in base empl only

ee is no

emp no.

Note:- when a simple view contains group fun's or expressions or rownum then we must create an alias name for these expressions.

Ex:- Create or replace view V3

as

Select deptno, max(sal) from

emp

group by deptno;

Error:- must name this expression with a Column alias

> Create or replace view V3 (1)

as

Select deptno, max(sal) "maximum"

alias name

from emp group by deptno;

> Select * from V3;

(or)

Deptno	maximum
10	5600.
20	3200
30	1525

⇒ alias name creation is 2 types

(1) With " "

(2) after Select & View create alias name like V3 (deptno, maximum)

> Create or replace view V3 (deptno, maxsal)
as
Select deptno, max(sal) from emp
group by deptno; O/P same

alias

> desc V3;

Name	
deptno	
maxsal	

> Create or replace view V4
as
Select rownum num, ename from emp;
alias

> Select * from V4 where num <5;

O/P:-

num	ename
6	(
7	(
8	(
9	(
15	(

WAQ to create a view from Emp table to display each deptnt wise total salary (sum) from Emp, dept tables & also display alias names as dname, total salary.

> Create or replace view V5 (dname, total(salary)
as
Select dname, sum(sal) from empe, dept d
where e.deptno = d.deptno
group by dname;

> Select * from V4;

DNAME	Total Salary
Accounting	9325
Research	10887.5
Sales	7775.

Read only Views:-

→ when ever we are using with read only clause then we can not perform DML operations through view to base table.

Syntax:-

```
Create or replace View Viewname
as
Select * from tablename where condition
With read only;
```

Ex:- > Create or replace view V4
as
select * from emp where deptno = 10
with read only;

testing:- > delete from V4 where deptno = 10;

Error:- Cannot delete from view.

With check option:- If we 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 can insert other than where condition values through view to base table.

Syntax:-

```
Create or replace view viewname
as
select * from tablename
where condition with check option;
```

Ex:- Create or replace view v5
as
select from emp
where deptno=10
With check option;

> insert into v5 (empno, ename, deptno)
values (2, 'abc', 30);

error:- View WITH CHECK OPTION where-clause Violated

> insert into v5 (empno, ename, deptno)
values (2, 'abc', 10);
1 row inserted.

forced
we C
those
Syntax:-
Ex:-
error:-
Sol:-
Warning
> creat
> allo
> des

to provide

s then we

check option

Condition

clause

forced View or force View :- (Interview purpose)

we can also create view without base table those views are also called as force views

Syntax:-

```
Create or replace force view viewname
as
select * from anyname;
```

Ex:-

```
Create or replace view V1
as
select * from Sunday;
```

Error:- table or view does not exist.

Sol:-

```
Create or replace force view V1
as
select * from Sunday;
```

Warning:- View created with compilation errors.

```
> create table Sunday (sno number(10));
> alter view V1 compile;
> desc V1;
```

11-8-14.

Complex View (or) join Views:-

Complex View is a view which is Created from multiple tables.

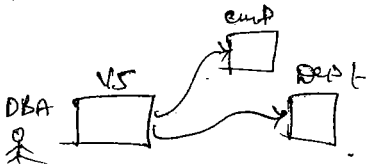
Ex:- Create or replace View V5

as

Select ename, sal, dname, loc

from emp, dept

where emp.deptno = dept.deptno;



— View Created

> Select * from V5

> update V5 set ename = 'abc' where ename = 'SMITH';

1 row updated.

> update V5 set dname = 'xyz' where dname = 'SALES';

Error:-

In all d.b. Systems generally we can not perform DML operations through Complex View to Base tables i.e, Here Some table Colms are effected some other table cols are not effected.

To overcome this problm oracle introduced "Instead of triggers".

By default this triggers are row level triggers.

In oracle if u want to view effectable or non effetable cols then we are using "user_updatable_columns" data dictionary.

Ex:- DESC user_updatable_columns;

> Select COLUMN_NAME, UPDATABLE from usr_updata_ble_ Columns where TABLE_NAME = 'V5';

COLUMN_NAME	UPDATABLE
ENAME	YES
SAL	YES
DNAME	NO
LOC	NO.

TRIGGERS (PL/SQL) :-

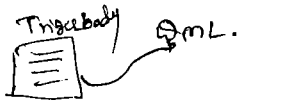
Triggers is also same as "Stored procedure and also it will automatically invoked when ever DML oprns performed on Table.

All D.B. Systems having 2 types of Triggers

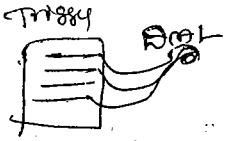
(1) Statement level Triggers.

(2) row level Triggers.

(1) In Statmt level Triggers :- Trigger body is executed only ~~for~~ per DML Stmts.



(2) row level Triggers :- Trigger body is executed for each row ~~for~~ Per DML Stmts.



Syntax:-

Create or replace Trigger Triggername
before/after insert/update/delete on tablename.
[for each row] ← stmt level
→ row level trigger.

→ Trigger Specification

Trigger body { begin ... end; } ← Complsary.

Difference b/w Stmt level, row level triggers:-

Ex:- row level Triggers:-

> create table test (col1 date);

-triggers

is executed

body DML.

related

DML

me.

ssary.

triggers:-

Stmt level trigger :-

```
Create or replace trigger tk1
after update on emp
begin
insert into test values (Sysdate);
end;
/
```

Testing :-

> update emp set Sal = Sal + 100 where deptno = 10.

3 rows updated.

> Select * from test;

Col
11-AUG-14. - ①

> drop trigger tk1;

> delete from test;

Row level trigger :-

```
Create or replace trigger tk2
after upate on emp for each row
begin insert into test values (Sysdate);
end;
/
```

testing:-

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

3 rows updated.

> Select * from test;

COL1

11-AUG-14
11-AUG-14
11-AUG-14.

Row level triggers:- In oracle in row level triggers Trigger body is executed for each row for DML Stmts. That why we are using for each row class in trigger specification.

And also DML transaction values are automatically stored into two rollback stmt qualifiers there are "old, new".

These buffers are used in either in triggers Specification or in trigger body.

When we are using this qualifiers in trigger body then we must use ':' in front of the

we must use ':' in front of the qualifies name

Syntax:-

```
:old.columnname
```

Syntax:-

```
:new.columnname
```

	insert	update	delete
:new	✓	✓	X
:old	X	✓	✓

WAP PL/SQL row level trigger on employee table whenever user deleting data on emp table then automatically deleted data stored in another table.

Ex:-

```
Create or replace trigger
> Create table backup as
  Select * from emp where 1=2;
> Select * from backup;
  no rows selected.
> desc backup;
> Create or replace trigger th1
  after delete on emp
  for each row begin
```

level triggers

for

each row

values are

qualifies

n triggers

fiers is

front of the

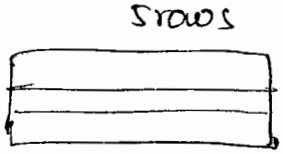
nerpar

```
create or replace trigger th1
after delete on emp
for each row
begin
  insert into backup ;
values (:old.empno, :old.ename, :old.job, :old.sal,
------ :old.deptno);
end;
/
```

testing:-

> delete from emp where sal > 2000;

> select * from backup;



Instead of triggers:-

generally we can not perform DML operations through Complex view to base table.

To overcome this problm oracle 8.0 introduced Instead of triggers.

Instead of triggers are created on views.

-By default Instead of triggers are row level triggers.

Syntax :-

```
create or replace trigger triggername
instead of insert / update / delete on viewname
for each row
begin
=========
end;
```

12-8-14.

Sol:

```
Create or replace trigger tm1
instead of update on V5
for each row
begin
update dept set
dname = :new.dname where
dname = :old.dname;
update dept set loc = :new.loc where loc = :old.loc;
end;
/
```

Testing :-

```
> update V5 set dname = 'XYZ'
  where dname = 'SALES';
  rows updated.
```

> desc user_updatable_Columns;

> Select COLUMN_NAME, UPDATABLE. from

user_updatable_Columns

where TABLE_NAME = 'V5';

COLUMN_NAME	UPDATABLE
ENAME	YES
SAL	YES.
DNAME	YES
LOC	YES.

Materialized Views :-

Oracle 8i introduced materialized views these views are handle by d.b. administrator.

materialized views are used in d.w.h. applications.

generally views doesnot store data where materialized views stores data.

——→ Generally materialized views are used to improve performance of the join or aggregatable queries.

materialized view stores result of the query,

" " " replication (copy) of the remote d.b. into local node.

When we are refreshing mat'zed view it synchronize data based on 'base table'.

mtd view only Admin

Syntax:-

```
Create materialized view viewname
as
select statement;
```

emp

*** Before we are creating materialized view DBA'r must give create any materialized view privilege to user. (Permission)

Syntax:-

```
grant create any materialized view to username;
```

these views

Ex:- Create materialized view mv1
as
select * from emp;

mtd view create only Administrator so when we are create error occurs so we will ask per to DBA. so grant

data where

Error:- Insufficient Privileges.

> Conn sys as sysdba;
Enter password: sys

all used to

> Conn scott / tiger;

egatable

> Create materialized view mv1;
as
select * from emp;

ery, the remote

Materialized view Created.

Note:- In oracle some times views also returns an error insufficient privileges.

To overcome this problem we must give

create any view privilege to any user

Syntax:-

grant create any view to username;

> create or replace view v1

as

select * from emp;

Error:- Insufficient privileges.

> Conn sys as sysdba;

Enter Pswd : Sys

> grant create any view to scott;

> Conn scott/tiger;

> create or replace view v1

as

select * from emp;

View created

returns an

Note: In oracle 1 of the materialized base table must contains primary key otherwise oracle server returns an error.

Ex: Create table test(sno number(10));

> Create materialized view mv1
as
Select * from test;

Error: table 'test' does not contains a primary key Constraint

Diff b/w View, materialized view:-

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

> insert into base values (.....);

> Select * from base;

SNO	NAME
1	a
2	b
3	c
4	d

Create view

> Create or replace view v1
as
Select * from base;
View created.

Create mat'd view.

> Create materialized view mk1
as
Select * from bak;

⇒ In this Case materialized view is also same as view.

— But when ever we are creating mat'd view also mati'd view definitions automatically stored in d.base same like a view definitions.

— In oracle if u want to view then materialized view definitions then we are using user_mviews data dictionary

> desc user_mviews;

> Select Query from
user_mviews where
MVIEW_NAME = 'MK1';

> Select rowid, sno, name from bak.

rowid	sno	name

> Select rowid, sno, name from V1;

In this Case view rowid's are same as bak table rowid's

— That's why view does not store data

— Through the view we can alter bak table data

Thats why Views are also called as "Virtual tables".

> Select rowid, Sno, Sname from mk1;

Here mat'd view rowid's are diff from base table rowid's.

Thats why mater'd views stores data.

> update base set

name = upper(name);

> Select * from V1;

SNo	Name
1	A
2	B
3	C
4	D.

> Select * from mk1;

SNO	Name
1	a
2	b
3	c
4	d

It is not effected

— Mat'd view also stores data same like a table but when we are refreshing materialized view it synchronizes data based on base tables.

— In oracle if u want to refresh mat'd view then we are using refresh method from dbms_mview package.

Syntax:

method name

```
dbms_mview.refresh('materialized viewname');
```

Package name.

> exec (K.W) dbms_mview.refresh('mv1');

SNO	Name
1	A
2	B
3	C
4	D

13-8-14.

Oracle having 2 types of Mat'd views:

(1) Complete refresh mat'd views.

(2) Fast Refresh mat'd views.

(1) Complete Refresh materialized views:-

- In oracle by default mat'd views are Complete Refresh mate'd views
- when we are refreshing then mat'd views then internally rowid's are recreated if u are not modified base table data also.

Syntax:-

```
Create materialized view viewname
Refresh complete
as
select statement;
```

Ex:

>

>

(2)-

The

Incre

high

mat'd

Refresh

Syn

13-8-14.

name');

Ex:- > Select rowid, sno, name from mk1;

> exec

dbms_mview.refresh ('mk1');

> Select rowid, sno, name from mk1;

(three rowids are changed)

② Fast refresh materid views :-

These type of mat'd views are also called as Incremental refresh mat'd views.

These mat'd views performance is very high compare to default mat'd views BCz. In these mat'd views rowid's does not change when we are refreshing mat'd views no. of times also modified

Syntax:-

```
create materialized view viewname
Refresh fast
as
Select statement;
```

- Before we are creating fast refresh mat'd view on we must create mat'd view log on base tables. ⟹

Syntax:-

```
Create materialized view log on base tablename;
```

> Example:-

> Create materialized view log on base;

> Create materialized view mk2
Refresh fast
as
Select * from base;

> Select rowid, sno, name from mk2;

> update base set name = 'xy' where sno = 1;

> Select * from base;

SNO	Name
1	xy
2	B
3	C
4	D

> exec dbms_mview.refresh('mk2');

> Select rowid, sno, name from mk2;

(here rowids are not changed).

> Select * from base;

1	A
2	B
3	C
4	D

(1) M
mat'
is c

(2) Au
using
on con

Syn

mat'd view

base tables

* tablename ;

- Select * from base

1	A
2	B
3	C
4	D

On demand, on Commit :- (With out PL/SQL)

⇒ generally we are refreshing mat'd view in 2 ways

(1) Manually.

(2) Automatically.

(1) **Manually:-** In this method we are refreshing mat'd view using dbms_mview package this method is also called as ondemand method.

By default method is ondemand.

(2) **Automatically:** We can also refresh mat'd views without using dbms_mview package this method is called oncommit method.

