

SQL

Class notes

Telegram - @jobs_city

Telegram - @jobs_city

Introduced in 2003. ← Oracle 11g → Oracle Technology

10.6

— oracle also RDBMS/s/w files. RS = BC

— Services (for alter) all are works on Port no. 1521

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

— These called dbms s/w.

← To store data

Permanently is Secondary Storage

device (Hard disk)

Oracle d.b.
SQL (language) PLSQL. Dynamic SQL

↓
Structured Query language (Procedure language extension for sql)

→ DBMS s/w → Relation DB management system

— Data is collection of facts.

Student marks → 90 80 90 20 0.

collection of no's.

names - Shekhar

Collection strings.

— Information

SQL

↓

isql

ASCII language

— Data store:— Data store is DBMS s/w,

→

→ Every org'n deals with lot of data

Data: Collection of raw facts.

Ex:- (1) Student marks

(2) Customer names

Sequel

data

is high

Information: when we are processing data we are 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.

Q.F

(2) flat files.

Q

(3) Data base.

→ w

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

- Only

Interface

Disadvantages:-

(1) data security

with

(2) data retrieval

(5) data indexing

Programs

(3) data redundancy

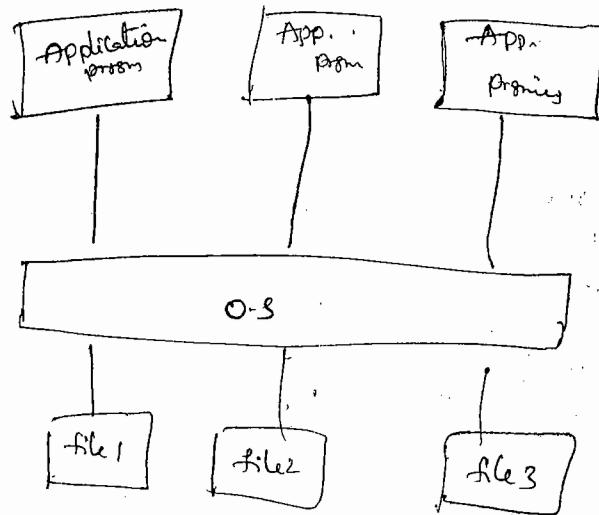
(6)

appn

(4) data integrity

⑪ Date Retrieval :-

To retrieve data from date bases we are using Sequel language, whereas as if we want to retrieve data from flat file we must develop Application program in high level languages.



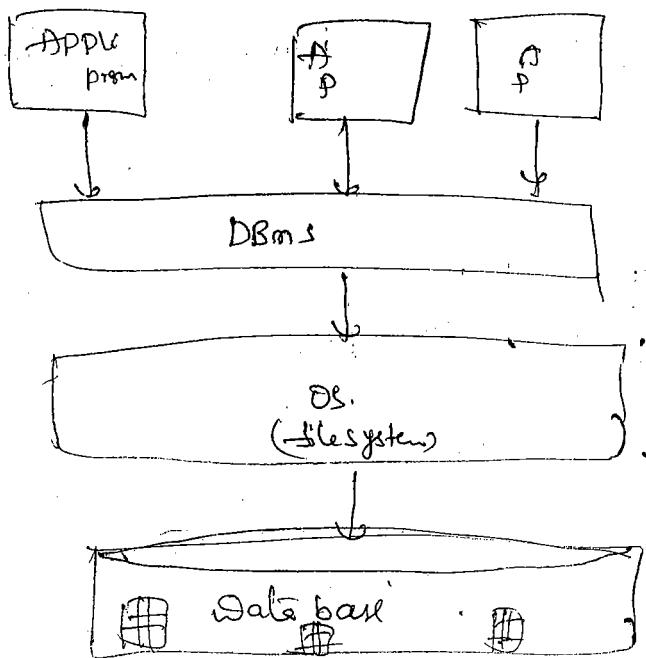
flat file approach.

② B.M.S. S/h/s - oracle, SQL.

② B - Collection of files.

→ when ever we are loading d.b.m.s sh/s automatically d.b. is created & also automatically interface is created that's why end user directly interacts with date base using interface otherwise application programs indirectly interacting with a database using appn programs.

DBMS approach



Telegram - @jobs_city

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 try to delete master table record if not possible.

Dept no to	
10	
10	
30	

To delete not possible
bcz we first delete the records in related emp (child) table record

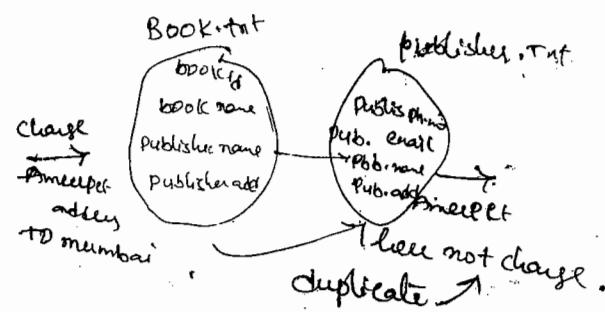
→ On - delete - cascade

when ever delete master table, all deleted in child table

Data redundancy :- (duplicate data) 11-6

Inconsistency is a biggest problem in flat file.

Consistently date
in dreams
S/10



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

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

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

→ where as D.B.m.s Shifts (scenes) automatically

maintains consistent state through transactions

En- Inconsistency ATM : Transaction 3000, but Money not given

↓
Every transaction has
4 properties

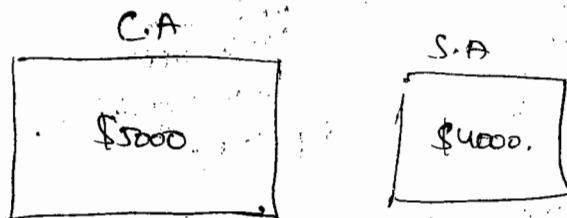
Transaction \leftrightarrow Algorithm.

before Trans + After Trans = Total data is same

Consistency property.

Atomacity \rightarrow Either all or none.

Roll back:



ACID
↓
Only consistency
data

Task: To transfer \$1000 from C.A. \rightarrow S.A.

rollback.

Step 1: Read amount C.A

2. deduct \$1000 from C.A

3. Write new amount into C.A.

4. Read S.A

5. add \$100

6. Write new amt to S.A

is this
Atomacity
broken
live atm

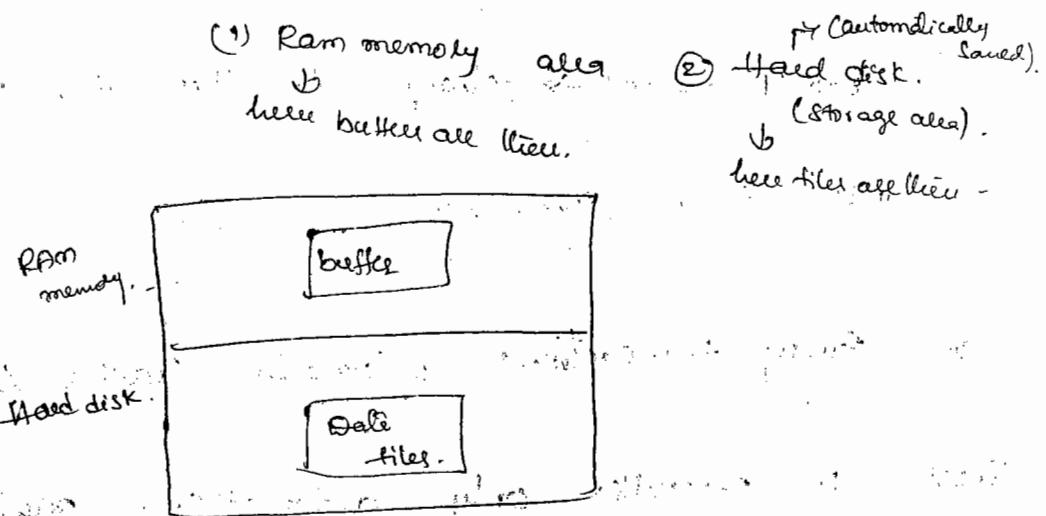
\rightarrow Permanently store all in Hard disk is \rightarrow durability.

\downarrow
(Commit)

\rightarrow Atomacity \rightarrow Rollback (cancel)

\rightarrow durability \rightarrow Commit (ok)

→ Every d.b. session having 2 parts.



→ Create '1' table. It is stored in Hard disk only with out saving also bcz ddl commands automatically saved.

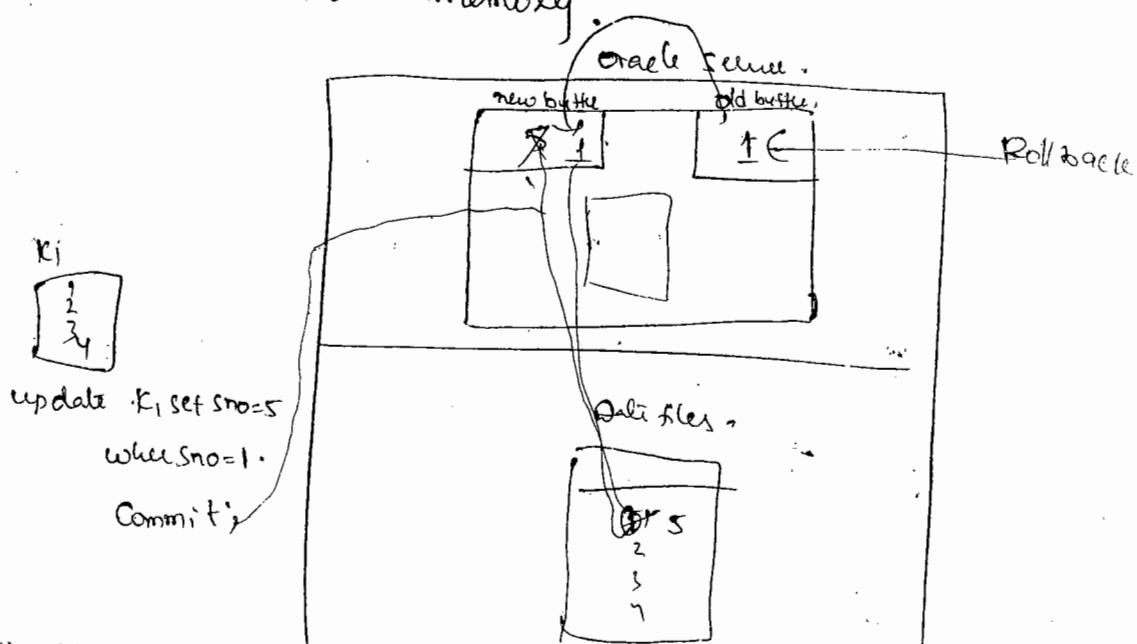
→ Commit

is used to save permanently in Hard disk.

→ Save transactions into disk.

→ Rollback (cancel the transaction),

** undo from memory.



update K1 set sno=5

where sno=1.

Commit;

→ After Commit, Roll back is not possible.

Commit means → changes in hard disk permanently.

insert, delete → These possible.

→ Every transaction internally having 4 properties

These 4 properties only automatically maintains consistent data in database. These are also called as "ACID" properties.

A → Automacity (either all or none)

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

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

D → Durability (Same like transactions permanently in disk).

→ All db. products introduced Commit Command

based on durability property, and also all db.

products introduce roll back Command based on
automacity property.

→ Reduce the redundancy we having Normalization

→ In d.b.s, if you want to reduce Redundancy

d.b. designer uses normalization process.

12-6

(3) Data Integrity (constraint)

Set of rules are there in projects

then set of rules called Business Rules

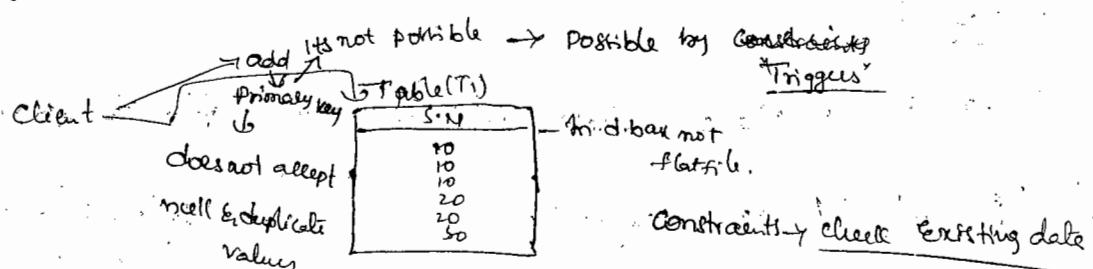
(To maintain proper data) → 2 constraints are there triggers.

→ Integrity means to maintain proper data

→ To maintain proper data developer 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 (by trigger) through Constraints,
triggers.



→ whereas in flat files we are defining business rules
using application programs (C/C++) in high level languages.

(4) Data Security: it is problem in flat files (class 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.

→ whereas d.b.s (Oracle) provides Role based Security.

data indexing (Btree Index, Bitwise indexing)

↓
very fast retrieval data

Airtel (very fast indexing)
B&NL (slow)

→ In databases to retrieve data very fastly then we are using indexing mechanism.

→ whereas flat files does not provide indexing mechanism

→ Orgns. Suffering from flat file approach to store data

→ In 1960 onwards orgns uses dbms's to store data permanently in hard disk.

(Interrelated data)

DBMS It is a

Collection of programs (cls) 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.

working

On 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

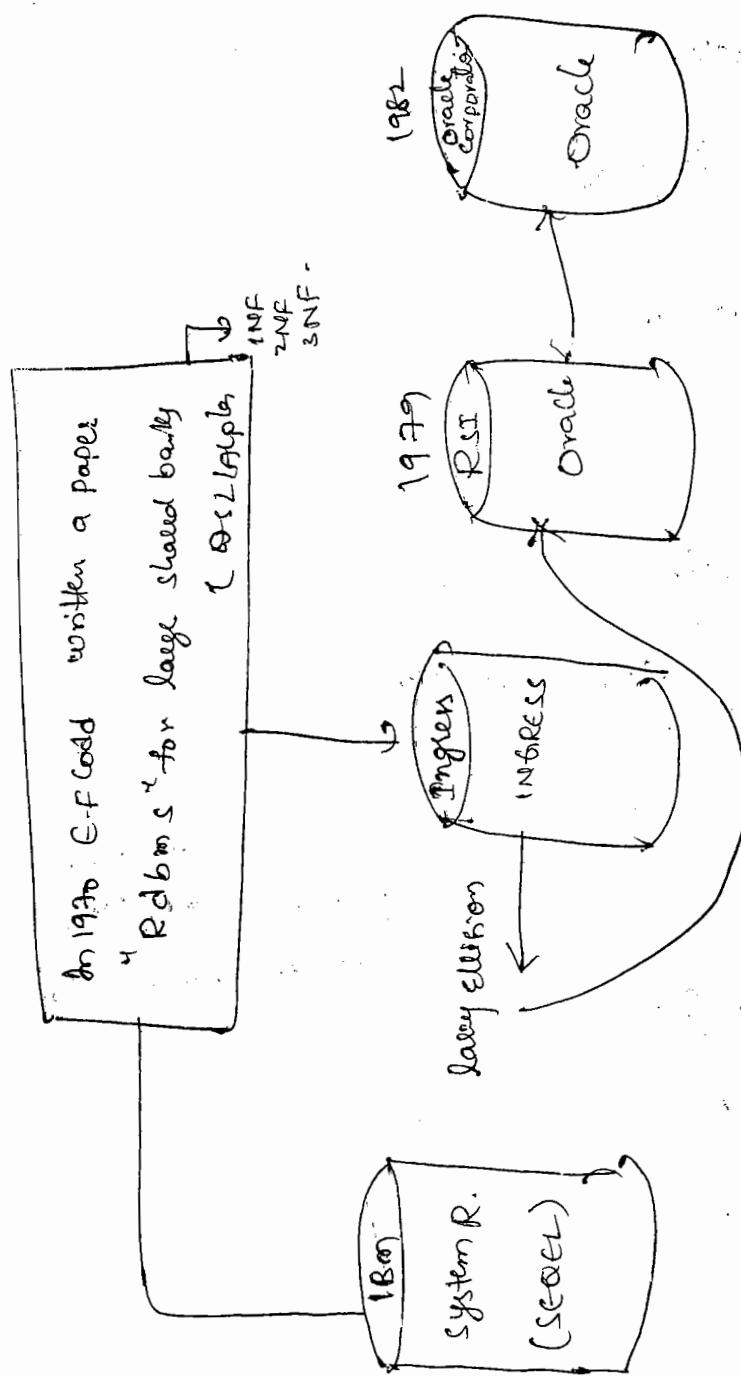
i.e. In this model we are store's data in the format

format of table.

- And also in this paper E.F. Codd introduced 1st 3rd normal forms.
- and also introduced SQL/Alpha language to operate relational databases.

Now → Oracle → SQL.

IBM [1970's] → System R → SQL/Alpha (SEQUEL)



In 1979 : Oracle Versions: (1) SQL

(2) PLSQL.

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.

there is no
up to 6.0 ↪ PL/SQL

⑥ 1986 → oracle 6.0.

PL/SQL there from 6.0.

→ introduced PL/SQL.

→ low level locks.

⑧

1992 → Oracle 7.0.

- Integrity Constraints (foreign key) introduced.
- Stored procedures, stored functions.
- Triggers are introduced.
- d.t. Varchar change into Varchar2.
- Truncate table.
- View Compilation (snapshot) introduced.

⑨

1992 → Oracle 7.1.

- Introduced dynamic SQL.
- ANSI/ISO SQL92 Standard.

⑩

Oracle 7.2.

- Inline Views.
- PL/SQL Introduced Ref Cursors.

⑪

Oracle 7.3.

- bitmap indexes.
- (readfile, writefile) (UTL_file package).

⑫

Oracle 8.0 → 1996.

- Object technology (class, C++)
- nested table.
- Varray
- Instead of triggers. Oracle 8.0

⑬ Oracle 8i (Internet) \rightarrow 1997

- rollup, cube
- columns increased per a table upto 1000.
- analytical functions

⑭ Oracle 9i \rightarrow 2000

- renaming a column
- anti joins
- multi-table inserts.
- flashback queries.
- merge statement

⑮ 9.2, 9.3.

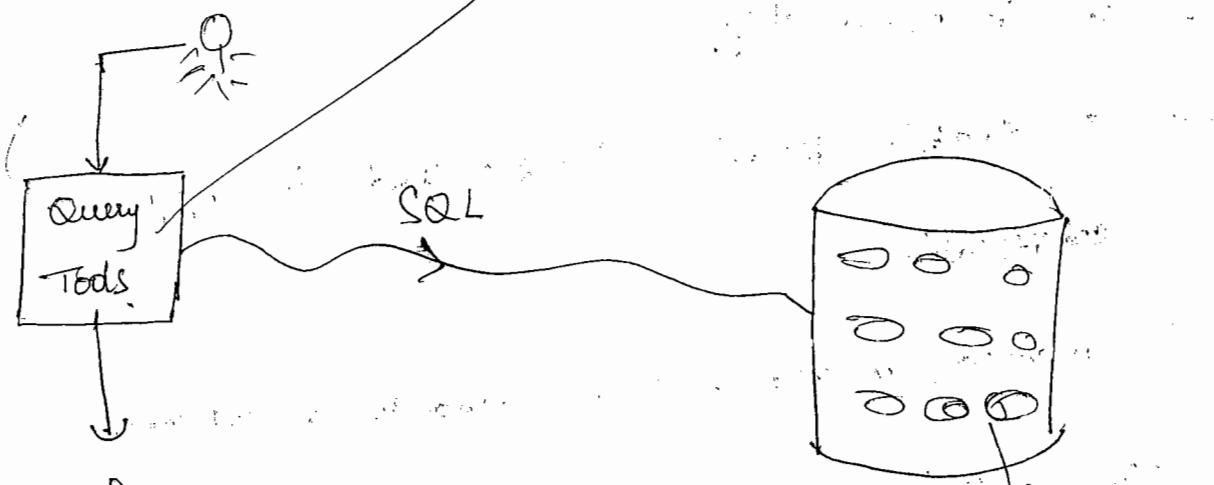
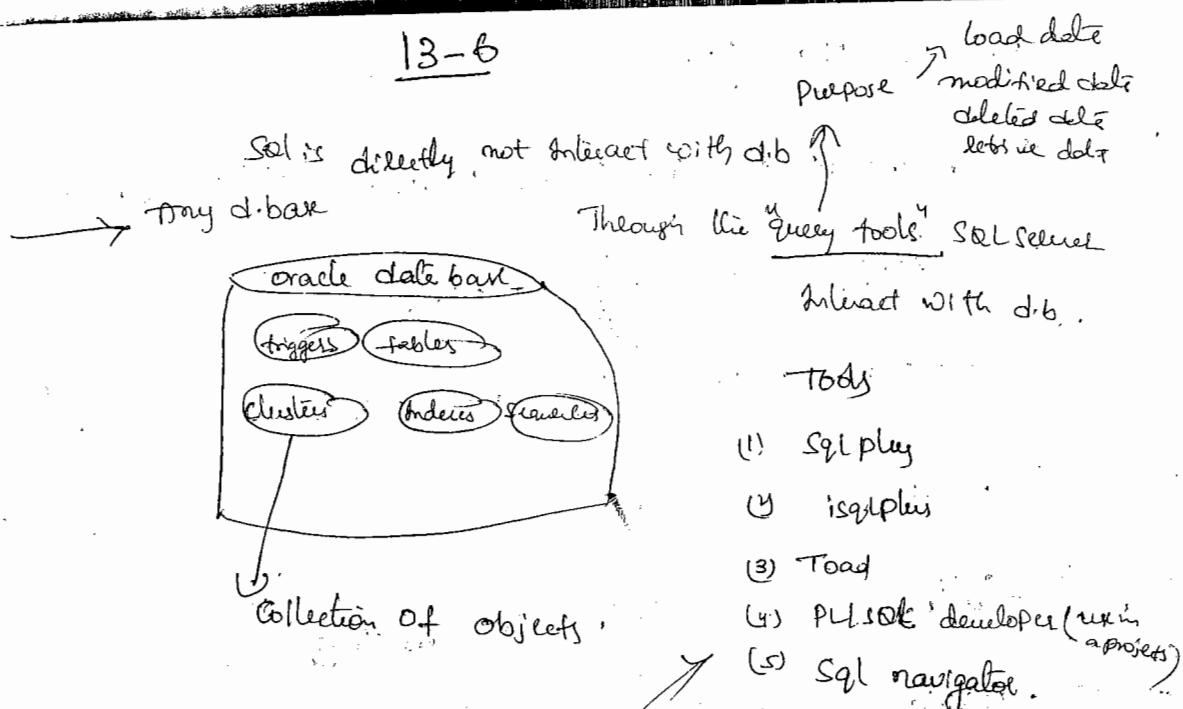
⑯ Oracle 10g (grid technology) \rightarrow 2005