Syntax:-

1;

```
Create materialized view view name
Refresh Complete / refresh fast ondemand / onCommit
                  ↓ bydefault      ↓ without writing is also no Prblm   ↓ bydefault.
as
Select Statement;
```

> Create materialized view mks
> refresh fast onCommit
> as
> select * from base;

> Select * from base;

1	xy
2	B
3	C
4	D

> update base set name = 'zz' where sno = 2;

1 row updated.

> Select * from base;

SNO	Name
1	xy
2	zz
3	C
4	D

> Select * from mk5;

SNO	NAME.
1	xy
2	B
3	C
4	D

→ no changes bcz we explicitly gives a commit then only view will be changed.

> Commit;.

> Select * from mk5;

SNO	NAME
1	xy
2	zz
3	C
4	D

DCL (Data Control Language) :-

(1) grant (giving permission).

(2) Revoke (Cancel permission).

Creating a user :-

> Conn Sys as Sysdba ; ↵

Enter Pswd : Sys ; ↵

(or)

> Conn System / manager

Syntax :-

(1) Create user username identified by password;

Syntax :-

(2) Grant connect, Resource to username;

or

DBA

Syntax :-

> SQL> Conn username / password;

> SQL> Conn Sys as Sysdba;

Enter Pswd : Sys.

> SQL> Create user india identified by india;

username Pswd

> Grant Connect, Resource to india;

> Conn india / india.

username Pswd

> Select * from emp; → in india user there is no data.

Error : table or view does not exists.

— from scott user we transfer emp and all data to india user

> Conn scott/tiger;

> grant all on emp to india;

> Conn india/india;

> Select * from emp;

error:- table or view doesnot exists.

> Select * from scott.emp; (username.object).

> Create synonym xy for scott.emp;

> select * from xy;

⟹

Data Security point of view all d.b. Systems having 2 types of privileges. (Permission)

(1) System privileges.

(2) Object Privileges.

14-8-14.

(1) System privileges:- System privileges are given by database administrator these privileges are used to Create or alter database objects.

India user

Oracle having more than 80 System privileges, These are :- create table, Create Session, Create procedure, Create any materialized view, Create any index. Create any index, Create public Synonym. ---- 5

Syntax :-

```
grant System privileges to username1, username2, ...;
```

object).

> conn sys as sysdba;

Enter password : sys.

> grant Create procedure Create any materialized View, Create any index to Scott, murali, India;

ons having

Note :-

In oracle If we want to Connect any user to the Server d.b. then we are using 'Create Session privileges'

> ~~Grant~~ Create Session is used to Connect to Server

given by

ROLE :- advantage only admin use.

used to

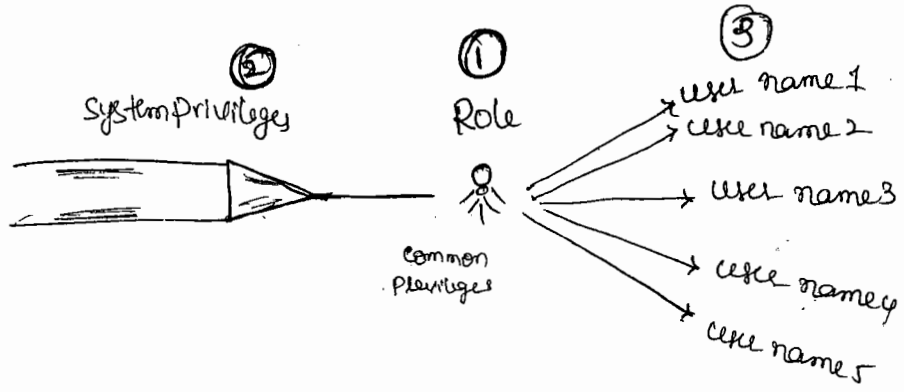
Role is nothing but Collection of either System or Collection of object privileges.

User 1

⟹ User defined roles are created by DBA's.

⟹ In multiuser environment no. of users works on same project.

In this case some users requires common (same) set of privileges. In this case only DBA'r creating a user defined Role & assigns common set of privileges into Role & then only that role given to the no. of users..



Creating user defined roles:-

Step 1:- Creating a role:-

Syntax:- Create role any name;

Step 2:- assign a System privileges to rolename:-

Syntax:- grant System privileges to rolename;

Step 3:- assign role name to no. of users:-

Syntax:- grant rolename to

(username1, username2, ...);

> Conn Sys as Sysdba;

Enter Password — Sys.

> create role ab;
role name

> grant create procedure create any materialized view, create trigger to ab;

> grant ab to

Scott, murali, India;

⟹ In oracle if we want to view system privileges related to role then we are using **role_sys_privs** data dictionary.

Example:-

> desc role_sys_privs;

> Select ROLE role, privilege from role_sys_privs

where ROLE = 'AB';

Predefined roles:-

when ever we are installing oracle server then automatically 3 predefined roles are created.

These are (1) Connect —designed for→ end users purpose

(2) Resource ——→ developer "

(3) DBA ——→ data base Administrator Purpose.

> desc role_sys_privs;

> Select ROLE, PRIVILEGE from
role_sys_privs
where ~~role~~ ROLE
in('CONNECT', 'RESOURCE');

> Select ROLE, PRIVILEGE from role_sys_privs.
where ROLE
in('DBA');

DBA
↓
create master
↓
resource
↓
basic
d.d.
create table

Object privileges:-

Object privileges are used to perform some operation of the object.

These privileges are given by either d.b. Developer or DBA'r.

Oracle having following object privileges these are {insert, delete, update, select}

These 4 are also called as **all** in oracle side

execute, read, write, ... (PLISQL side)

Syntax:-

```
grant object privileges on objectname
to usernames / role name / public;
```

> conn scott/tiger;

Connected.

> grant all on emp to murali, india;

> grant all on emp to ab;

> grant all on emp to public;

With grant option:- (XX not use roles)

who receives with grant option clause those users giving same object previleges to another user.

Syntax:-

```
grant object privileges on object name
to username/ public with grant option;
```

> grant all on emp to india, murali with grant option;

Grant Succeeded.

> grant all on emp to ab with grant option;

Error:- Cannot GRANT to a role with grand option

Note:-
In all d.b. system roles does not work with grant option.

18-8-14.

In oracle, if we want to view object privileges related to user then we are using user_tab_privs d.d. SQL> desc user_tab_privs;

Revoke:-

It is used to cancel either system or object privileges from the user

Syntax:-

```
Revoke System privileges From username1, username2 ... ;
```

Syntax:-

```
Revoke objectPrivileges on objectname
From usernames / Rolename / Public;
```

In all d.b. systems if u want to restrict table columns from 1 user to another user then only we are using views.

i.e, d.b. administrators creating a view with required columns then the view given to the no. of users.

Syntax:-

```
grant all on Viewname to
username1, username2 ... ... ;
```

> Create or replace view V1
as
Select empno, ename, sal from emp
where deptno=10;

> select * from V1;

> grant all on V1 to india;

> Conn india/india;

> Select * from scott.V1;

**

⇒ We can also drop view using drop view view name;

⇒ When ever we are droping a base table we can not access a view.

But view definitions are still exists in d.b.

Ex:- > Create or replace view V1
as
select * from base;

> Select * from V1;

> drop table base;

> desc user_views;

> Select Text from user_views where
VIEW_NAME = 'V1';

Text

Select "SNO", "NAME" from bank.

MERGE Stmt: (DML) [9i]

oracle 9i Introduced "Merge Stmt".

- This Stmt is used to transfer data from source table into target table.
- But here these 2 tables Structure must be same
- merge stmt used in datawarehousing applications.
- In merge stmt we are using update, insert stmt that's y this stmt is also called as:

"UPSERT" stmt.

- In merge stmt we must create reference name for the table.

Syntax:-

```
merge into target table name
using source tablename
on (Join condition)
when matched then
update set target table col1 = source table col1, .....
when not matched then
Insert (target table colnames) Values
(Source table colnames);
```

> Select * from dept ; (target table);

10
20
30
40

> select * from dept;

> Create table depts as select * from dept;

> insert into depts values (1, 'a', 'b');

> select * from depts; (source table);

10
20
30
40
1 a b

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);

> Select * from dept;

deptno	name	loc
10		
20		
30		
40		
1	a	b

Note:- In merge stmt we can not update "On clause columns".

d.deptno = s.deptno — error.

SEQUENCE: (Independent Obj)

→ 'Sequence' is a d.b. object which is used to generate sequence no's automatically.

⟹ generally sequences are used to generate primary key values automatically.

⟹ Sequence is an Independent d.b. object once a sequence has been created no. of users simultaneously access that sequence.

Syntax:-

```
Create Sequence  sequencename
 start with n
 increment by n
 min value  n
 max value n
 cycle / nocycle
 cache / no cache;
```

we want to generate Sequence values then we are using 2 pseudo columns.

(1) Currval
(2) nextval.

Syntax:-

```
SequenceName.Currval;
```

Syntax:-

```
Sequencename.nextval;
```

These pseudo colms are used in Insert, update, delete, Select Statements (Dml)

→ In oracle if we want to generate Sequence values using select stmt then we must use dual-table.

Syntax:-

```
Select sequencename.Currval from dual;
```

Syntax:-

```
Select sequencename.nextval from dual;
```

➢ Create Sequence S1
Start with 5
Increment by 2
maxValue 100;

> Select S1.currval from dual;

Error:
Sequence S1.currval is not yet defined in this Session.

> Select S1.nextval from dual;

5

> Select S1.nextval from dual;

7.

— In all d.b.sys' if u want to generates 1st sequence no. then we must use next val pseudo colmn:
— BCz Currval pseudo colmn returns (Current Value of the Sequence if Sequence Session already having a value

(or)

> Create sequence S1
start with 5
increment by 2
max value 100;

> Select S1.nextval, S1.Currval from dual;

Nextval	Currval
5	5

Note:- If u want to view default values for a sequence then we are using user_sequences d.d.

Ex:- Create Sequence S1

Start with 1;

> Select S1.nextval from dual;

1.

> desc user_sequences;

> Select * from user_sequences;

Note:- We can alter all sequence parameter values except "Start with"

Syntax:-

alter sequence S1
Start with 7
Increment by 1
min value 5
max value 20;

> Select S1.nextval from dual;

7.

> Select S1.nextval from dual;

8.

> alter sequence S1
incremented by -1;

= Sequence altered.

> Select S1.nextval from dual;

7

> alter sequence S1

start with 8;

error:- Cannot alter starting sequence number.

20-8-14

Note:- Start with cannot be less than min value.

> Create sequence S1
start with 4
incremented by 1
minvalue 5;

error:- START WITH cannot be less than MINVALUE

Cycle / nocycle:-

> Create sequence S1
Start with 5
incremented by 1
minvalue 4
max value 8
cycle;
no cache;

> select S1.nextval from dual;

o.p:- 5.

> / 6 ... 8 ...

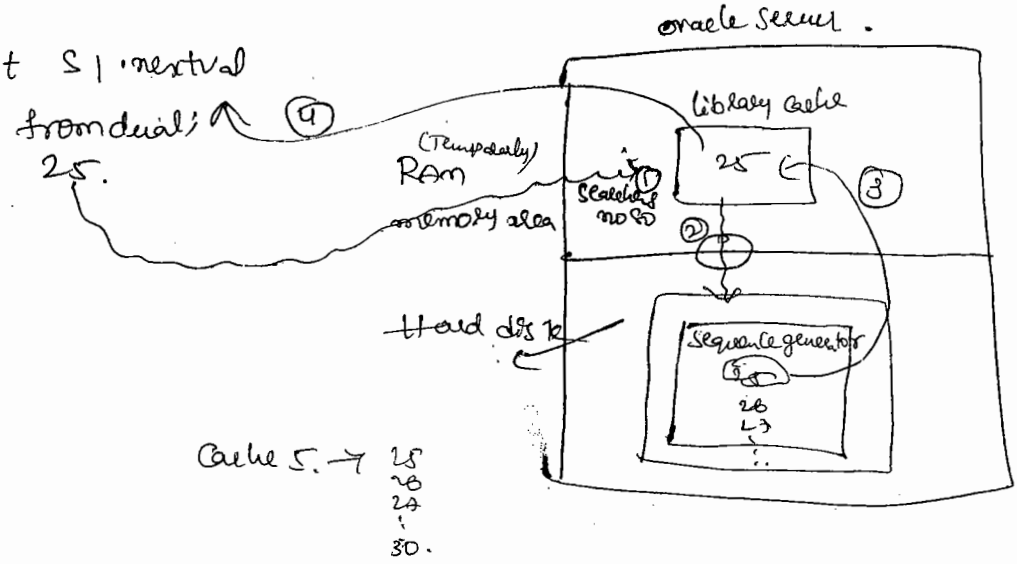
Nocycle:-

> create sequence S1
start with 5
incremented by 1
minvalue 4
max value 8.
nocycle
no cache;

o.p:- 5
6
7
8
error:-

4 5 6 1
4 5 6 1

> error:- Sequence S1.NEXTVAL exceeds MAXVALUE and cannot be instantiated.



Cache:- Cache is a memory area which is used to access Sequence values very fastly.

→ generally if we want to improves performance of the application then only dBA define Cache values for the Sequence through Cache option.

⇒ Cache is a memory area which is used to store Set of Sequence no's

⇒ generally when ever we are creating Sequence the d.b. Servers generates Sequence values using Sequence generator within Hard Disk.

→ when ever we are requesting Sequence value then Server process checks requested Sequence no's are available or not available in Cache memory Area

⟹ If 1st time Sequence values are not available in cache memory area then only Server process Searching cache values in disk & then only those values are transfered into cache memory area.

⟹ This process automatically (reduces) & degrades perfmce of the appln

→ To overcome this problem we are defining set of Sequence no's using cache option

In oracle by default Cache Values 20 and also cache min Value is '2'

Note :- When ever Server is shut down or System crashes then automatically cache values are "losted"

> Create Sequence s1
- Start with 1
increment by 1
Cache 1;

Error :- the no. of values to CACHE must be greater than 1

→ Generally if u want to generate primary key values automatically then only we are using Sequences

Ex:- Create table test (sno number(10) primary key,
name varchar2(10));

> Create sequence s1
start with 1;

> insert into test (sno, name)
values (s1.nextval, '&name');

→ Enter value for name : abc
: xyz.
: klm.

> Select * from test;

sno	name
1	abc.
2	xyz.
3	klm.

⟹ All sequences information stored under "User_Sequences" d.d.

> desc user_sequences;

→ we can also drop. sequence by using ~~drop~~
drop sequence Sequence name;

Locks

- Locking is a mechanism which prevents unauthorised access for our resources (tables, d.d ....)

All d.b. Systems having 2 types of locks.

(1) Row Level Locks.

(2) Table Level Locks.

(1) Row Level Locks :-

In this method we are locking set of rows by using "for update" clause.

→ This clause is used in Select stmt only.

→ In oracle Row Level Locks are introduced in Oracle 6.0.

Syntax :-

```
Select * from tablename
Where condition for update [no wait];
                            ↳ optional
```

⇒ rows are locked bt it doesnot display any msgs bt internally rows are locked. Bt DML operations working bcz authorised person accessed bt unauthorized users cannot perform DML oprns. Select only working

⇒ when ever we are performing locks another user query (select stmt) the data but they can not perform DML Operations. & also

In all d.b. Systems when ever we are using Commit then only automatically locks are released.

Scott/Tiger

> Select * from emp where deptno = 10 for update;

> Commit; (or) > rollback;

[For releasing Locks].

murali/murali

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

[We cannot perform on LS]

21-8-14.

Nowait:-

nowait is an optional clause used along with "for update" clause.

→ when ever we are using this clause oracle server automatically get the control into current session if another user not releasing locks also.

→ In this case oracle server returns an error ORA-54 resource busy.

Scott/tiger

> Select * from emp where deptno = 10 for update nowait;

ORA - 00054: resource busy.

murali/murali.

> Select * from Scott. emp where deptno = 10 for update;

Table level Locks :-

In this method we are locking a table, table level locks are handled by DBA's only.

In oracle d.b. developers uses 2 types of table level locks. (1) Share lock.

(2) Exclusive lock.

(1) Share lock :- (more than 1 use)

when we are using this lock another user query the data but we can not perform DML operations & also no. of users locks the resource (table) still waiting all the users release the locks.

Syntax:-

```
lock table tablename in share mode;
```

> show user;

user is scott.

> lock table scott.emp in share mode;

Scott | Tiger

> lock table emp in share mode;

murali | murali

> select * from scott.emp; (✓)

> lock table scott.emp (another locks) in share mode; (✓)

> update scott.emp set sal = sal + 100;

— (we cannot perform DMLs)

② Exclusive Lock:- (only 1 user like row level lock)

when we are using this lock another user query the data but they can not perform DML operations.

& also at a time only 1 user lock the resource

Syntax:-

```
lock table tablename in exclusive mode;
```

Scott/tiger

```
> lock table emp
  in exclusive mode;
```

murali/murali.

```
> select * from scott.emp (✓)
> lock table scott.emp
  in share mode ; ( X )
  Exclusive mode ; ( X ) any locks are working
```

Streeks to commit

Note:- In oracle when ever we are using DML stmts then automatically oracle server internal uses Exclusive locks [same data performs then only ex. locks applied in same user]

Scott/Tiger

```
> update emp set
  ename = 'abc'
  where ename = 'SMITH';
```

Scott/tiger.

```
> update emp set ename='abc'
  where ename = 'SMITH';
```

[Can not perform DMLS bcz internally server uses exclusive locks]

NORMALIZATION.

→ "Normalization" is a "Scientific process". (designers only use)

→ which is used to decomposing a table into no. of tables.

→ This process automatically reduce redundancy (duplicate data) and also automatically avoids insertion, updation, deletion problems.

⟹ In Design phase of the SDLC d.b. designers use "normalization" process i.e.,

→ In Design phase of the SDLC D.b. designers designs logical model of the database.

→ In this logical model only designers uses normalization process through "Normal Forms".

⟹ In 1970 "E.F.CODD" written a paper "relational model of data for large shared data banks".

In this paper only E.F.CODD introduced 1st 3 Normal Forms.

(1)

Candida
Ke

Normal Forms :--

(1) first normal form

(2) Second normal form

(3) third normal form.

(4) BCNF

(5) Fourth normal form.

(6) Fifth normal form.

(7) → {Domain key NF}

(1) First Normal Form :-

Item table. → multiple values

Item name	Color	price	tax.
marker	black, red	20	0.02
pen	blue, green	30	0.03

cell1 ← marker

cell2 ← pen

Not in 1st NF

↓ 1NF

Item table.

duplicates ko

Candidate key

Item name	Colr	price	tax
marker	black	20	0.02
marker	red	20	0.02
pen	blue	30	0.03
pen	green	30	0.03

⇒ A table is in 1NF in that table every cell having an atomic value and also.

② Identifying a record uniquely using a key

(Grocery management) (Crucial for electrical shop)

Order Form 1

- Order no
- Order date
- Cust name
- Address
- Ph.no
- Item name
- Amount

Submit

F 2.

- Ord.no
- Ord
- Custn
- add
- Ph.no
- I1
- A1
- I2
- A2
- I3
- A3

Submit

(1)

Child table

Order no	O: d:	Cus na	add	Ph	I n	A

(2)

O.no	O: da	Cu om	ad	phno	I1	A1	I2	A2	I3	A3

not possible in d.b.

Order Form 3

- Order no
- Order date
- Cust name
- Address
- Ph.no.
- Item name

Refrigerator
LED TV

☑ add another item

Amount

Submit

P.K.

Order master Table

Order no	Or.date	Cus name	add	ph.no

Orders detailed Table

F.K.

1NF Table

Ord.no	Item name	Amount

(Child Table)

bcz duplicates allowed

Every Cell

a key

child table

add phy [illegible] A

[illegible] A2 [illegible] [illegible]

Possible in d.b.

no

d-Table

366

Amount

[illegible]

Process:-

Identifying repeating groups & puting into seperate table in more automic form.

⟹ By default 1NF process table is a child table. Bcz in this table 1 col having duplicate data (must)

22-8-14.

2nd Normal Form :-

→ A Table is in 1NF & also all non-key attributes are "Fully Functionally" dependents on total Candidate Key.

→ Always 1st normal form deals with atomacity where as 2nd NF deals with Relation Ship b/w Key, non key attributes.

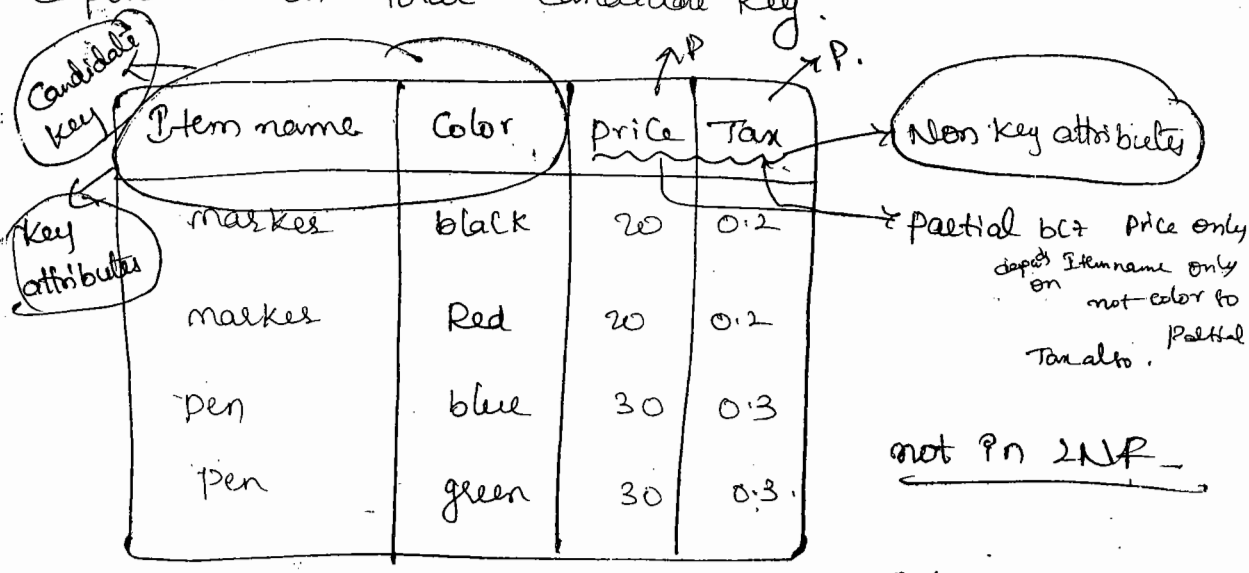
→ If a table is in 1NF and also the table Contains Partial non-key attributes then that table not in 2NF.

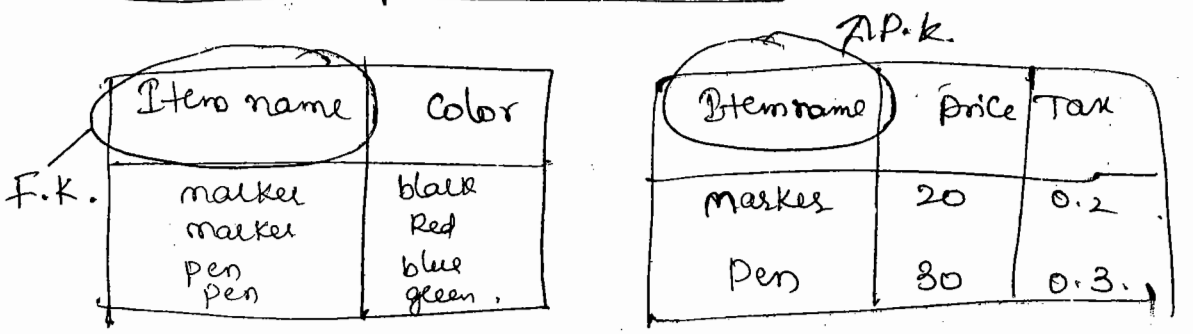
Process :-

→ Identify partial Non-key attributes which depends on Partial key attributes. Those all attributes put into seperate Table.

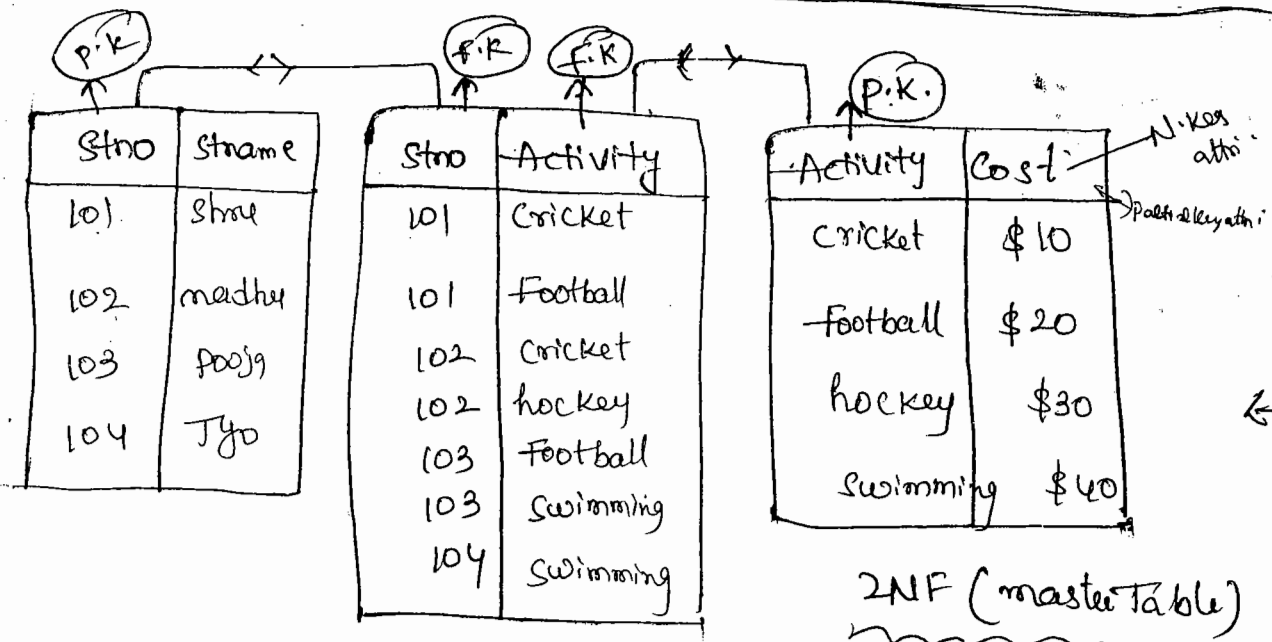
→ The seperate Table is called 2NF Table.

→ By default this is a master table (bcz no duplicates). In this table only all Non key attributes are Fully Functionally dependent on total Candidate key.





(master table) 2NF.



2NF (master Table)

Student Sports Activity table.

Sno	Sname	Act1	Cost1	Act 2	Cost
101	Shruthi	Cricket	$10	Foot ball	$20
102	madhu	Cricket	$10	hockey	$30
103	Pooja	Football	$20	Swim mig	$40
104	Jyothsna	Swimming	$40		

Sno	Sname
101	Shruthi
102	madhu
103	Pooja
104	Jyo

table.