- introduces administration side
- Recycle bin concept
- flashback table.
- rename tablespa
- wri_concat()
- indices of char (1/sey)
- 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 clauses introduced in triggers.
- Using sequences without using dual table in PL/SQL.
- Compound triggers.
- enable, disable clauses used in trigger specification.
- named, mixed notations used in a subprogram executed using select stmt.
- regexp_count().

13-6



Purpose -

- (1) load date
- (2) modify date
- (3) delete date
- (4) retrieve date

→ don't load diff. stuff

oracle
— Express Edition.
— Standard u.

good db. is
Enterprise Edition S/w
→ log, log, log & p. edition's

→ Each version having 3 many editions

date
double
del
dat

→ start prompts → any Oracle SQL we know the "SQLplus Tel by default" application
↓
SQL

→ Express Edition we use user name: System-developer.

but here no tables.

P.W.: Edition.

→ So we use Enterprise Edition to work in production

previous days.

→ 10g, 11g → SCOTT, TIGER → not work in Enterprise Edition
but ^{first time} 10g onwards security mechanism is there. (locking mechanism)

Environment (real time)

U.N.: [sys AS SYSDBA] → unlock user.

Oracle log.

U.N.: [scott]

P.W.: [Tiger]

Error: Account locked.

unlock user → U.N.: [sys AS SYSDBA]

automatically asked like password.

Enter password: [sys]

SQL) Alter user scott account unlock;

SQL) Conn scott /Tiger

connecting (username)
pwd.

Pwd: Tiger

Confirm Pwd: Tiger

→ Clear Screen:-

Shift + delete or cl spc scrs;

(2)

→ To view all tables

→ Select * from tab;

→ Predefined tables we must
and 4 tables → R-tables
DB-tables.

(3)

(1) Dept ^{m.t}
(2) Emp ^{1:n}
(3) Bonus
(4) Salgrade.

most powerful tables, then 2 are
normalized table.

(4)

Dept → master table. Primarily by

Emp → child table. ↓
master key.

↓
foreign key
child table

→ To view particular table;

(5)

→ Select * from Dept;

→ SQL is divided into sublanguages

(1) DDL (Data Definition language): - (All DDL Commands are

(1) Create automatically saved)

(2) Alter

(3) Drop

(4) Truncate

Common.

↓
In universal standard SQL

have comment (row) the
table

(5) Rename (oracle 9i)

② DML (Data manipulation language):-

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

(1) Insert

(2) update } Common commands

(3) delete

③ Data retrieval language / Data query language :- (SQL)

(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

GRANT

REVOKE

dsale

)

1. SQL

2. DML

SQL

PL/SQL

DML, DML

TCL

TCL, DML

SQL

Data types

PL/SQL	^{Variable} a number(10)	l
SQL	Column number(10)	int a; SQL:-

→ **D-types** identifies type of data in a column

Data type:-

- ① number (p, s)
 - ② char
 - ↳ Varchar2 (maxsize,)
 - ③ Date

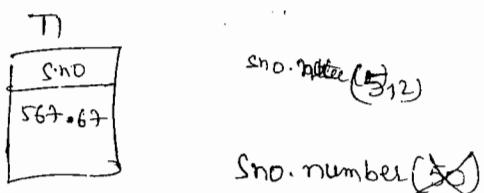
① Number (P.S) :-

Precision. $\leftarrow P \rightarrow$ Total no. of digits.

Scale: $\leftarrow S \rightarrow$ floating point no.s.

It is used to store fixed, floating pt nos.

Syntax:- Column name number(P.i.s).



max. limit of 'p' is '38',
↓
Plethora ↑ to 38.

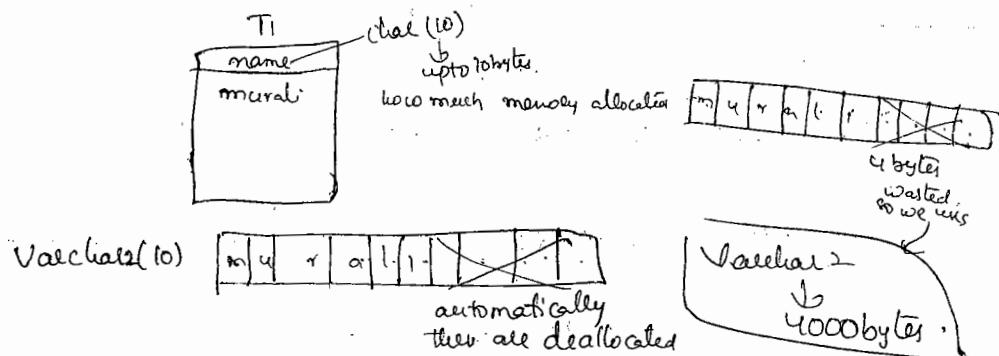
② char :- Alpha numeric data.

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

→ max. limit upto '2000' bytes.

→ By default character data leaving 1 byte.

Syntax:- Columnname char(size).



➤ Create table t1(name char(10));
↓ table create ↓ name

➤ Insert into t1 Values ('murali');
↓ row create ↓ Screen

➤ Select * from t1;

Name
murali.

→ memory storage is display by dump

➤ Select dump(name) from t1;

dump(name)

Type=96 Len=10; 20 40 30 40 50 30

waster
↑
32 32 32 32
32 ASCII of 'space'

Variable

↓ 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: Columnname Varchar2 (maxsize)

→ DATE:

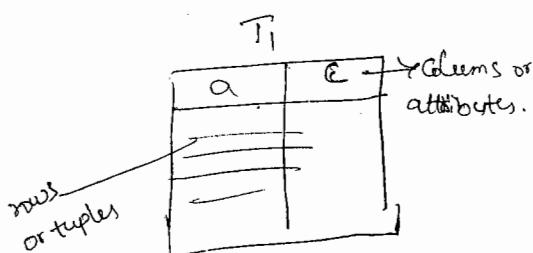
It is used to store dates in oracle date format

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

→ Select SYSDATE from Dual;

Syntax: Columnname Date
14-6-13.

→ Data Definition Language:-



These commands are used to define structure of the table.

(1) ~~Drop~~

- 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 (column1 datatype(size), column2

datatype(size), ...);

Ex:- Create table d1 (sno number (10), name varchar (10));
A table created

To View structure of the table:-

Syntax:- desc tablename;

Ex:-

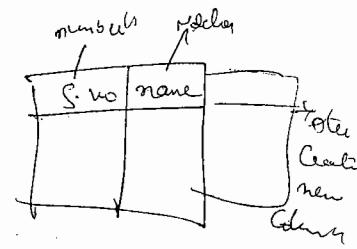
sql> desc d1;

SNO	Type
name	Number(10)
	Varchar(10)

→ Alter:- use Add or remove column.

It is used to change existing table structure.

Alter ^{use} (3 Commands: add, modify, drop.)





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

Syntax:

alter table tablename (Col1 datatype, ...);

new column already not like
add ↑ col will take

Add sal column.

Ex:

alter table d1 add sal number(10);

If Column is altered

Modify:

It is used to modify Column datatype.

Or Column datatype size only.

		date	
no.	name	datatype	value

Syntax: alter table tablename modify (Col1 datatype);

Empty table

S.no. is changed to date.

Ex:

alter table d1 modify sno varchar(10);

desc d1;

sno	varchar(10)
name	varchar(10)

de

(
(

m
n

at

mes

(
-
~

Ex

me

Column

follo

(
-
,

Ex:

drop? removing the Column.

existing

2 methods.

① using () →

② using single column → use column keyword

→ It is used to remove columns from the table
(column list)

method 1:-

If you want to drop a single column
at a time with out using parenthesis then we
must use column key in front of the column name.

Syntax:-

alter table tablename drop column columnname;
K.W.

Ex:- alter table d1 drop column sal;
(no need to write datatype)

✓

method 2:- If you want to drop a single or multiple
columns with using parenthesis 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 occurred.

drop:- drop a table, triggers, sequences, clusters, objects, etc.

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

Syntax:- (Objects)

Ex:- drop object-type objname;

— drop table tablename

— drop view viewname

— drop procedure proceduename

Ex:- drop table d1;

up to 109 ^{tables} → permanently dropped

but 109, 119 → not permanently dropped bcz here

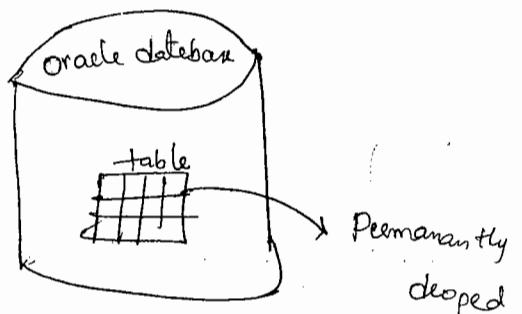
we have Ctr + Recycle bin

→ we remove permanently we use ~~trunc~~ in 109, 119 Enterprise Edition

dropping a table.

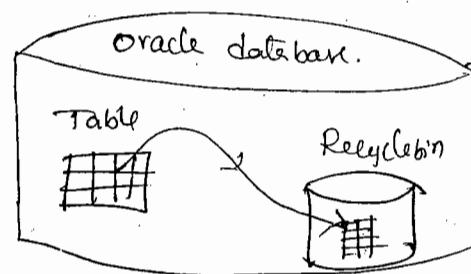
before oracle 10g.

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 permanently:-

Syntax: `drop table tablename purge;`
`or drop table d1 purge;`

→ `drop table d1;`

Table dropped.

→ get it back:-

➤ `flashback table d1 to before drop;`

➤ `desc d1;`

To drop permanently:-

➤ `drop table d1 purge;`

➤ again apply flashback comment.

➤ Error:- object not in Recycle bin.

→ Truncate— (7.0 onwards) (late command)

don't apply.

Original tables in projects. not columns

→ d.b.a sever always delete data permanently.

→ Delete command → is not freely.
→ rollback → it comes back data

→ It is used to delete data permanently

from table (not columns)

Syntax:

truncate table tablename;

Copy Table:- ^{Copy} date one table to d.table

Y Create table d1 as select * from emp;

Y Select * from d1;

Y truncate table d1;

Table truncated.

Testing:

Y Select * from d1;

no rows selected.

Y desc d1;

Only columns are displayed

Syntax

→

XY
N

C

1)

Rename :- It is used to rename a table, & renaming a column also (nowadays)

Syntax:-

(Rename Parameter 'Renamed')

rename oldtablename to newtablename ;

Ex:-

rename d1 to z1 ;

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

Renaming a Column (Oracle q1),

Column related to → Alter.

Only 1 column at a time rename is possible.

Syntax:-

alter table tablename rename Column _{IC.W.}

oldcolname to newcolname ;

→ 2 column is same is not possible in oracle.

→ alter table emp rename Column empno to sno;
Ex:-

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

XML (Data manipulation language) :-

These Comments are used to manipulate

data in a table these are ① insect

4 update

3) Delete

4) merge

4) merge (oracle 9i)

① Insert :-

It is used to insert data into table

3 methods are there.

melliod 17

Syntax: Insert into tablename Values (value1, value2...);

Eg:- Create table test (sno number (10) , name

Varchar 2 (10))

> Insert onto test values (1, 'meal1');

1 row created

→ insect into in (2, 'Saelius')

I now. Created

Y Select * from test;

Sno	Name
1	Reena
2	Sachin

leg 1-

method 2^o

Using Substitutional operator (&));

Syntax

Statement: Insert into tablename values (&col1, &col2, ...)

eg: > insert into test values (&eno, '&name');

Enter value for sno: 4

Enter value for name: xyz

Y Select * from test;

method 83-

Skipping Columns

Syntax

Syntax `Insert into tablename (Col1, Col2, ...) Values`

(Value1, Value2 ...),

egt insert into test (name) values ('bbb');

7 Select * from test;

sno	name
1	Musali
2	Salleem

Update:

It is used to modify data within a table.

Syntax:- update tablename.

Set `Column.name = newValue` where `Column = old values`

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

7 Select α from exp,

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

Eg:- \rightarrow alter table test add address Value2(10);

\rightarrow update test set address = 'mumbai' where name = 'Sahein';

\rightarrow Select * from test;

Sno	name	address
1	murali	
2	Sahein	mumbai
3	abc	

Q.:

\rightarrow

Ans

date

base

\rightarrow

total

all

the

•

Delete:- It is used to delete all rows in

Particulars rows from a table.

Syntax:- \rightarrow delete from tablename; \downarrow (to delete all rows)

Syntax:- Delete from tablename where condition;
 \downarrow (to delete particular rows)

Eg:- \rightarrow Delete from test (temporarily deleted)
 \downarrow 4 rows deleted

\rightarrow rollback; \rightarrow (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 trunc.

→ whenever we are using delete from table name
the total data deleted is 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 trunc - table - table name
total data permanently deleted because by default
all ddl commands are automatically committed
that's why we can get it back deleted data.

Data Query Language (or) data retrieval language

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

Syntax: Select Col1, Col2 from tablename where Condition group by Column name having condition Order by Columnname [asc/desc];

- ① Select all ~~columns~~^{rows} and all rows
- ② Select all columns and particular rows ~~where~~
- ③ Select ~~particular~~^{col & row} particular columns and all rows
- ④ Select particular columns in particular rows

~~Creating a new table from another table.~~

16-6

Syntax:-

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

Ex- & Create table X, as select * from emp;

Y Select * from x_1 ;

display as like acceptable.

Columns — Primary key, foreign key,
Constraint.

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

→ Only Copy Columns

→ data { ^{Set of rows} ^{where condition} will false only columns will selected }

Create a new table from another table without

Copying date:-

Syntax: Create table newtablename as Select

→ from tablename where table condition;

Ex 7) Create table X3 as Select * from emp

where $i=2;$ no rows selected \rightarrow select * from X3;

false

like emp%0000000; \rightarrow desc - X3, only columns are opted.

$i=9;$

$i=1$

$\} \{ \text{false condition.}$

$\} \{ \text{true It displays data.}$

\rightarrow when ever condition is false data will not copy.

\rightarrow Operators used in Select Statement.

\rightarrow after select some columns will be (Select col1 col2)

(1) Arithmetic operators ($*, /, +, -$) (\rightarrow instead of the word min)

(2) Comparison op's ($=, <, <=, >, >=, \{ << \text{ or } != \}$)

(3) Logical op's (AND, OR, NOT)

(4) Special op's.

where

used in where clause

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

Select display ename, sal, annsal from emp table.

\rightarrow Select ename, sal, sal*2 "annsal" from emp;

ename . sal annsal

Column alias.

➤ select sysdate from dual;

sysdate

17-FUN-13

➤ select sysdate+1 from dual;

Tomorrow date

➤ select sysdate-1 from dual;

Yesterdays date

➤ Se

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

➤ Select * from emp where job != 'clerk';

After where compulsory write column

Q> display the employees who are getting more

than 2000 sal from emp table.

➤ Select * from emp where SAL > 2000;

using ANDOp's

➤ Select * from emp where JOB = 'clerk' AND
SAL > 2000;

➤ Select * from emp where job = 'CLERK' OR

sal > 2500

OR \rightarrow Clerk and

all sal above 2500. display

OR is nothing but 'g' (comma)

→ display the employee who all belongs to

deptno 10, 30, 50, 70, 90. from emp

➤ Select * from emp where deptno = 10 OR

deptno = 30 OR deptno = 50 OR deptno = 70 OR deptno = 90;

↓

Instead of we can 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

Ans: This opr is used to ~~take~~ ^{Pick} the values

one by one from list of values.

→ generally we are using Ans opr instead of 'OR' operator.

→ Bcz 'Ans' opr performance is very high compare to OR opr

→ In multiple row subqueries we must use Ans operator.

System [select * from tablename where

Columnname IN(list of values);

↓
Separated with ,.

→ Select * from emp where deptno IN(10,20,30,40);

→ Select * from emp where ename IN('King', 'Smith');

→ NOT IN ^{with} NULL
does not work

↓
No rows selected

Y Select * from emp where Deptno

Not in (20,30);

— It displays deptno 10.

Y Select * from emp where Deptno

NOT in (20,30, NULL);

— no rows selected.

Note: NOT in opr does not work 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 arithmetic 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
 $2400 + \text{NULL} = \text{NULL}$.

from emp where ename = 'SMITH';

O/P:-

ename	sal	comm	sal+comm
Smith	2400	(NULL)	(NULL)

→ Sal + comm

$2400 + \text{NULL} = \text{NULL}$

— If ^{like} the above ex. any problem is there to overcome
problem
NULL).

→ To overcome this problem oracle provided

NULL) function

Ex:- $\text{NULL} + 50$

→ NULL

In place of NULL we replace '0'

NULL) $\rightarrow 0 + 50$
 $\rightarrow 50$.

fun automatically ^{user defined value} substitute '500'

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

Syntax :-

$$\boxed{\text{NVL(expr1, expr2).}}$$

must be same data type only.

→ These 2 expressions must belongs to same data type.

Ex :-

$$\text{NVL(Null, 0)} = 0$$
, $\text{NVL(Null, 20)} \rightarrow 20$

1st exp - Null, 2nd exp value = 2nd exp

- If Expr1 is Null It returns expression 2.
- Otherwise It returns expression 1.

Ex :- $\text{NVL(30, 20)} = 30$.

→ select NVL(Null, 20) from Dual;

$$\boxed{\text{NVL(Null)}}$$

20

→ Select ename, sal, comm, sal+NVL(comm, 0)

from emp where ename = 'SMITH';

ename	sal	comm	Sal+Comm
Smith	2000	(null)	2000

→ Sal + NVL(Comm, 0)

2000 + NVL(Comm, 0)

2000 + 0

= 2000

18 - 6

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

Q

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 & NOT NULL; for
not
Index, not work.

These oprs are used in where clause only.

→ These oprs are used to Test whether a column having null values or not.

Dept No

10 — Display employees 10th dept.

20 → Select * from emp where

50

70

90

DeptNo=10

— who are working in 10th 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 ^{whose}

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.

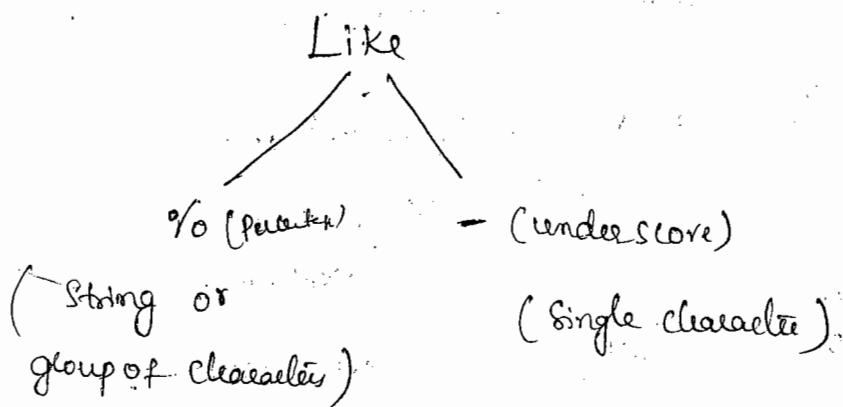
→ Select * from emp where

Comm IS NOT NULL;

Comm values
display

LIKE: This opr is used to retrieve data very fastly based on character pattern.

→ along with like opr 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:-

ALLEN MARTIN
%o. %o
 ^ ends with 'N'

Display the employees whose ename start with 'M' from emp table using like opr.

→ Select * from emp where

ename like 'M%';

→ Martin
→ Miller

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

Y Select * from emp where

ename like '-Lolo;

- ALLEN

— BLAKE.

- CLARK.

— Concatenation $\text{Op} := \underline{(||)}$ (2 pipes).

It is used to Concatenate diff. data-type Columns and also used to Concatenate Column Values with Strings.