Sno	Act1	Cost1	Act2	Cost2
101	Cricket	$10	Football	$20
102	Cricket	$10	hockey	$30
103	Football	$20	Swimming	$40
104	Swimming	$40		

1NF.

candidate key

Key attribute

Non key attribute

Sno	Activity	Cost1
101	Cricket	$10
101	Football	$20
102	Cricket	$10
102	hockey	$30
103	Football	$20
103	Swimming	$40
104	Swimming	$40.

not in 2NF

depends on activity

Sno	Sname
101	Shru
102	madhu
103	Pooja
104	Jyo

2NF

2NF

Convert to 2NF

key attribute Candidate key P P non key attributes

Ecode	projCode	Dept (only Ecode)	Depthead (only code)	Hours (Ecode + Projcode) → F
E101	P1	Systems	D1	3
E102	P1	SALES	D2	5
E101	P2	Systems	D1	8
E103	P2	Research	D3	9
E102	P3	SALES	D2	10
E101	P3	Systems	D1	7

not in 2NF.

Insertion E105

Phase 1:-

Functional dependency Relation table

FD

① Ecode → hours (X)

Code	
	3
	8
	7

FD

② projCode → Hours (X)

P1 → 3
P1 → 5

FFD

③ Ecode + projCode → Hours (✓)

Phase 2:-

FD

① Ecode → Dept (✓)

E101 → systems

② projCode → Dept (X)

③ F-8 20

Phase 3:-

FD

① Ecode → Depthead (✓)

② projcode → Depthead (X)

P.K

2NF

Ecode	dept	Depthead
E101	Systems	D1
E102	Sales	D2
E103	Research	D3

2NF master table

F.K.

Ecode	projcode	Hours
E101	P1	3
E101	P2	8
E101	P3	7
E102	P1	5
E102	P3	10
E103	P2	9

1NF (Child table)

23-8-14.

→ Before Normalization process above resource table having insertion, updation, deletion problems

① Insertion problem :-

In the above resource table when we are try to insert particular department employee details then we must assign project code. (1st day company does not assign project) SO)

This is called "Insertion problem".

② updation problem :- In the above resource table Ecode, dept, depthead attribute values are repeated whenever particular dept employee transfered 1 dept in another dept

then we must modify all then 3 attribute values correctly. (update sales dept to ex:- Bkw).

← Otherwise inconsistency Prblm occured. This is called "updation Prblm".

(3) Deletion prblm:- In the above resource table when we are try to delete a particular employee then automatically deptmt details also deleted. This is called "deletion problem".

⇒ when ever we are using normalizes process then automatically insertion, updation, deletion problems are avoided (Invid)

3NF.

⟶ If a table is in 2NF & also all Non-key attributes are "only" dependents on Candidate (or) (P.K) key

⟶ If a table is in 2NF & also if Non-key attributes which depends on another Non-key attributes then that table "not in 3NF".

attribute

This is

table

employee then

is called

ies ties

olems

n-key

or
(P.K) key

ttributes

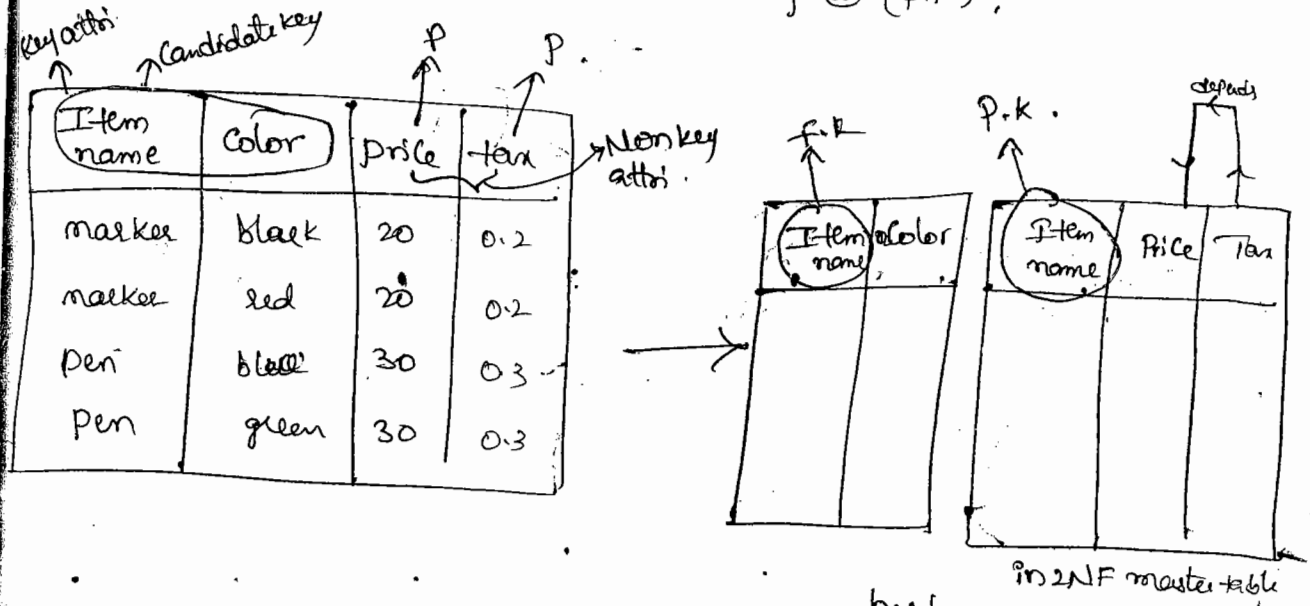
that

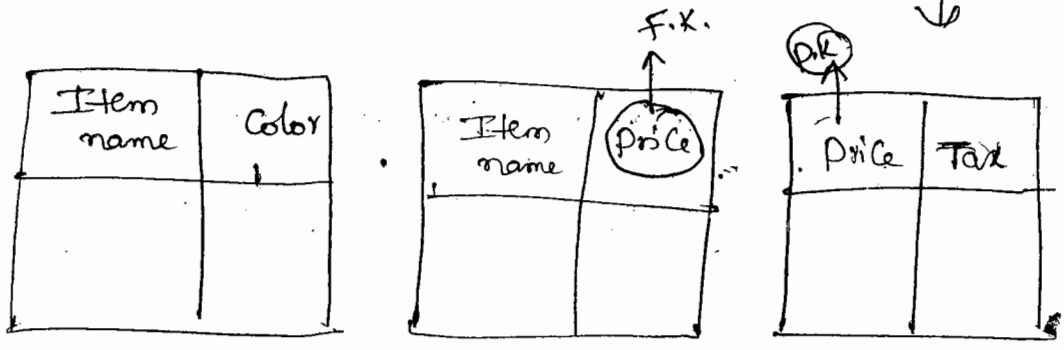
Process :-

→ Identify Non key attributes, which depends on another Non-key attributes put into seperate table.

⇒ This table is called "3NF table". By default this table is master table.

→ In this table only all non-key attributes are only dependents on Candidate key (or) (P.K).





master table (3NF table).

Order form.

Order no

order date

Cust no

Customer Enquiry Form

Cust. no

Cust. name

address

Ph. no

Submit

qty

Discount

Item name | Item no

Amount

add another Item

Submit

P.K Customer master table (3NF)

Custno	Cust' name	addres	Ph.no

P.K order master table F.K

orderno	orderdate	Custno

Item master

Item no	Item name	Amount

order detailed master table.

Order no	Item no.	qty	Discount

25-8-14.

Key attribute → Candidate Key

2NF

(not in 3NF) ← N.key FD on N. key

→ Non-key attributes.

Sname	courseid	Grade	Gradevalue
abc	CS111	A	4.00.
xyz	CS111	B	3.00
zzz	CS222	C	2.00
aaa	CS222	A	4.00

Student table.

Sname	Courseid	Grade (F.K.)
abc	CS111	A
xyz	CS111	B
zzz	CS222	C
aaa	CS222	A

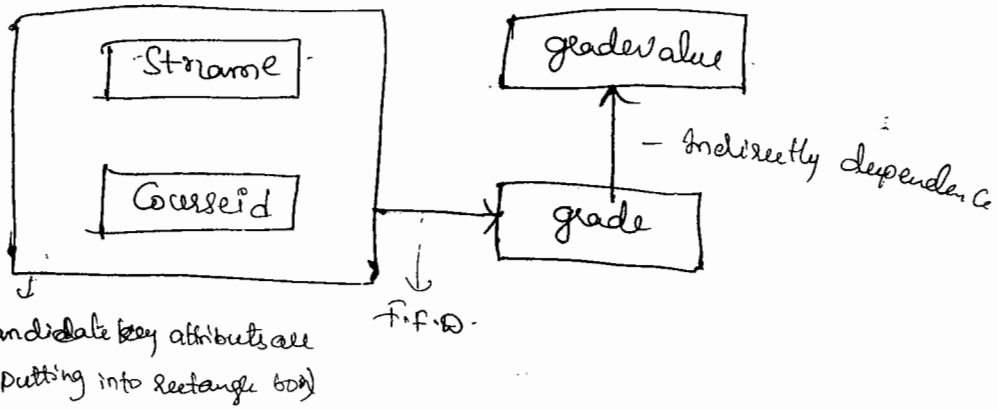
(Child table)

Grade (P.K.)	Gradevalue
A	4.00
B	3.00
C	2.00

master table

3NF

logical Diagram.



Functional Dependency (uniqueness) :- (FD) :-

If any given 2 tuples in a relation 'r' then, X (an attribute set) agrees then Y (another attribute set) cannot disagree (agree) then $X \rightarrow Y$ is called "F.D."

$X \rightarrow Y$.

here Y is functionally dependent on X ' $(X \rightarrow Y)$

(or)

X functionally determines 'Y' $(X \rightarrow Y)$

	Sno	Sname (attributes)	Hall
t1 tuples	S_1	abc	H1
t1 →	S_2	xyz	H1
	S_3	aaa	H2

Relation 'r'.

1NF :- Remove repeating groups

2NF :- Remove partial attributes

3NF :- Remove attributes which are not dependent on Candidate key (P.K).

(or)

Remove non key attributes which are dependent on another non-key attributes.

1st & 2nd & 3rd NF.

Course id ☐ Course name ☐

tutor id ☐ tutor name ☐

Stno	Stname	D.OB	Gender	last attendence

UNF

- CourseId
- Coursename
- tutor id
- tutor name
- Stno
- Stname
- D.OB
- Gender
- last attnce

1NF

Course id

Course name

Tutor name

Stno

Sname

D.O.B

Gender

last attnce

not in 2NF

- Sname → P
- D.O.B → P
- Gender → P
- last Attnce → F

not in 3NF

Course id

Course name

Tutor id (M.K)

Tutor name (M.K)

2NF

Stno

Sname

D.O.B

Gender

Course id

Stno

last attendence

Course id

Course name

F.K → Tutor id

P.K

tutor id

3NF

tutor name

Stno

Sname

2NF D.O.B.

Gender

last attendence

Cluster:-

Cluster is a d.b. object which Contains group of tables together & also clusters shares same data Block.

→ clusters are used to improve performance of the joins.

→ clusters are created by DBA's

→ clusters are created based on a Common Column name.

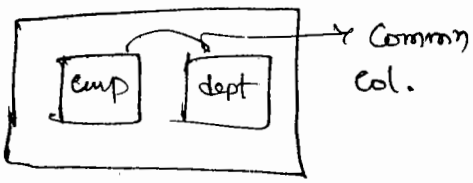
→ This Common Col also Called as "Cluster key".

→ Generally in all d.b. Systems when ever we are Submitting Inner joins or outer joins then d.b. Server checks "From" Clause tables are available in clusters or not.

→ If those tables are available in Cluster then d.b. Server very fastly retrive data from cluster tables.

→ This process automatically improves performance of the application.

Cluster



Group of 2 data

⇒ Select dname, ename, sal, d.deptno, hiredate, loc

1 From emp e, dept d

2 where e.deptno = d.deptno; → Three common cols

Creating cluster:-

26-8-14.

Steps to Creating clusters:-

Step 1) Create a cluster using clusterkey (Common Col)

Syntax: Create cluster clustername (Common Colname d.ty (size));

Step 2: Create an index on cluster:-

syntax:

```
Create index indexname on cluster clustername;
```

Syntax: Create actual tables:-

Step 3:-

```
Create table tablename (Col1 d.ty (size), Col2 d.ty (size)--- ).
    cluster
    clustername (Common colname);
```

> Create cluster

emp_dept (deptno number (10));

> Create index in1 on cluster emp_dept;

> Create table emp1 (empno number(10), ename varchar2(10), sal number(10), deptno number(10)) cluster emp_dept(deptno));

> desc emp1;

> Create table dept1 (deptno number(10), dname varchar2(10), loc varchar2(10)) cluster emp_dept(deptno);

> desc dept1;

> Insert into emp1 values (1, 'abc', 2000, 10);

> Insert into dept1 values (10, 'siva', 'ameerpet');

> Select * from emp1;

> Select * from dept1;

> select rowid from emp1;

> Select rowid from dept1;

*** in clusters Same rowid's only.

Note:- Generally two rowid's are not same in a table But cluster tables having same rowid's.

Note:- Generally we cannot drop cluster if cluster having tables.

→ To over come this problem oracle 8i introduced "including tables" clause to drop clusters along with tables.

Syntax:-

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

Ex:

```
drop cluster emp_dept ;
```

Error:- cluster not empty;

8i onwards.

Ex:- `drop cluster emp_dept including tables;`

→ All clusters information stored under "user_clusters" d.dictionary.

Ex: `desc user_clusters;`

Super key, Candidate key:-

Super key:- A column or combination of columns which uniquely identifying a record is called "super key".

↓ in a table

Candidate Key:-

A minimal Super key which uniqely identifying a record in a table is called "Candidate Key".