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

my employee name is Smith
4 4 4 Allen

Functions

- Functions are used to solve particular task and also fun's return a value.
- All dban systems having 2-types of fun's

- (1) predefined fun's (in SQL)
- (2) user defined (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.	<code>abs(-50)</code>
	50

→ Test Predefined fun we use 'dual' table;
Virtual table → Predefined 1 row
generate sequel values.

Y Select abs(-50) from emp; Select abs(-50) from dept;

① 50
50
50
50

② 50 - 4 rows selected.

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.

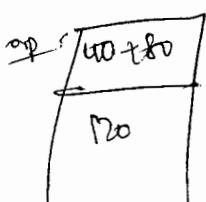
→ generally this table is used to test Predefined,
User defined fun's functionalities. and also

→ This table is used to generates Sequence Values
through Sequence object. & also

→ This table is used to perform mathematical
operations (in projects like calculator). Y select 40+80 from dual;

40+80
120

Y Select 40+80 from dual;



✓ select * from dual;

column.

Ø (dummy)

-

X

row

✓ desc dual;

Dummy

datatype

Value12 (D)

→ Select predefined or user defined

1st check on dual bcz it is proper table or not.

✓ select mul(null,50) from dual;

OP:- 50

✓ select mul(30,50) from dual;

OP:- 30.

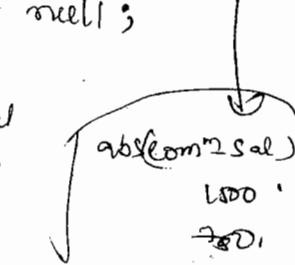
✓ select ename, comm, sal, com-sal emp

abs(comm-sal)

where comm is not null;

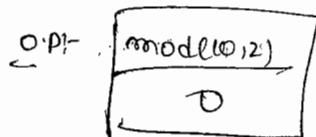
OP:- ename comm sal com-sal

1	1	1	-100
1	1	1	-50



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

➤ Select $\text{mod}(10,2)$ from deal;



➤ greatest(expr, expr, ..., expr); least(expr, expr, ..., expr)

→ greatest returns "maximum value" within given expressions

→ when at least returns min value among given expressions.

➤ Select greatest(3,6,9) from deal;



→ Display greatest sal from emp;

➤ Select greatest(sal) from emp;

Op:-

1800
1200 } Same values but not greatest value,

→ Single column don't apply

greatest → expr, expr, ...

max → Single column.

ename com sal

Smith	300	1800
		→ now greatest (sal, com)

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

(E.C)
113V
18

→ Select ename, comm, sal,

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

19-6

when ever necessary flng 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'.

→ Select round (1.23456, 3) from dual;

→ after decimal 3 places.

4.23410,3
↓
1.234 only

OP 1.23456
↓
1.234 above 50%
↓
1.235 add 1

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

→ Select ename, sal from emp;

113y
 > Select ename, sal, round(sal/2), round(sal/2,1)
 from emp;

O.P. sal 1132.5
 1192.7

Trunc (m,n) :-

It ^{removal} ~~Truncates~~ gives floaled member in,
 based on 'n'.

> Select trunc(1.7) from deal;
^{→ High place}
^{Truncates}
 O.P :- '1'.

> Select trunc(1.23456, 3) from deal;
[↓]
^{Truncated}
 O.P :- 1.234.

→ Ceil(), floor()
[↓]
 returns a
 nearest greatest
 integer value

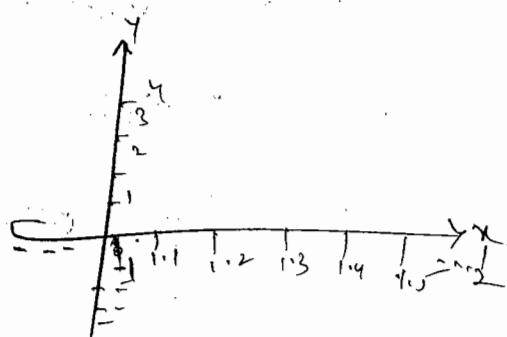
nearest, lowest

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

→ select floor(1.9) from deal;
 '1' (nearest lowest integer)

→ Ceil() returns nearest greatest integer.

→ Whereas floor() returns nearest lowest
 integer



Character functions

These fun's operate over character data.

① upper() :-

upper('abc')

upper('ename')

↳ col name do not use ' '.

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

>Select * from emp;

all enames are upper case

→ select upper('ename') from emp;

→ select upper('abc') from dual;

② lower() :-

Select lower('ename') from emp;

→ 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 should 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 including spaces also en: 'sh. su. the' = 9/ length

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

o/p: 5 when $length(ename) = 5$
Select emp from where
length(ename) = 5
- Bruce
- Miller

*** used in every d.b.

Substr

Substr () :-

Output

Output

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

> Substr (' ABCDEF ', 2, 3)

Counting
↓
BCD

→ 2, 3

~~ABC~~ . GH

→ S14) (-5)

DEF6.

DEF6, H

> Select Substr (' ABCDEF GH ', 2, 3) from dedb;

Output BCD.

(-614).

CDEF.

Syntax:-

Substr

(Colname (or) 'String name', Starting position, no. of characters, from position)

must be
int's

Ne

9.

→ Select * from emp;
→ Using Substring display the employees whose
ename's second letter would be 'LA' (substring)
from emp table.

[BLARK]
[CLARK]

Powered

✗ substr (ename, 2) → wrong.

✗ Select * from emp where

→ Select * from emp where

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

O.P:- BLARK, CLARK.

→ After Select in most fun & character fun's
where clause is possible.

after selection
select max

→ Select * from emp where

after

max (so)

→ group fun is not accepted

Note:- We are not allowed to use group fun's

in where clause. But we ^{all} allow to use
no. fun's, character fun's, date fun's in
where clause.

→ INSTR() → Most powerful. But dead easy

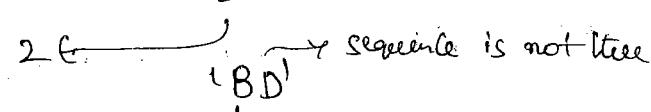
Always this fun returns no. data type

— It returns position of the DELIMITER (special symbols $\{, \}, *, \&, \$, \%$)
position of the character, position of the string within given string.

Y Select INSTR ('ABCD', '&') From dead;

O.P:- 0; 
0
0

INSTR ('AB*CD', '*')

O.P:- 3. 'A'
1 'B'
2 'BC'
26 
'BD'
OR

INSTR ('AB*CD*EFG*KJH*NM', '*'); molecule no. of occures

O.P:- 3. (3, *, 2) (1, *, 3)

O.P:- 9. O.P:- 13

INSTR() is always contains '4' parameters

→ goes to 2nd star.

(1, *, 5)

O.P:- 0;

(1, *, 5, 2)

O.P:- 13

easy

type

ITER (special
symbols
\$, *, ?)

$\frac{AB*}{\text{Count}}$ (11, '1*'), -5, 2)

O.PT 3.

$\frac{AB*}{\text{Counts}}$ (11, '1*'), -4, 2)

O.PT 3.

(11, '1*'), -3, 2)

O.PT 3.

-3 - it is also count.

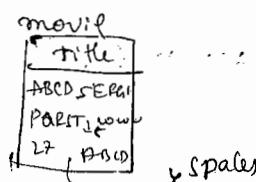
Syntax:-

instr (columnname (or) 'String name', 'Str',

Searching, no. of occurrences
Position from Position
(must be nos.)

Note:- Always instr() returns position of
the string 'band' on last '2' parameters

But Oracle counts no. of characters
left side 1st position onwards (AB*)



> Select substr (title, 1, *, instr(title, '\$')) from movie;

→ lpad:- (and is hierarchical query) 20.6
 adding some thing.

lpad ('ABCD', ^{total} ^{length} 10, ('*'))

→ 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.

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

O.P:- *****Smith.

→ RPad:-

→ Select RPad ('ABCD', 10, ('*')) from dual;

O.P:- ABCD*****.

> Select Rpad ('Ename', 40, 'c') from emp;

O.P: Smith -----

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

O.P: Smith ----- 2000

Allen ----- 4000

→ LTRIM():-

→ removed.

It removes left side specified characters.

> Select LTrim ('MISSTHSS', '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 - aleman.

Clerk - leek.

→ (JOB, 'CSM') from emp:

O.P: Clerk - leek.

Sales man - aleman.

Manager - anager.

> (JOB, 'CL') from emp.

O.P: Clerk - lek.

→ RTRIM:

>Select RTRIM ('ssmissthss') from dual;

O.P:-
= ssmissth

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

>Select Trim ('ssmissthss', 's') from dual;

MITH (wrong).
D.P. missth wrong syntax.

→ Trim remove only left & right side characters only.

→ middle characters are not removed

→ And also used to removed 1st & last places

Syntax:

Trim ('character @)
string , From
 Column
 given string @)
 Column name

→ Select Trim ('s' from ('semissthss')) from dual.

O.P:- missth

➤ Select Trim ('Smith') from deal;

Op: SMITH for clearing op. bcz there is spacing
↓

➤ Select length (Trim ('Smith')) from deal

Op: 5 (SMITH).

→ Translate(), Replace();

➤ Select translate ('JOHN', 'H', 'N') from deal;

Op: 'TONN'

➤ Select Replace ('JOHN', 'H', 'N') from deal;

Op: TONN

➤ Select Replace ('JOHN', 'H', 'NHFDFN') from deal;

Op: TONHFDFN.

➤ Select Translate ('JOHN', 'H', 'NHFDFN') from deal;

Op: TONN

➤ Select Translate ('JOHN', 'HNOJ', 'M@Q') from deal;

Op: QP@M.

→ Translate is used to replaces character by character where as

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

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

O.P. 4321

➤ Select Replace ('A B C', ' ', ' ') from dual;

O.P. A oracle B oracle C

→ String with string also Replace.

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

O.P. Salesman Connecting into marketeis.

~~Replaces~~ Select Replace ('SS mississ', 'S', ' ') from dual;

O.P. MITH.

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

DATE Functions: [Sysdate - 1 → yesterday]
[Sysdate + 1 → tomorrow]

→ In oracle by default date format is

dd-mon-yy

① SYSDATE → no. (reduce column)

⑤ months-BETWEEN()

② ADD_MONTHS()

③ LAST_DAY()

④ NEXT_DAY()

① SYSDATE— It retrieves Current System Date.

> Select Sysdate From Dual ;

Op: 20-Jun-13.

② ADD-MONTHS()— It is used to add or subtract no. of months to the specified date and on 2nd parameter.

> Select Add-months (sysdate, 1) from dual ;

Op: 20-JUL-13.

> (sysdate+15) > (sysdate, -1)

Op: 20-SEP-14.

Op: 20-MAY-13.

> Select add-months ('15-aug-05',1) from dual

15-Sep-05

↳ on date

③ LAST_DAY()— It retrieves last date of the Specified month.

↳ passing only 1 parameter

> Select last-day (sysdate) from dual

Op: 30-JUN-13.

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

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

➤ Select `Next-day(sysdate, 'Sunday')` from `deall`.
next Sunday

23-JUN-13.

⑤ MONTHS_BETWEEN() :- retires number date type
no. only

`months-between(Date1, Date2)`.

Always this fun. Retires no. date type. i.e., It retrieves no. of 'months' b/w 2 dates. Always Date1 more than Date2. [Date1 > Date2] otherwise - worse It retrieves -ve sign.

➤ Select `months-between('16-SEP-04', '16-JUL-04')`
from deall

➤ Select `months-between('16-FEB-04', '16-JUL-04')`;

O.P. -5

➤ Select `months-between('16-FEB-04', '16-JUL-04')`;

O.P. 5

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

— how many employees works.

➤ Select months - between (sysdate, hiredate) for emp;

→ Opt - 390.1600

bcz 1st, lowest date, 2nd highest date. } floating pts.

➤ Select months - between (sysdate, hiredate). 1/12

Opt: 390
36 →

1/12
↓
years.

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

➤ Select ename, round (months - between (sysdate, hiredate)/12) from emp;

Opt: Smith - 22
Allen - 33.

Date Arithmetic
in

① Date + Number ✓ (possible)

② Date - Number ✓

③ Date1 + Date2 ✗

④ Date1 - Date2 ✓ - 8

① Select sysdate+1 from deal;

Tomorrow.

② Select sysdate+1 from deal;

Yesterday.

③ Select sysdate - sysdate from deal;

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 deal;

01-JUN-03.

Date Conversion — To date.

To date.

21-6

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).

It is used to convert date type into date string.

>Select TO_CHAR(SYSDATE, 'Day month yyyy') from dual

Output: Friday JUNE 2013

→ Convert any date into client requirement format

>Select TO_CHAR(SYSDATE, 'DD/MM/YYYY') from dual.

Output: 21/06/2013.

Day
dy
Month
mon
Year

Return Value
Character

dd
DD
DDD
mon
yy

→ Select to_char (sysdate , ' DAY') from dual;

o.p:- ~~Friday~~ **FRIDAY** **day**
 'DAY' (DAY) (D) (DD00)
~~FR~~ 21 6 172.
 SAT

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

DAY - FRIDAY. DAY - FRI.

day - friday.

→ Select to_char (sysdate , ' D') from dual;

O.P:- 6 (21-~~dd~~dd)

Sunday - 1. 6-friday
 mon - 2. Friday - 6.

31



→ DDD → day of the year (completed days)

→ DDth → 20th, 23rd

→ Ddsp → Date Spell out. O.P:- Twenty-first

→ mon → JUN.

→ month - June.

→ MM → 06.

→ YEAR → Twenty thirteen. Y YYYY - 2013

→ HHTT: m T:SS → 12:04:46.
 hours $\frac{1}{60}$ min $\frac{1}{60}$ sec

→ YY - 13

FM - Full mode

- > If any space is occurs in To-char O.P. we use 'FM'.
- > Select to-char (sysdate, 'DD Month YY') from dual;

O.P.:

21/JUN/13

12013

- > Select to-char (sysdate, 'DD Month YY') from dual.

O.P.:

21/JUNE/2013

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

- > Select to-char ('15-Aug-05', 'DD Month YY') from dual;

Error O.P.: Invalid number.

> In TO-char() we must & should follow rules:

→ 1st Parameter → date-type.

→ whenever we are giving a single quote ' ' it is taken as string.

→ String is convert into date using TO-date().

- > Select to-char (to-date ('15-Aug-05', 'DD Month YY')) from dual;

O.P.:

15/August/05.

DAY

Monday/August/05.

→ If 1st parameter is date, no need to-date.

Error Note:- when ever we are using `TO-CHAR()` always 1st parameter must be date type. otherwise oracle server returns an error. `ORA-1722: Invalid number.` (ⁱⁿ SQL 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)

> Select to_date ('15 JUNE 2007') from deal;

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')

When using `Task` return types must match with `await`

DD - men - 44
↓ ↓ ↓
no - chae no. no no no
↓
do not match 15/06/07
↓
invalid month

no date no
DD-mon-yy
15 06/07
no no invalid
error :- month.

→ when ever we get an error in To-date in SQL
we use a 2nd Parameter Same as 1st Parameter.

➤ Select to_date('15/06/07', 'DD/MM/YY') from dual
Op: 15-JUN-07

Note: whenever we are using To-date
between values match with the default date format
otherwise use a 2nd parameter as same as 1st
parameter format.

Then only Oracle Server internally converts
date string into date type.

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

➤ Select to_date('09-feb-04') + 5 from dual

Op: 14-feb-04

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

↓
error so my to_date

Y Select ^{year, month and palandt}
to_date ('09-02-04', 'DD-MM-YY') + 5 days

O.P/ 14 Feb-03

Y) Display the Employees who are joining in the month December from emp table using To_date()

Y Select To_date(*Hiredate, 'DD-Dec-YY') from emp;

(error)

Y Select To_date(

This is Conditionally this query. So use where clause

Y Select * from emp where To_date(Hiredate, mon) = 'Dec';

O.P = Dec

(Hiredate, mon) = 'Dec';

Y display the employees who are joining in the

year 81 from emp table using To_date()

Y Select * from emp where To_date(Hiredate, yy) = '81';

→ we want to know year is 1987 or 1971 or 1871, 1771.

Y Select * from emp where To_date(Hiredate, yy) = 81;

Y Select min date > To-Date (headdate, '4444')

From Emp;

O.P.: - 17 Dec - 80 To Date
1980
1981

27-06

memps;

1980

1981

1982

1983

date (mm)=15

is the

class()

1) = '81';
then

12+1.

1441) ~~100~~

Date
in

Conversion Functions :-

13-6-2014.

(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

PLA 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; \times

Select * from emp where to-char (hiredate, 'mon') =
to o/p
'Dec';

PLA 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 character is to-clear so gaps is there in o.p.

Fill Mode:- (FM):

when ever to-clear returns gaps then we must use fill mode this mode is represented by 'FM'.

Ex:- Select to-clear (sysdate, 'DAY/MON/YY') from dual

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

Select to-clear (sysdate, 'FM DAY/MON/YY')

O.P:- FRIDAY/JUN/2014

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

> Select * from emp where to-clear (H1ledate,

'month' = December;

O.P:- NO rows Selected.

> --- 'FM month' = December;

O.P will occurs.

to-clear helps in another situation

does Year is 1980 or 2080 @ uro.

> Select H1ledate, to-clear (H1ledate, 'YYYY')

from emp;

O.P:- H1ledate
1980 1981 1982

Note:- In Oracle whenever we are passing

date string '05-Jun-06' into date function then

(last-day)

next-month).

use 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.

For this case we must use to_date (It error will come)

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

O.P.F 30-JUN-05.

Select last-day ('15-06-05') from dual

O.P.F not a valid month.

Solutions:-

Select last-day (to_date ('15-06-05', 'DD-mm-yy')) from dual;

O.P.F 30-JUN-05.

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

* Every d.b. date data type ^{contains} is 2 parts

① date part ② time part

(visible)

(not visible)

→ In oracle when ever we are using round(), trunc() fun's of dates ^{then} automatically date part is changed based on timeportion.

→ whenever we are using a round() fun then oracle server automatically add 1 day to the date in time portion i.e. 12 Noon. And also time portion automatically set to 'Zero' (00:00:00)

Testing at 7:40 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 time() oracle
it returns same date and also time portion

automatically set to 'Zero' (00:00:00). If time portion \geq 12 Noon also.

➤ Select to char (trunc(sysdate), ('DD-MON-YY

HH24:MI:SS') from dual;

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

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

➤ Exe Insert into emp (empno, ename, hiredate)

values (1, 'murali', sysdate);

➤ Select * from emp where hiredate = sysdate;

\Rightarrow no rows selected.

Solution :-

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

1 murali 13-JUN-14.

Group functions (or) aggregate fun's:		14-6-14.
		compulsory part
(1) max()	(4) sum()	Column name not our own
(2) min()	(5) count(*)	
(3) avg()	(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; (freshie)
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)

Note:- We are not allowed 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 except "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. data type column.

➤ Select $sum(sal)$ from emp;

34075

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

Y Select Count(*) from emp;

O.P:- 14 (rows).

Count (Column name) :-

It Counts no. of not null values within a Column

Y Select Count (Comm) from emp;

O.P:- 4.

Y 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 Column name from Table name

Group by Column name;

Y Ques to display no. of employees in each department from emp table using group by.

Y Select Deptno, Count(*) from emp group by Deptno;

Deptno	Count(*)
30	6
20	5
10	3

W.A.Q 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:-

<u>Deptno</u>	<u>min(Sal)</u>	<u>max(Sal)</u>
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"

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

group by deptno, ename,
↓

O.P. will come. here we do not use group-fun

Y Select deptno, job from emp

group by deptno, job.

O.P. ✓

Y Select deptno from emp
→ here less cols is workers

group by deptno, job;

→ but here after select how many cols here
also that no. of cols

<u>Deptno</u>
20
30
40
10
20
10

Note: when we are try to display group fun columns
with another column then database servers return
errors to overcome this problem we must use
group by clause

Ex: ^{Step 1} Select sum(sal) from emp;

➤ Step 2

➤ Select deptno, sum(sal) from emp;

error: ~~sum~~ not a single-group function.

Solution:- using group by

Select deptno, sum(sal) from emp

group by deptno;

<u>O.P:-</u>	<u>Deptno</u>	<u>Sum(sal)</u>
	30	9100
	20	13625
	10	11350

16-6-14

➤ W.A.Q 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 = 30;

Error:-

after group by we can't use where clause

Sol:-

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

O.P:-

<u>Deptno</u>	<u>Count(*)</u>
30	6
20	5

Having:- After group by clause we are not allowed 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 we want to restrict groups after group by then we are using "Having clause".

⇒ Generally we are not allowed to use group func in "where clause".

Whereas we can also use group func's in "Having clause".

Order by:- not allowed 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. They are Asc, Desc. (1) Asc (2) Desc

→ by default Order by clause having 'Asc' order

Syntax Select * from table name

Order by column name [asc/desc];

↳ Select Sal from emp order by Sal desc

OP: 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
OP: eno ename - must use only when use the column position.
1
X
X
:

↳ Select deptno, Count(*) from emp

where Sal > 1000

group by deptno

group having Count(*) > 2

Order by deptno asc.

deptno	COUNT(*)
30	3
20	5
10	4

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

→ Oracle 8i introduced rollup, cube clauses. These clauses are optional clauses use along with group by clause only.

→ There 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 band on a single column.

Whereas if you want to calculate Subtotal, Grand total band on no. of columns then we are using 'Cube'.