Note:- A Super key which Subset of (combination) another S.key is not a Candidate key.

empno	ename	Skillid	SKILL	Voteid.
1	murali	1	Oracle	V_1
1	murali	2	Sybase	V_1
1	murali	3	Ingres	V_1
2	abc	4	DB_2	V_2
2	abc	1	oracle	V_2
3	xyz	5	Informix	V_3
4	zzz			V_4
5	aaa	6	Sqlservers	V_5
5	aaa	4	DB_2	V_5

Superkey

1. Empno + Skill.
2. Empno + ename + Skill.
3. Empno + Skill + Voteid.
4. ename + skill
5. ename + skill + voteid
6. voteid + skill
7. empno + ename + voteid + skill

not a S.key

1. empno + ename.
2. Empno + voteid.

Candidate Key.

(1) empno + Skills.

(2) ename + Skills.

(3) Voterid + Skills.

BCNF.

⟹ If a table is in BCNF then every determinant is a "Candidate Key."

→ If a table contains multiple Candidate keys & also those Candidate keys are overlapped and also 1 Candidate key non key attribute which depends on another candidate key nonkey attribute then only we are using "BCNF" process.

Process:-

→ Identify 1 Candidate key n.key.attribute which depends on another candikey nonkey.attribute put into seperate Table this table is called "BCNF Table"

In this table only every determinent is a Candite key.

BCNF table does not have nonkey attributes

→ Generally if a table contains only 1 Candidate key then BCNF is also same as 3NF.

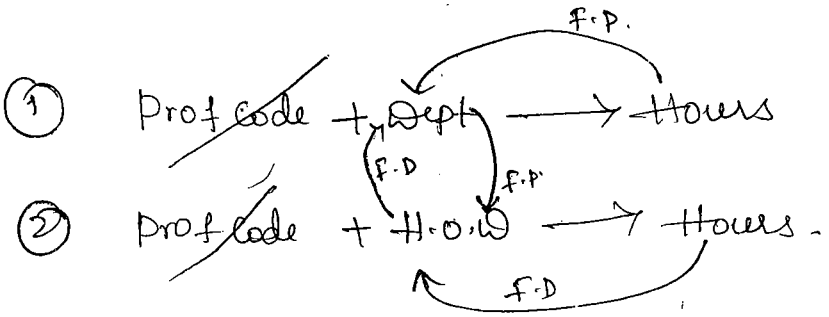
→ Generally when a table contains multiple Candidate keys then deletion problem occured.

→ To avoid deletion problm d.b. designers uses BCNF process.

→ Generally in BCNF we are specifying relation ship b/w candidate keys itself.

Prof code	Dept	H.O.D	Hours.
P_1	Physics	H1	3
P_1	Computers	H2	5
P_2	Chemistry	H3	7
P_2	Botany	H4	9

→ Non key attributes.



1 Candidate key

ple Candi.

is BCNF

relation

on key

ributes.

BCNF

Dept	H.O.D.
phy	H1
com	H2
che	H3
Bot	H4

ProfCode	Dept	Hours
P1	phy	3
P1	Com	5
P2	che	7
P2	Bot	9

(1) Dept ← H.O.D.

Determinent → (Candidate key)

(2) H.O.D → Dept

determinent → (Candidate key)

27-8-14.

4NF

4NF:-

If a table is in 4NF, then that table does not contain more than on independent multi-valued attribute

— If a table contains more than two attributes & also identifying a record uniquely using combination of all these attributes.

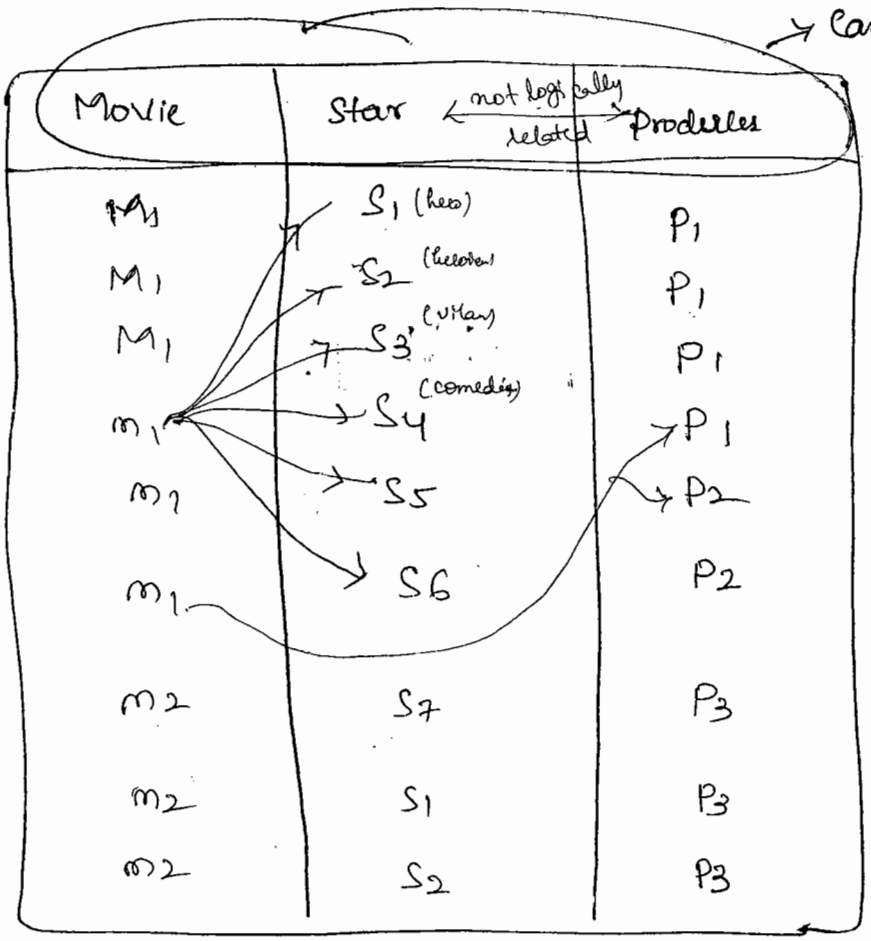
→ & also 1 attribute set of values which depends on another attribute set of values & also same attribute set of values which depends on another set of values & also some attributes are not logically related. Then only we are using 4NF

In 4th NF resolve table also there is no non-key attributes.

- Process:-

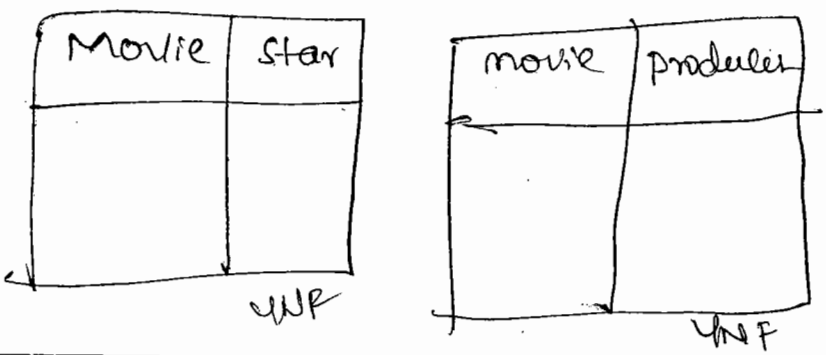
— Identify a column which depends on another independent multivalued attribute column. Put into seperate table.

— These tables are called 4NF tables



not in 4NF

↓4NF



Assumption:- Each employee having multiple projects & each employee having multiple skills.

Empid	Projectid	Skills
1111	Railway reservation	JSP
1111	Railway reservation	Asp.net
1111	Railway reservation	oracle
1111	Library management	php
1111	Library management	Sqlserver
1111	S/W	Null
1111	Null	SAP

Candidate key

if we insert null value problem will occur.

(not in 4NF.

↓ 4NF

4NF

Empid	projectid
1111	Rail·Res
1111	Libra mar
1111	S/W

no problem

4NF

Empid	Skills
1111	JSP
1111	.net
1111	oracle
1111	php
1111	SQL
1111	SAP

→ Before 4NF process whenever an employee got a new project then we need to supply null values for the skills in the above resultant table.

→ & also when ever an employee got a new skills then we need to supply null values for the project in the above resultant table

→ These probms are automatically avoided when we are using 4NF tables.

Projection Joined NF (5NF):

→ If a table is in 5NF there are no cyclic dependencies

→ when a table contains more than 2 attributes and also identifying a record uniquely using combination of all attributes & also all fields sets of values related to one another.

→ Then only we are using 5NF process.

→ This NF is also called as "projection-joined NF" bcz when ever we are joining process a table then resultant record must be available in resolve table.

Note:

→ In 4NF some field are not logically related where as in 5NF all fields are logically related

Buyer	Itemname	Company
B1	shirt	Pepe
B1	Tshirt	levis
B1	Jeans	denim

→ Candidate key

not in 5NF

Buyer table

Buyer	Itemname

Item name	company

Buyer	Company

SET OPERATORS

→ SET operators are used to retrive date from single or multiple tables.

→ These operators are also called "Vertical joins."

There are.

(1) union

(2) union all

(3) intersect

(4) minus.

(1) union:- It returns values 1 time only. (unique)

(2) union all:- unique + duplicate (all values).

(3) intersect:- Common values.

(4) minus:- It returns values in 1st query those values are not in 2nd query.

→ union:-

```
Select job from emp where deptno=10
union (unall) (in)(minus)
Select job from emp where deptno=20;
```

Note:- when ever we are using Set operators always corresponding expressions must belongs to same data type

& also Set operator returns 1st query column names as col headings.

→ we can also retrive data from set operators if Corresponding expressions not belongs to same d.type also, In this case we must use appropriate type Conversion fun's.

Ex:- Select deptno "deptno", to_char(null) "deptname" from emp union select to_number(null), dname from dept.

deptno	dname
10	
20	
30	

~~Account~~
Clerk
~~Allen~~

Conversions:- Converting 1 d.ty. into another d.ty. is called "Conversions".

28-8-14.

Oracle having 2 types of Conversions.

(1) Implicit Conversions.

(2) Explicit con ".

(1) Implicit Conversions:-

These Conversions are also called as "automatic Conversions".

In this case oracle Serves only automatically converts 1 d.ty. into another d.ty.

Note 1:-

In oracle when ever string representing pure number then oracle Serves automatically converts string type to no-type.

Ex:- Select sal + '100' from emp;

Sal + 100 ↓ convert into pure no.

(2) Note 2:- In oracle when ever we are passing nos into character fun's (length upper, lower, initcap) then oracle Server automatically converts no-type into character type.

Ex:- Select length (1111) from dual;

O/P:- 4

Note 3:- In oracle when ever we are passing date string into date fun's then oracle server automatically converts date string into date type but here passed parameter must be in default date formate

Ex:- select last_day ('17-dec-05') from dual;

O/P:- 31-DEC-05.

Implicit Conversion table.

From	To	Assignment	Evaluation of Expression.
Varchar2 (or) char	number	yes	yes
Varchar2 (or) char	Date	yes	yes
number	Varchar2 (or) char	yes	no
Date	Varchar2 (or) char	yes	no

Explicit Conversions:- In this case developers only explicitly convert 1 d.ty. into another d.ty. using explicit Conversion fun's.

(1) -
Con
Valu
a n
Ex:-

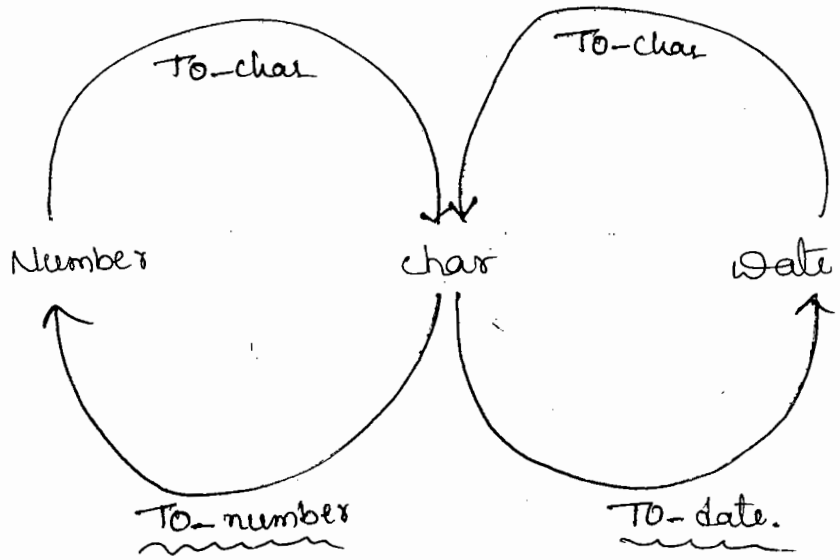
Solu

→ There are

(1) to_number()

(2) to_char()

(3) to_date().



① to_number:-

Converting a string representing ('') a numeric value (45.6) with format ($) into a string representing('') a numeric value without format (45.6)

Ex:- Select '$45.6' + 3 from dual;

error: invalid number.

Solution :-

Select to_number('$45.6')+3 from dual

o/p:- 48.6 error: invalid no.

> Select

to_number ('$45.6', '$99.9') + 3 from dual;

O.P:- 48.6.

In oracle.

① $ — $ (or) L (local currency)
② . — . (d). (only special Symbols)
③ 123 — 999
④ ab — x Invalid Format model.
⑤ , — , (or) g.
$ — L

TO_char:- to_char is an overloading fun. which is used to convert no. type into character type and also convert date type into date string.

format:

(1) g ⟶ group seperator.
(2) d ⟶ decimal indicator.

→ Select to_char(20000, '9999') from dual
1 2 3 4 5 1 2 3 4

O.P:- ####

→ Select to_char(20000, '99999') from dual

20000.