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

only this column total

→ Select deptno, job, sum(sal), count(*) from emp group by cube (deptno, job);

→ WAP to sumof sal and with ename with grand total
Select ename, sum(sal) from emp

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

Joins.

Joins are used to retrieve data from multiple tables.

Rule- If you are joining 'n' table then we are using (n-1) join conditions.

- Oracle Server having following types of joins

- 8th 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 (9th)

These joins are called "8th joins"

9th 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 is why only in Oracle.

Internally Oracle Server uses Cross join by default.

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

Cross join between more duplicate data because Cross join works based on Cartesian product.

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

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

Based on equality Condition we are retrieving 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 column name = tablename2.

Common column name
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 we 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. Then alias names also called as reference names created in from clause and also their reference names must be different names.

Syntax from tablename1 aliasname1,
tablename2 aliasname2.

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

Note: using equi join - we are retrieving "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)

— Way to display the employees who are working
from emp, dept table.
In the location Chicago using equi join

Select e.ename, d.lo e.~~location~~

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

Ques - to display dname, sum(sal) from emp, dept table using equi join

Select d.dname, sum(sal)

from ~~group by~~ ~~deptno~~ dname, 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.

<u>Deptno</u>	<u>Dname</u>	<u>Sum(sal)</u>	<u>Outer join</u>
10	Accounting	18247.5	

Using clause:-

Ques - 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

Q1 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.

Q2 to display deptname & sum of salary of department where no. of employees > 3 & also calculate total of salary automatically

X 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

Note :- 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 \geq losal and sal \leq 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 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 referring a table using diff table name then we must use table as aliasname using clause → All joins

These are also called referenced for a table.

These aliasnames must be diff. name.

These aliasnames behaves as table where query executed.

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

Q) write 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

Smith
Salary

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

ename sal

SMITH 1600

ALLAN 1600

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

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

emp e₂

empno emp mgr

7369 SMITH 7902

7499 ALLEN 7932

➤ Select e₁.ename "employees", e₂.ename "managers" from emp e₁, emp e₂ 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

Outer join:-

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

Generally equi join retrieves matching rows only if we want to retrieve non matching rows also then we are using join operator (+) in joining condition of the equijoin this is called 8i outer join

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

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

matching rows → All rows

O.P.: 40 operations
Best DB

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 conditionnal 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) SQL 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';

<u>ENAME</u>	<u>LOC</u>
WARD	CHICAGO
TURNER	CHICAGO
ALLEN	CHICAGO
JAMES	CHICAGO.

22-7-14

Using clause:-

→ In q_i joins, we can also use using clause in place of 'on' clause using clause performance is very high compared to on clause. Always using clause 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

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

A	B	C
x	y	z

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

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

11. *Thien*

Joining

m

left outer join :- (left side table).
 This join always retains all rows from left side table  & matching rows from right side table & also retains 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 $t_1.b = t_2.b$;

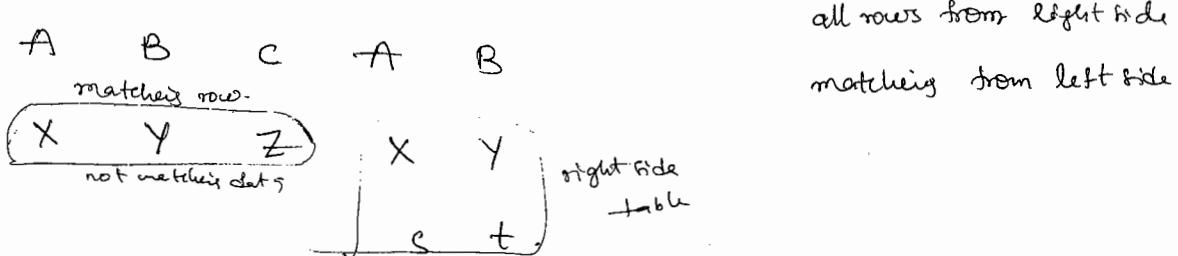
A	B	C	A	B
X	Y	Z	X	Y
P	Q	R	(null)	(null)

ll
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 in place of non matching rows in another table.

Ex:-

>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	X	Y
Q	-	-	S	T

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 8th qn.

where not requires

Natural join

this join also returns matching rows only.

1. ~~rows~~ only.

① In this join we are not allowed 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 that's why these joins 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 allowed to use alias name on joining conditional column. Bcz natural join internally uses using clause.

Ex:- Select

d-Dept X

ename, sal, deptno, dname, loc

from emp e natural join dept d;

Cross join:- (default join)

➤ Select ename, sal, dname, loc
from emp cross join dept;
→ like all
dname, loc from emp,
dept;

O.P.T. → 10 rows (56 rows) 14x4

Joining
(2 conditions)

Joining more than 2 tables.

Ex:- Joining 3 tables. → 2 join conditions.
8th join.

Syntax:-

Select col₁, col₂
from table name₁, table name₂, table₃.

Where

table name₁. Comm col = table name₂. Comm col₂,

AND.

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

9th join. → (3 tables) (ANSI joins)

Syntax:-

Select col₁, col₂

from table₁ join table₂

on

table₁. Comm col = table₂. Comm col

Join table₃.

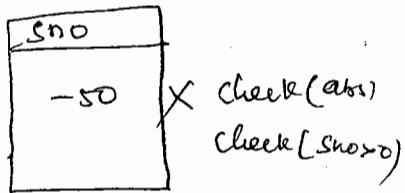
on

table₂. Comm col = table₃. Comm col.

2. Constraints

23-7-14

t1



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

— Constraints are created on table columns.

— Oracle Server having following types of constraint

- (1) not null (3) primary key. (5) check.
- (2) unique (4) foreign key.

> For all the above constraints are created in 2 ways.

(1) Column level.

(2) Table level.

(1) Column level: In this method we are defining ^(Creating) constraints on individual columns.

in this method

e.g. When ever we are defining the column then immediately 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 varchar(20) primary key,

Accname varchar(20) primary key; XX

In this method we are defining constraints on group of columns. In this method i.e., 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 datatype(size),

Col2 datatype(size), ..., constraint-type (Col1, Col2, ...);

constraint types	① Not null.	Column	Table	Create table bank (Acc number(10), name varchar(20), balance number(10))
or	② unique.	✓	✓	
constraints	③ P.K.	✓	✗	Primary Key (Accno, name)
	④ F.K.	✓	✓	
	⑤ Check	✓	✓	

notnull	unique
✓	✗
✓	✓
✗	✗

→ Only 1 P.K. in 1 table.

→ P.K.

→ Uniquification of record in a table

→ Acc no, name, balance
1001 multi 00000

→ If 1 P.K. column not possible programer assumption 1002 abc 20000

Combination of columns will take as P.Key

① NOT NULL :- Not null constraint does not support table level.

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

Column level :-

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

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

testing :-

> insert into d₁ values (null,'abc');

error :- Cannot 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 accept duplicate values but it will accept null values (oppo. not null). unique in P.K. column using where clause bcz (internally db server creates indexes) bcz improving performance.

Note :- when ever we are creating unique constraint oracle server internally automatically creates 'btree' indexes on those columns.

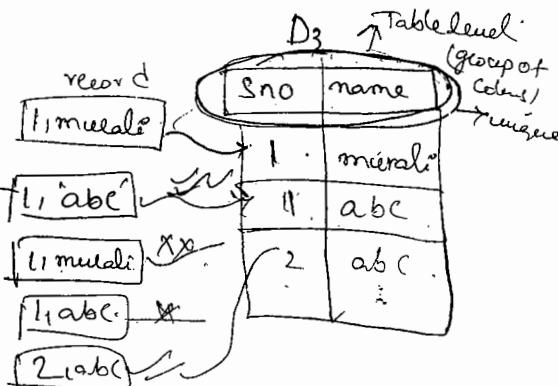
Column level :-

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

Table level :- record 

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

Table level to check record



> Select * from d3;

Sno	Name
1	merali
1	abc

➤ Insert into d3 values (1, 'abc');

error: ORA-00001: unique constraint violated.

③ Primary key :- P. key. uniquely identifies ~~one~~ 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 accept
- duplicate, null values. and also
 - when we are creating p.k. oracle ~~seems~~ 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 establish relationship b/w tables then we must use referential integrity constraint

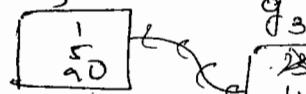
"Foreign key".

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

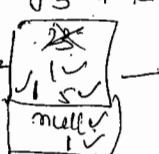
→ Here then 2 columns must belongs to same datatype.

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

d3 P.K.



g3 F.K.



→ duplicate & nulls also accepted

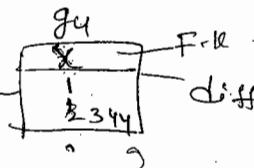
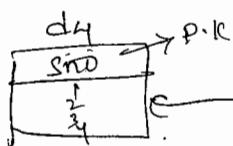
→ Generally P.K. does not accept duplicate, null values, whereas F.K. accepts duplicate, null values.



Column level:-

create table g4

(sno number(10) references d4(sno));



diff col name ✓
Pdt type is same only it works. ✓
accepted -

create table g4

(x number(10) references d4(sno));

→ Column level :- (referenes) :-

Syntax :-

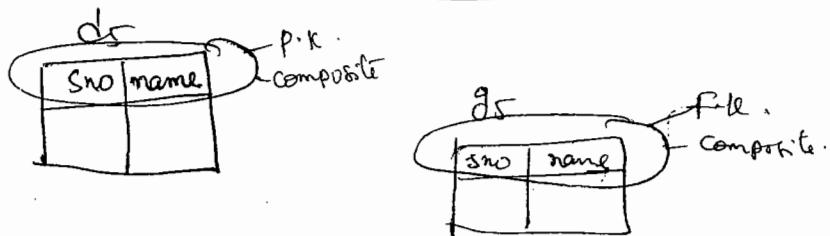
create table tablename (col1 datatype (size))

referenes 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
 master tablename (primary colnames));



> 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 seems automatically violates
 2 rules.

(1) Deletion in master table.

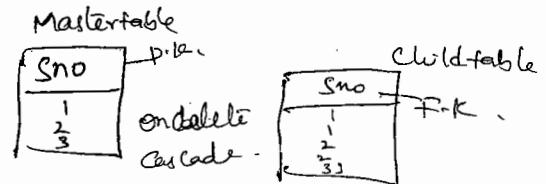
(2) Insertion in child table.

① Deletion in Master table :- In oracle if we all 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 are using "on delete Cascade clause"

→ On delete Cascade clause :-



when ever we are using "on delete Cascade clause" in child table then if we 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);

→ Create Insert into mas Values (.....);

→ Select * from mas;

Sno
1
2
3

- > Create table ~~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
3
3

On delete set null :-

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

— whenever 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 masstable (Primary Key Colname) on delete set null;

leaves masstable (Primary Key Colname) on delete set null;

ales

Insertion in child table :- In oracle if we are trying to insert other than P.K. values into foreign key then oracle server returns an error ORA-2291.