→ Select to_char(20000, '99g999g99d9') from dual

~~20,00,00.0~~ 2,00,00.0.

→ Select to_char(20000, '$99999d99') from dual

20000.00

→ Select

Sal, to_char (sal, 'L9999', 'NLS_CURRENCY=Rs')

from emp

SAL	TO-CHAR
800	Rs 800
1600	Rs 1,600
1250	Rs 1,250.

→ Select

Sal, to_char (sal, 'L 9999') from emp.

SAL	TO-CHAR.
800	$800
1600	$ 1,600.
1250	$ 1,250

L – local currency.

→ Select ename, nvl (to_char (comm), 'no commission')

O/P:

Comm
2000
no commission (null replace with)
no commission
500
200

from emp

→ Select ename, nvl (to_char(mgr), 'no manager') from emp;

→ Select to_char (sysdate, 'DD/mm/yyyy') from dual

28/08/2014.

TO_date: It is used to convert date string into date type

→ Select to_date ('16/June/05') from dual;

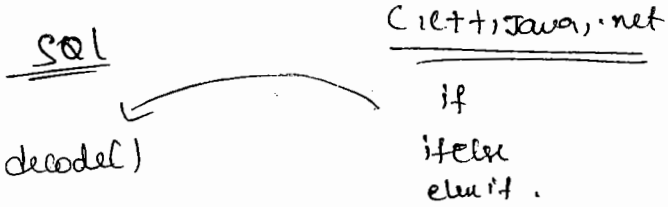
16-JUN-05.

decode() :- (Conversion fun) :-

decode() is a Conversion fun which is used to Convert 1 d.ty. into another d.ty.

→ It is used to Convert no.type into String type & also used Converts Single character into string.

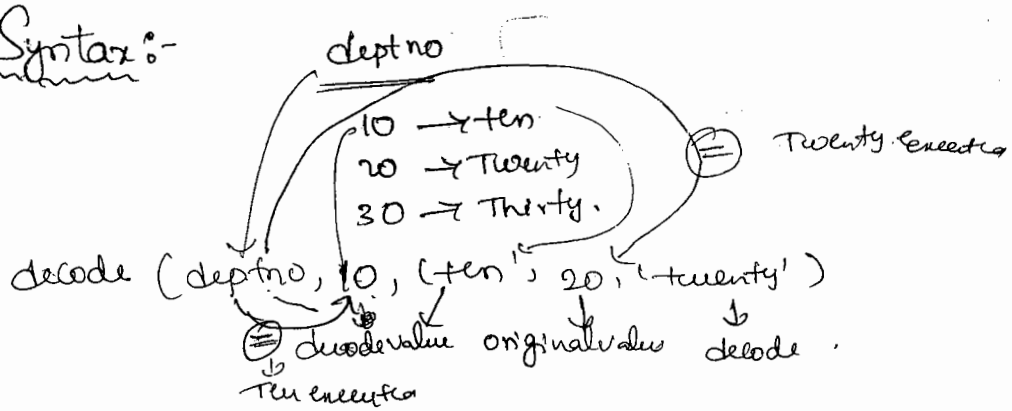
'a' → 'ab' 123 → '123'.



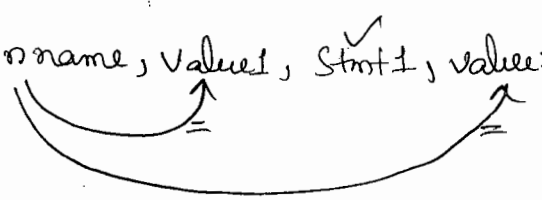
It is also same as if -- then -- elsif Construct in PL/SQL.

→ decode() internally uses equality operator (=)

Syntax :-



decode (Column name, Value1, Strn1, value2, Strn2, ... Strnn)



> Select ename, sal, deptno, decode (deptno, 10, 'ten', 20, 'twenty', 'others') from emp

29-8-14.

Note:- We can also use decode() for modifying data in a table conditionally. In this case, we are using decode() in update stmt.

(Q) WAQ to modify commission of the employees from emp table based on following conditions.

(1) if job = 'CLERK' then Comm → 10% of Sal.

(2) if job = 'SALESMAN' then Comm → 20% of Sal.

(3) if job = 'ANALYST' then Comm → 30% of Salary.

(4) else Comm → 50% of Salary.

> update emp set Comm = decode (job, 'CLERK', Sal*0.1, 'SALESMAN', Sal*0.2, 'ANALYST', Sal*0.3, Sal*0.5);

NOTE:- We can also use decode() within group by clause to display summarize data conditionally.

For eg:- > select job, sum (decode (deptno, 10, sal)) "deptno10", sum (decode (deptno, 20, sal)) "deptno20", sum (decode (deptno, 30, sal)) "deptno30" from emp group by job;

O/P:-

JOB	deptno 10	deptno 20	deptno 30
Analyst		6000	
clerk	1300	1900	950
manager	2450	2975	2850
President	5000		

whenever we want to display multiple values for single columns, then we use decode() : decode() is also used for counting purpose. for eg Counting no. of clerks in sales department.

Sum(decode(job, 'SALESMAN', 1)) → Here, for every one Salesman, it count 1 and sum means add 1 to previous value.

Eg:- select dname, sum(decode(job, 'CLERK', 1, 0)) "Clerk",
Sum(decode(job, 'SALESMAN', 1, 0)) "Salesmans",
Sum(decode(job, 'ANALYST', 1, 0)) "Analyst",
where e.deptno = d.deptno group by dname;

O.P:-

DNAME	Clerks	Salesman	Analyst
Accounting	1	0	0
Research	2	0	2
Sales	1	4	0

{Now days, decode is replaced by Case i.e, Oracle 8.0 onwards}

CASE Statement:-

→ Oracle 8.0 introduced Case stmt which is used to decoding the value.

→ oracle 8i introduced Case Conditional stmt.

→ Case stmt performance is very high Compared to decode()

decode() internally uses '=' operator implicitly.
Case Explicitly uses all SQL relational operator from 8i which is advtages

Note:- decode() internally uses equality (=) operator where as in Case stmt we can also use all SQL operators explicitly.

Method 1 (8.0):- Syntax:-

```
Case Column name
when value1 then stmt1
when value2 then stmt2
- - - - - - - - - - - -
else stmt end
```

Ex:- Select ename, sal, deptno
Case deptno
When 10 then 'ten'
when 20 then 'twenty'
else 'others' end
from emp;

Method(2):- Case Conditional stmt (oracle 8i):-

Syntax:- Case
when condition1 then stmt1
when con2 then stmt2
- - - - - - - - - - - -
else stmts end

Ex:- Select ename, sal, Case when sal < 1000 then 'low salary' when sal btw 1000 and 2000 then 'medium sal' when sal in(2300, 2500, 3800) then 'special sal' else 'other salary' end
from emp;

Oracle Server - architecture 80-8-14.

Oracle Server mainly Consists of 2 parts.

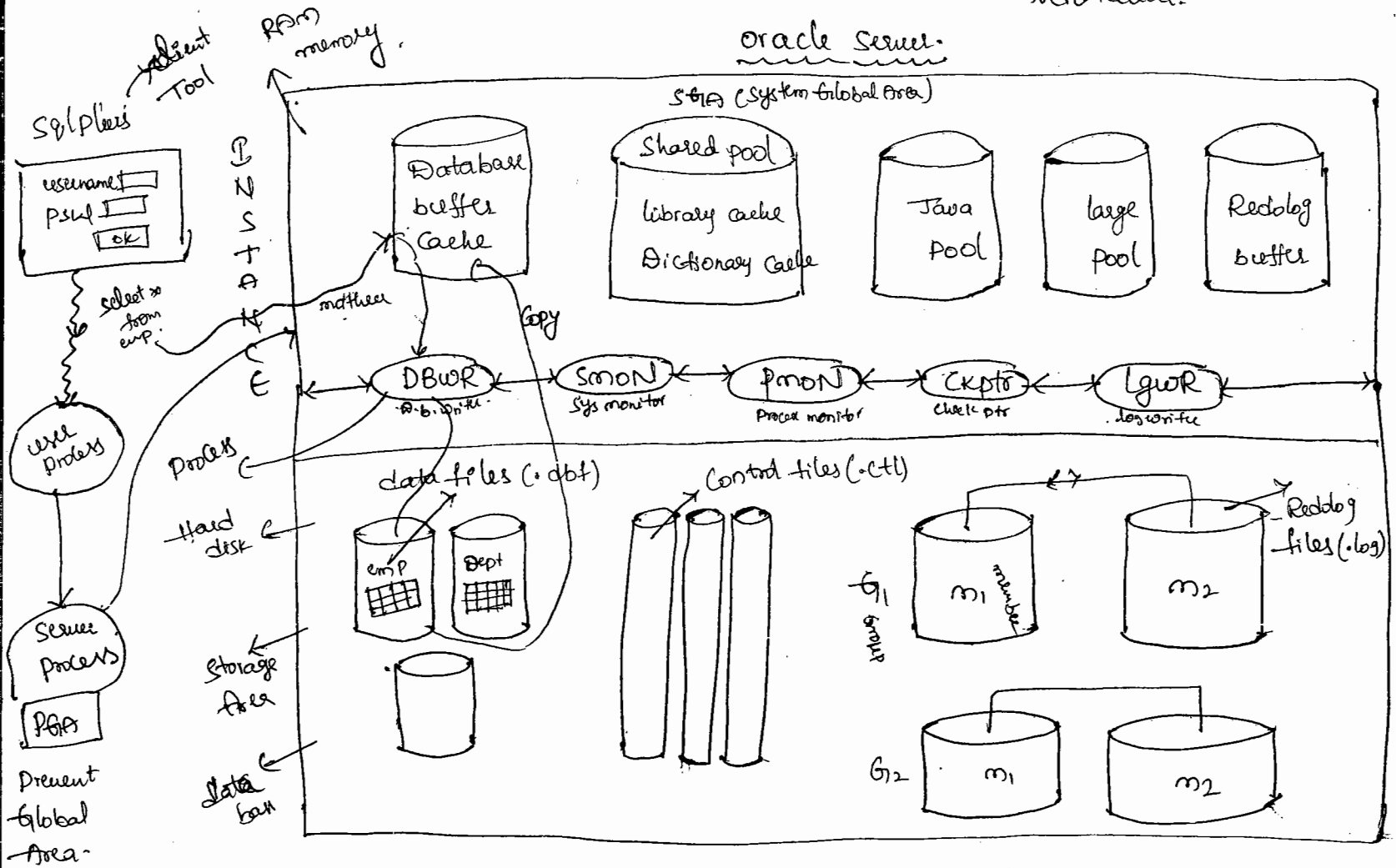
(1) Storage Area.
(2) Instance.

(1) Storage Area:- when ever we are installing Oracle Server then automatically 3 files are created in disk. These are

(1) data files. (.dbf)
(2) Control files (.ctl).
(3) Redolog files (.log).

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

Oracle Server Architecture.



→ Data files stores all database objects physically

→ Data files Extension is .dbf

→ In oracle if u want to view path of the data files then we are using "dba_data_files" data dictionary

Ex:- > Conne sys as sysdba;

Entre psw: sys.

> desc dba_data_files;

> Select FILE_NAME from dba_data_files;

→ Control files stores databases information these files extension is .ctl. these files are used by DBA's in recovery process.

→ All Control files information stored under "V$controlfile" data dictionary.

Ex: > Conn sys as sysdba;

Entre psw : sys.

> desc V$ controlfile;

> Select NAME from "V$ controlfile";

→ Redolog files stores Committed (saved) transactions from Redolog buffer.

→ These files also used by DBA in recovery process

→ All Non log files information stored under "V$ logfile" data dictionary

Ex: > desc V$logfile;

> Select MEMBER from V$logfile;

Instance:-

when ever we are connecting to the Server using SQL plus tool then automatically an Instance is created in "RAM" memory area.

This instance having 2 parts.

(1) SGA

(2) Back ground process.

(1) SGA:- (Shared / System Global Area)

SGA is also called as shared (or) system SGA.

→ SGA Contains set of buffers. These are database buffer cache, Shared pool, Java Pool, large pool, Redolog buffer.

→ when ever we are submitting SQL, PL/SQL Code that code is stored in library cache within shared pool.

→ Library Cache always checks Syntaxes Symantics

→ library cache reduces parsing by using parsing algorithms.

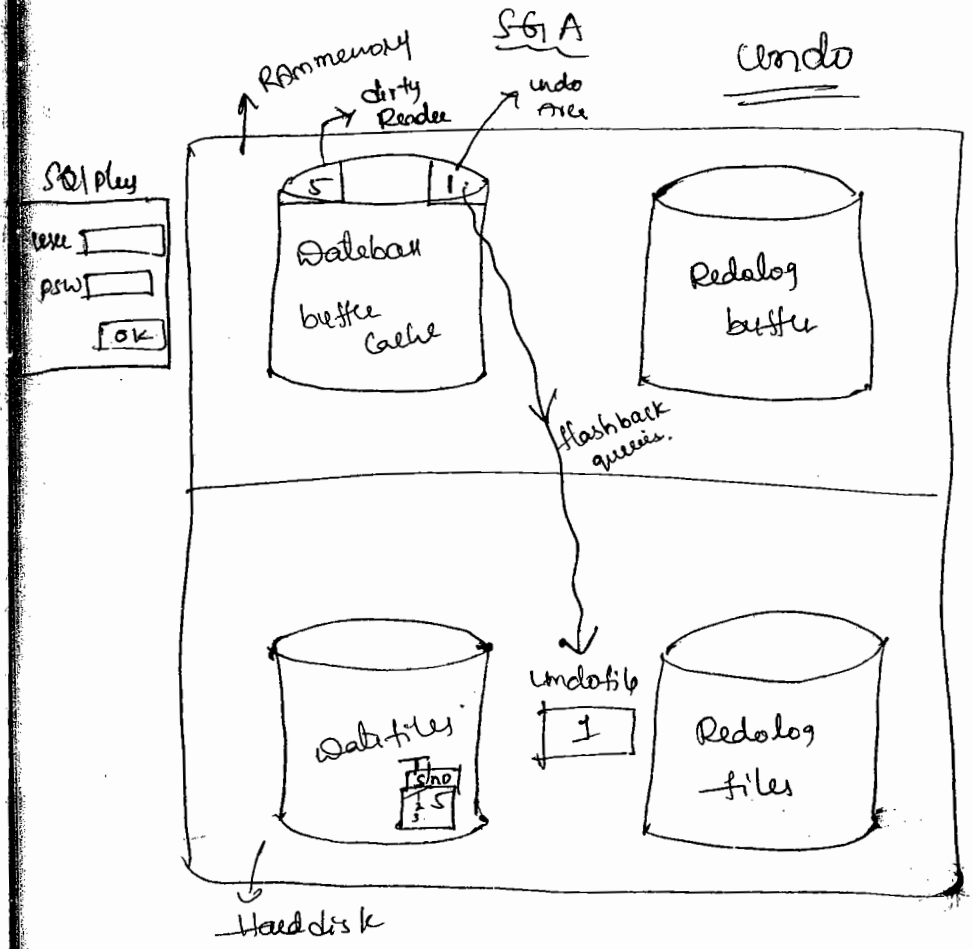
→ library cache also stores Cache values defined in sequence Object.

→ Shared pool also contains dictionary cache, which executes DCL related objects.

→ Java pool executes java related objects large pool used by the DBA. in recovery process.

→ Redolog buffer stores new information for the transaction.

→ Server process also contains PGA which identify no. of clients uniquely & also PGA stores all Collections informations from the PL/SQL.



> Update T1 set Sno=5 where Sno=1;

> Commit;

T1

Sno
1
2
3

→ In oracle when ever we all performing DML transa
tions new values for the transaction automatically
stores in dirty reader & old values for the transactio
stored in undoarea. within d.b. buffer cache.

→ when ever user gevins Commit dirty reader values
automatically parmenantly stored in data-files and also
undo area data automatically stored in undo-file
with in database.

⟹ oracle 9i introduced flashback queries to retrive
old data after Commiting the transaction also

→ These flashback queries are works based on undofile

Indexes:-

Indexes (usu_ind_columns)

→ Index is a d.b. object which is used to retrive the data very fastly from the d.b. This process automatically improves performance of the application.

→ Generally indexes are created on table Columns

→ These indexes are created by d.b.A's.

→ In all D.B. Systems we are creating indexes in 2 ways (1) Automatically

(2) manually.

→ In every D.B. Systems whenever we are requesting data using where clause (or) order by clause then only D.B. Server Searching for indexes.

Note :- In oracle whenever where clause Contains not not equal to (<>), is null, is not null operators then oracle server does not Search for indexes if those Columns already having indexes also. (group by clause) 2-8-14.

(1) Automatically :- when ever we are creating P.key (or) Unique key Constraints then oracle Server Internally automatically Creates "btree indexes" on those cols

(2) Manually :- We can also create indexes explicitly by using create index previlege (permission)

Syntax:-

```
Create index indexname on tablename(columnname);
```

→ Oracle having 2 types of indexes

(1) btree indexes.

(2) bitmap indexes.

→ generally when ever high Cardinality cols are there then only we are using "btree indexes".

where as in low cardinality cols only we are creating "bitmap indexes".

Formulae:-

$$\text{Cardinality} = \frac{\text{distinct no. of records}}{\text{total no. of records}}$$

Ex: find empno. Cardinality.

→ Select distinct(empno) from emp;

o/p 14.

empno Cardinality = $\frac{14}{14} = 1$ (high cardinality)(btree)

(2) job cardinality.

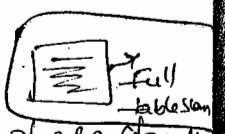
→ Select distinct(job) from emp;

— 5.

job Cardinality = $\frac{5}{14} = 0.35 \rightarrow$ (low Cardinality)

bitmap index.

btree Indexes:- In oracle by default indexes are "btree Indexes". (unique, non unique) (default when P.K & U.key → explicitly using index) (unique index, non unique indexes)



→ when ever we are requesting data using where clause or order by clause then oracle server searching for indexes on those columns in appropriate data dictionaries.

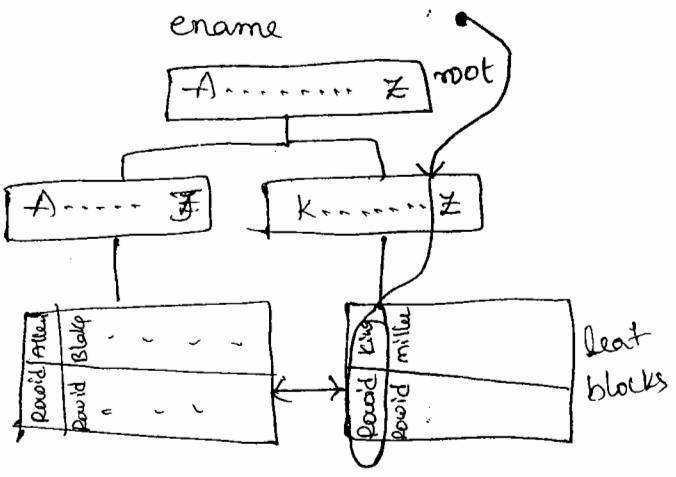
→ If btree index is available then oracle server automatically creates btree structure on those columns

→ In btree structure always "leaf blocks" stores actual data along with rowid's. Based on the Indexes scan mechanism oracle server search data very fastly in btree structure & also retrive data very fastly from the leaf blocks.

→ This process automatically improves performance of the application.

Developer.

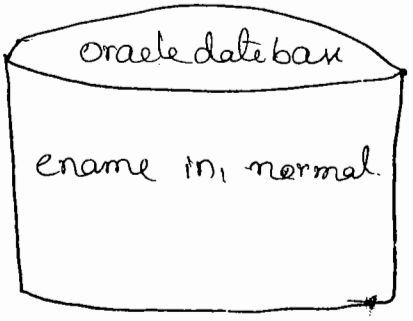
> Select * from emp (ename created as index & btree formed)
where ename='KING';



(Indexes scan mechanism)

DBA. (manual creation)

Create index in1
on emp(ename);



btree K.W → normal.

Calculate performance of the Query:-

Step 1:- Syntax:-

> explain plan for select * from tablename where condition;

Step 2:-

After planning select query then we must use display method from "dbms_xplan" package using following

Syntax.

> select * from table (dbms_xplan.display());

Ex:- without using indexes:-

> explain plan for
select * from emp where ename = 'KING';

> Select * from
table (dbms_xplan.display());

O.P:- Cost (%CPU)
3.

Using indexes:-

> Create index in* on emp(ename);

> explain plan for select * from emp where ename = 'KING';

> Select * from table (dbms_xplan.display());

O.P:- Cost
1 (High performance)

→ In oracle all Indexes information stored under "user_indexes" d.dictionary

> desc user_indexes;

> Select INDEX_NAME, INDEX_TYPE from user_indexes where TABLE_NAME='EMP';

INDEX_NAME	INDEX_TYPE
PK_EMP (auto)	NORMAL
IN1 (manu)	NORMAL.

(automatically when P.K created then this index created)

(manually we all created index)

User_IND_Columns:-

→ In oracle if u want to view column names along with index names then we all using "User_IND_COLUMNS" data dictionary.

Ex:- > desc user_IND_COLUMNS;

> Select INDEX_NAME, COLUMN_NAME from user_ind_columns where TABLE_NAME='Emp';

O.P:-

INDEX_NAME	COLUMN_NAME
PK_EMP	EMPNO.
IN1	ENAME.

Function Based indexes:- (by default it is normal index) → btree

Oracle 8i introduced F.B.I's.

→ Generally in oracle when ever where clause contains fun's (or) expressions then oracle server does not search for indexes. if those columns already having indexes also.

→ To overcome this problem oracle 8i introduced extension of the btree indexes called funbased indexes which creates indexes on fun's along with columns

→ Syntax:-

```
Create index index name on tablename (functionname (Colname));
                                or
                            expression
```

Ex:-

```
Create index inx on
        emp( upper (ename));
```

> Select * from emp where upper (ename) = 'KING';

> Select index_name, index_type from
user_indexes where
table_name = 'Emp';

INDEX_NAME	INDEX_TYPE
IN2	Function_Based Normal

Virtual Columns:- 3-8-14.

Oracle 11g introduce Virtual Columns, which stores stored expressions directly in database.

→ generally before oracle 11g we can also stored stored expressions in oracle d.b. indirectly through views, fun based indexes.

→ oracle 11g having Virtual Columns which stores stored expression through "generated always as clause"

Syntax :-

```
Columnname datatype(size) generated always as
(function_name(Colname)) [virtual].
   (or)
 expression
```

> Create table t1 (a number(10), b number(10),
c number(10) generated always as (a+b) virtual);

(a+b) → stored expression (previous col values)

> desc t1;

> insert into t1(a,b) values (10,20);

> Select * from t1;

a	b	c
10	20	30

Note:- If we want to view Virtual Column expressions then we are using "data_default" property from "user_tab_columns" data dictionary.

Ex:- > desc user_tab_columns.

> Select column_name, data_default from user_tab_columns where table_name = 'T1';

COLUMN_NAME	DATA_DEFAULT
C	"A+B".

Bitmap Indexes:-

Oracle 7.3 introduced Bitmap Indexes

→ Generally Bitmap indexes are created on "low cardinality" columns

→ Bitmap indexes are created on low cardinality cols in "datahousing applications"

→ Generally we are not allow to create unique btree indexes on "duplicate value columns" (Job)

→ In place of this one we can also create bitmap indexes.

Syntax:-

```
create bitmap index indexname on tablename(columnsname);
```

Create unique index in1 on emp(ename);

Index created

Create unique index in2 on emp(job);

error: Can not create unique syntax Index: duplicate value colum.

so

Solving:-

> Create bitmap index in2 on emp(Job);

→ Index Created.

⟹ whenever we are creating bitmap index then oracle Server internally automatically Creates a "bitmap table"

→ whenever user requesting data using logical operators then oracle Server directly operates on the bits within bitmap table.

→ Then resultant bitmap automatically Converted into rowid using an internal bitmap fun.

> Create bitmap index in2 on emp (Job).

Job	1	2	3	4	5	6	7	8	9	10	11	12	13	14
clerk	1	0	0	0	0	0	0	0	0	0	1	1	0	1
Salesman	0	1	1	0	1	0	0	0	0	1	0	0	0	0
manager	0	0	0	1	0	1	1	0	0	0	0	0	0	0
Analyst	0	0	0	0	0	0	0	1	0	0	0	0	1	0
President	0	0	0	0	0	0	0	0	1	0	0	0	0	0

We can also drop Index using "drop index index name"

Synonyms :-

(reference name) & views (Security purpose) (by default private) droping Synonym ~~table~~ also does

→ Synonym is a d.b. object

→ which hides another schema, username, object name, d.b. link

→ Synonym is an alias name or reference name for the table, view, procedure, function etc

→ Synonyms are created by d.b. administrator

→ All d.b. systems having 2 types of Synonyms

(1) public Synonyms.

(2) Private Synonyms.

→ In oracle d.b. by default Synonyms are "Private"

→ If u want to create public Synonyms then we must give "Create public Synonym priveleg" to user

Syntax:-

Grant Create public Synonym to username;

Private Synonym Syntax:-

→ Create Synonym Synonymname for username.object name@database link;

Public Synonym:-

```
Create public Synonym Synonymname
for username.objectname @ database link;
```

Scott/Tiger	murali/murali	a1/a1
> Grant all on emp to murali a1;	> Select * from scott.emp; (✓)	> Select * from scott.emp; (✓)
	> Create synonym abc (s.name) for scott.emp;	> Select * from abc;
	> Select * from abc; (✓)	Error:- Table or view does not exist
	> Create public synonym xyz for scott.emp;	
	Error:- Insufficient Privileges	
	> Conn sys as sysdba; ↵ Enter psw: sys	> Select * from xyz; (✓)
	> Grant Create public synonym to murali;	
	> Conn murali/murali;	
	> Create public Synonym xyz for scott.emp;	
	> Select * from xyz; (✓)	

4-9-14.

→ All synonym information stored under user_synonyms data dictionary.

> desc user_synonyms;

TCL:- (Transaction Control Language):-

Transaction:- A logical unit of work in between 2 points is called "transaction".

→ All d.b. systems having 2 transactional commands

(1) Commit.

(2) Savepoint.

(1) Commit:- This command is used to save the transaction permanently into d.b. (All DDL cmds automatically Committed)

Syntax:-

```
Commit;
```

(2) Savepoint:- It is a logical mark b/w transactions

Syntax:-

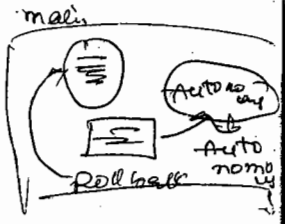
```
Savepoint  savepointname;
```

rollback:- This command is used to undo the transactions from memory.

Syntax:-

```
rollback;
```

Rollback to Savepoint (Particular) :-



Syntax:- `rollback to Savepointname ;`

Ex:- delete from emp where empno=1;

> update emp set sal = sal - 200 where ename = 'SMITH';

> Savepoint S1;

> update emp set sal = sal - 100 where ename = 'SMITH';

> Insert into emp (empno) Values (5);

> Savepoint S2;

> update emp set sal = sal + 100 where ename = 'SMITH';

> Rollback to S1; (S1, S2 Rollback) above S1 are commited.

> Commit;

— Commit Completed.

Nestead Table :- (Create or replace type) (user defined dty) (Index by Table not like in db. overcome problem extendents is NT):-

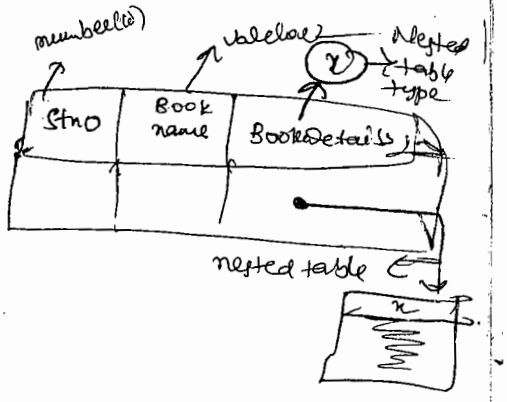
→ Oracle 8.0 introduced nestead table basically nestead table is a user defined type which is used to store no. of data items in a 'single unit'.

→ generally if u want to store large amount of data into single unit then we are using index by table in PL/SQL.

→ But we are not allow to store index by tables permanently into database.

→ TO overcome this problem oracle 8.0 introduced extension of the index by tables called nested tables which stores

Permenantly into d.b. using SQL..



Creating Nestead Tables:-

Step 1:- Create an object type.

Step 2:- Create nestead table type

Step 3:- Create actual table.

Create a object type:-

This is a user defined type which is used to represent different data types into single unit.

It is also same as structures in 'c' language.

Syntax:-

```
Create or Replace Type Typename
as object (attributename1 datatype(size), attributename2 dty(size)...);
```

Create as nestead table type:-

Sy

```
Create or replace type typename
as table of objectname type;
```

Step 3:- Create actual table:-

Syntax:-

```
Create table tablename (col1 datatype (size),....
...., coln nestead tabletype ) nestead table colname
                                              same
    Stores as anyname
```

Ex: Create or replace type e1 as object (bookId number (10), bookname varchar2(10), price number (10));

/

> Create or replace type xy as table of e1;

/ - compulsory.

> Create table Student (Stno number(10), Sname varchar2(10), Col3 xy) nestead table Col3 stored as pqr;
Same Kind address (any name its works)

> desc student;

name	type.
Stno	Number (10)
Stname	Varchar2 (10)
Col3	xy ← typename → Coltname

Note: If u want to store data into nestead table type then we must use Constructor.

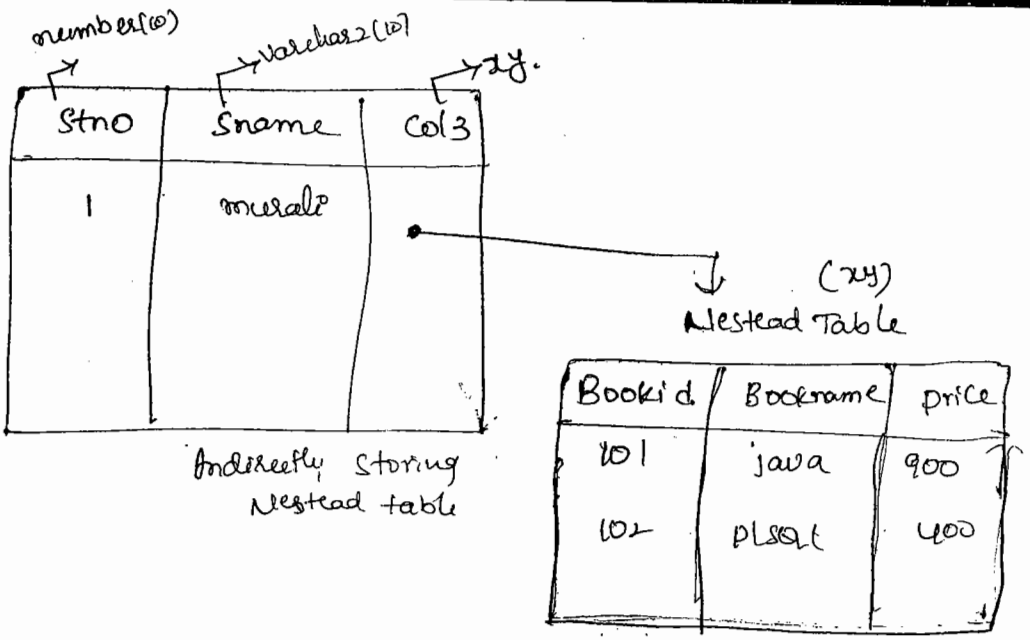
→ here constructor name is same as type name.

> Insert into student values (1, 'murali' xy (e1 (101, 'Java', 900), e1 (102, 'PLSQL', 400)));

> Select * from student;

O.P:-

Col3 (BOOKID, Bookname, price)
murali
xy (e1 (101, 'Java', 900), e1 (102, 'PLSQL', 400));
1 row created.



> Select * from table (Select Col3 from student);

o/p:-

Bookid	Bname	price
101	Java	900
102	PLSQL	400

> update table (Select Col3 from Student) Set price=1000 where BookId = 101;

> Select * from table (Select Col3 from student);

101	Java	1000
102	PLSQL	400

→ If u want to view nestead table data or modified nestead table data then we must use 'Table' operator.

→ & also along with table operator we must Select nestead table Columns within Subquery

Syntax:-

→ Select * from table (select col3 from Student);

→ Select * from table (select nestedtablecolname from actual table name);
* Operator

O/P :-

101	Java	900
102	PLSQL	400.

Partitions.

→ partition tables are created by database administrator partition tables are used in back up and recovery process which is used to improves performance of

→ Partitions tables are used in date warehousing applications.

→ generally partitions are created based on a key Colmn

→ This Col is also called as partition key

→ when database Contains large amount of date then only we are Creating partition table.

→ Oracle having 3 types of partition

(1) Range partitions.
(2) list partitions.
(3) Hash partitions.

Range partitions:- In this method database admin-strator create partitions based on range of values.

Syntax:-

```
Create table tablename(col, datatype(size),....)
partition by range(key column name)
(Partition partitionname values less than(actual value),
partition partitionname2 values less than (actual value),
- - - - - - - - -
- - - - - - - - -
partition partitionname value less than (max value)
```

retrive particulas partitions:-

Syntax:-

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

Ex:-

```
Create table test(empno number(10), ename
varchar2(10), sal number(10))
Partition by range(sal)
(Partition P1 values less than (1000),
Partition P2 values less than (2000),
Partition others values less than (max value));
```

> insert into test values (....);

> Select * from test;

> Select * from test;

Partition (p1);

EMPNO	ENAME	SAL
1	a	500.

Note: We can also add partition by using alter with add into the existing partition table

Syntax:

```
alter table tablename add
partition partitionname values
less than (actual value);
```

List Partitions:

Oracle 9i introduces list partition in this method data base administrator partitioning the table by using list of values.

Syntax: Create table tablename (col, datatype (size), ...)

partition by list (key columnname)

(Partition partitionname1 values (value1, value2, ...),

Partition partitionname2 values (value1, value2, ...),

- - - - - - - - - -

Particular particularname values (Default));

Ex:- Create table test1 (sno number(10), name varchar2(10))

partition by list (name)

(partition P1 values ('india', 'pakistan'),

partition P2

values ('us', 'uk', 'canada'),

partition others

values (default));

> insert into test1 values (.....);

> select * from test1 partition (others);

SNO	Name
1	Japan
2	australia

Hash Partition:- In this method oracle server only create partition based on specified column. by using an internal "Hash algorithm".

Syntax:-

Create table tablename (Col1 datatype(size), ...)

Partition by hash (key Colname)

Partitions any number;

Ex:- Create table test3 (sno number(10), sal number(10));

Partition by hash(sal)

Partition 5;

→ all partition information stored user_tab_partition data dictionary.

Ex:- SQL> `desc user_tab_partitions;`

> Select partition_name from user_tab_partitions where table_name = 'Test3';

Hierarchical queries:- In relational data bases we can also stored hierarchical data

→ In oracle if we want to retrive hierarchical data then we are using level pseudo column along with following clauses.

(1) Start with

(2) Connect by.

→ If we want to store hierarsical data in relational table minimum 2 column are requied and also these 2 column must belong to same datatype & also having some relation.

→ level is a pseudo column which allignes nos to each level in Tree Structure

→ Start with:-

This clause specify Searching condition with in hierarchy

Syntax :- Start with Condition

Connect by :- This clause is used to specify relationship b/w parent, child Column.

- In this clause we must used prior operator

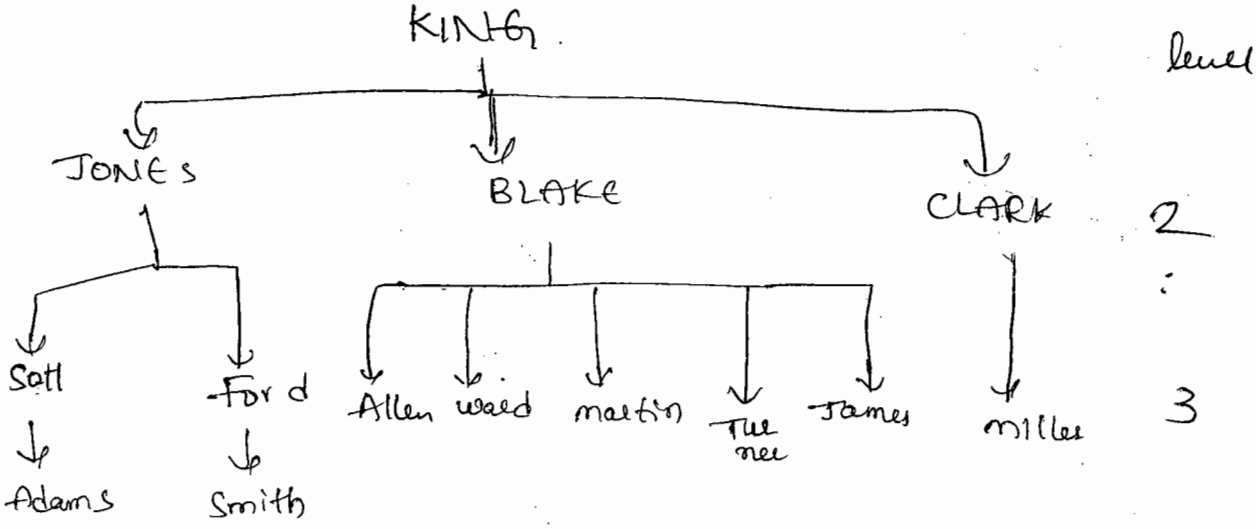
Syntax:

```
Select Col1, Col2 ......
from tablename
where condition
Start with condition
Connect by prior
parent Columnname = child Columnname;
```

> Write a hierarchical query to display employees who are working under another employees root node onwards from emp table using employee number, MGR Columns.

```
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() function which returns path of hierarchy entry structure.

This fun accepts 2 parameters

Syntax:-

sys_connect_by_path

(columnname, 'delimetername')

Ex:- Select

level, sys_connect_by_path (ename, '→')

From Emp

Start with mgr is null

Connect by prior empno = mgr

/.

level

2

:

3

-path()

entry

)

Q) write a hierarchical query to display the employee who are working under Blake from emp table using emp no, mgr columns.

```
Select level,
Sys_Connect_by_path(ename,'→')
from emp
start with ename='Blake'
Connect by empno = MGR
/
```

```
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 is an unitary operator and along with Connect by clause always prior operator returns Parent Column in hierarchy

Using prior operator we can also travel top to bottom (or) Bottom to top in the Structure.

Note:- oracle 10g introduce `Connect_By_IsLEAF` Pseudo Column which returns 1 to the leaf node and '0' to the all other nodes in the structure

Syntax:-

```
Select
    level, ename, Connect_by_IsLEAF
    from emp
    start with mgr is null
    Connect by prior empno = MGR
```

Note:- oracle 10g introduced `Connect_By_root` operation which returns root node in the hierarchy

Syntax:-

`Connect_by_Root Columnname`,

Ex:-

```
Select level, ename,
Connect_by_root ename
from emp
Start with mgr is null
Connect by prior empno = mgr
```

Interview questions — # Flashback Queries.

→ Oracle 9i introduced flashback queries using flash back queries data base administrator retrive accidental data

→ flash back query always allows content of a table can be queried with respect to a specific point of time using As of clause

→ Flash back queries works based on undo file using flash back query we can retrive previous data after committing data also

→ To retrive accidental data then we are using 2 methods

(1) using timestamp datatype

(2) using SCN (System Changed Number)

Method - 1:- using timstamp datatype:-

Syntax:

```
Select * from tablename
as of timstamp(specific point of time);
```

Ex:- > Create table t1 (sno number(10));

> insert into t1 values (.....);

> Commit;

> Select * from t1;

O/P:-

SNO
1
2
3
4

> delete from t1;

> Commit;

— Commit Complete.

> Select * from t1;

— no rows selected.

> Select * from t10 as of timestamp (systimestamp - interval '2' minute);

Sno
1
2
3
4

Method 2:- (using scn)

→ In oracle whenever we are creating transaction then automatically a unique identification no. is created in the data base.

→ This unique identification number is also called as system changed number based on this number also data base administer retrive accidental date

Syntax:-

```
Select * from tablename
as of scn systemchangenumber;
```

Ex:- > Conn sys as sysdba;

Enter password: sys.

> Create table m1 (sno number(10));

> desc v$database;

> Select current_scn from v$database;

CURRENT_SCN:-

2306347.

> Insert into m1 values(5);

> Commit;

> Select * from m1;

SNO
5.

> Select Count(*) from m1 as of scn 2306347;

Count(*)
0.