→ BECAUSE in all d.b. system F.K values are based on P.K. values only.

⑤ Check :- This constraint is used to define logical conditions according to our business rules 25-7-14.

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-00533)

➤ Insert into test values (8000); // \$ is now created.

➤ Select * from test;

SAL
8000.

Note :- Check constraint does not work with "Sedate".

It allows user defined date.

- Insert into
- Create table test1(name varchar2(10))
 - Check(name = upper(name));
- Insert into test1 values ('murali');
- error ORA-02290: check constraint (SCOTT.SYS-0005336)
- Insert into test1 values ('MURALI'); Violated
- 1 row created

Table level :- (AND) operator

- Create table test2 (name varchar2(10), sal number(10),
 - Check(name = upper(name) and sal > 5000);

Assigns undefined 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 format of SYS-00056789 @
SYS-CN # 8h

This is called "Predefined Constraint name".

Instead of this one we can also create our own name using Constraint keyword. This is called undefined Constraint name.

Syntax :-

Constraint	user defined name	Constraint type
Pre defined	Constraint name :-	P.K natural F.K check

- > Create Pre defined 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-fan (SCOTT . P-FAN) violated .

Dictionary :-

- when ever we are installing oracle Seem like a lot of tables are created
- These read only tables are also called as Dictionary

all constraints are stored in

Y Yes USER_CONSTRAINTS;

oracle database

user_Constraints

owner	Constr-name	Const-type

user_cons_columns

Constrname	Column-name

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;

Y Select constraint-name, constraint-type from

user_Constraints where table-name = 'EMP';
after = Joins 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 table

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;

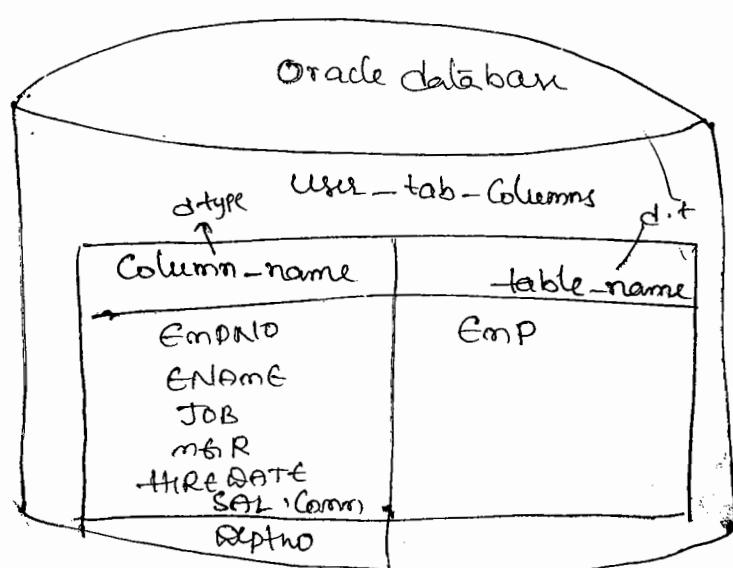
> Select SEARCH_CONDITION from user_constraints
where TABLE_NAME = 'TEST1';

SEARCH_CONDITION

name = upper(name)

→ Select count * emp; no. of rows → 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

MIN

HIRE DATE

SAL

Comm

Deptno

no. of Column no.

* If interview question

SQL 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(10)
default 2000);

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

➤ Select * from test;

name sal

abc 2000 - default

➤ Create 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

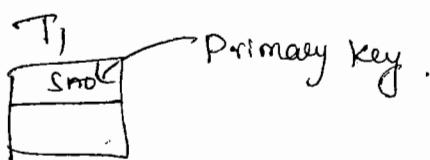
adding or dropping constraint on existing table
 we can also add and dropping constraint on existing table using 'alter' command.

Note: new constraint :- Primary key

(1) If u want to add constraint on existing table existing column then we are using table level

Syntax method. (1st constraint then column ex primary key (sno))

Y

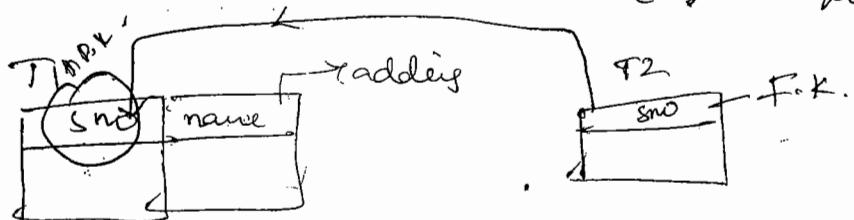


Y Alter Table T1 add primary key (sno);

new column unique

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

Y alter table t1 add name varchar(10) unique;

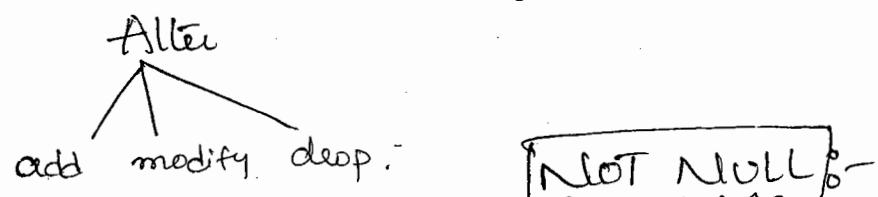


Y adding foreign key } add 2
 references } add 2

Y create table t2 (sno number(10));

Table created

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



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

↓
problem is we are modify.

Syntax :-

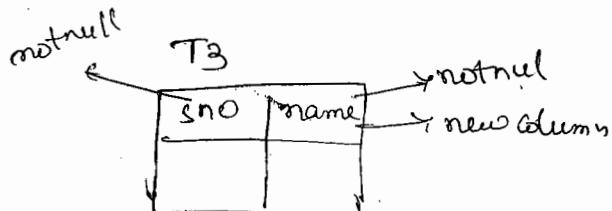
alter table tablename modify Columnname
not null;

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

> alter table t3 modify sno not null;

→ new column:-

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



→ Copy 1 table to another table. Constraints are not copied.

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

106

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

In this case we must use alter cmd which is used to add constraint on existing table to existing column.

- > Create table newemp as Select * from emp;
- > Create table newdept as select * from dept;
- > alter table newdept add constraint D-hh
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;
dropping constraints Ans 28-9-14 monday.

Method 1:-

Syntax:- Alter table tablename drop constraint Constraint name

Method 2:-

Syntax:- ① Alter table tablename drop primary key;

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

from

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

१०८

> alle 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 number(10) primary key);

7. Create table t_2 (sno number (10) reference t_1);

> alter table *ti* drop primary key;

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

Solutions:-

d)

7 alter table to drop primary key cascade;

→ Table attēd

Ex:- Create table Y1(Sno number(10) Constraint as notnull);

> alter table Y1 drop Constraint ab;

-- Search constraint name from user-con-
Column 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 2 types of Subqueries
 - (1) Non Correlated Subqueries, 1st → Q.1
 - (2) Correlated Subqueries, 1st → P.1
- In non Correlated Sub queries, child query is executed 1st then only parent query is executed. whereas in Correlated Subqueries, parent query is executed 1st, then only child query is executed.

Non Correlated 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-Correlated Subqueries:-

- (1) Single row Subquery
- (2) Multiple row Subquery
- (3) Multiple Column Subquery
- (4) Inline views (or) Subqueries and in from clause

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

➤ Select * from emp where Sal \geq (Select avg(sal) from emp);
→ Here, child query returns single value. That's why, these type of subqueries are also called as single row subqueries. 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 \geq 2101.78571.

WAP 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');

- { 1. Same columns
2. Group functions
3. Expression
4. diff. columns, but of same data type

WAP to display senior most employee from emp table

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

(Hiredate = 14-2-1987 DATE)

30-7-14.

WAP 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');

- WAP to display the employees who are working under BLAKE has manager 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')

WAP to display lowest avg. Salary job from emp table.

Select job, avg(sal) from emp group by job

where ~~avg(sal)~~ avg(sal) = (Select low(avg(sal)) from emp);

- where clause is a parent clause
- Parent clause in having } } either where or having } it parent query

➤ Select job, avg(sal) from group by job having
avg(sal) = (Select min(avg(sal)) from emp)
child query
nested group fun.

⇒ Oracle ORA - nested group fun without GROUP BY

Note:- In all database 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

SQL to display the jobs whose avg salary more than
the manager's 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,

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

Dept no	max(sal)
30	2850
20	3000
10	5000

Q4. 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 column so we write subquery 1 column only

error Single-row subquery returns more than 1 row.
child query.

Allen 1000
Smith 1000

900 \neq (2000
3000
5000)

900 = 2850
3000
5000

900 in (2000
3000
5000) ✓ 900 \neq n()

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

For 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.

→ WAP 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 returns multiple values
So, we cannot assign multi values to single variable

TOP-N-Analysis:-

(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);

Imp

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	0 smith-AAA BCD
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.

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

→ Select * from emp where rownum = 1;

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

> Select * from emp where rownum = 2;
Q: 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;

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

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

→ WAP 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);

→ WAP 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 workers

players

- > Select * from emp where rownum >= 1;
 - rows are selected.
- > Select * from emp where rownum > 1
 - no rows selected.

emp

Q) What 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 line view that
alias names works with all SQL operators.

to 30

11.11

others

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

Select * from (Select rownum δ , ename,



Sal from emp) where $\delta=5$;

R	<u>ename</u>	<u>sal</u>
5	MARTIN	1250.

(or)

Select * from (Select rownum δ , e.* from emp e)
where $\delta=5$

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

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

O.P	<u>δ</u>	<u>ename</u>	<u>sal</u>
	2		
	3		
	4		
	5		
	6		
	7		
	8		

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

Select * from (Select rownum δ , ename, sal from emp)
where δ in(2,5,7,9);

or 5 or 7 or 9

WAP to display 1st, last row from emp table using rownum alias name. $\text{Count}(\star) = 13^{\text{th}}$ record

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

where $\sigma = 1$ ~~and~~ or $\sigma = (\text{select Count}(\star) \text{ from emp})$.

(Or) $\sigma \in (1, 14)$.

R ENAME SAL.

1	SMITH	800
4	MILLER	1300

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

— Select * from (Select rownum σ , ename, sal from emp)

where ~~$\sigma \leq 10$~~ ~~$\sigma \mod(2) = 0$~~

$\mod(m, n)$

2
4
6
8
10
12
14

(%)

mod

$\sigma \leq \frac{10}{2} = 5$

where $\mod(\sigma, 2) = 0$. even

$\mod(\sigma, 2) = 1$. odd 3 5

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

— Select * from (Select rownum σ , ename, sal from

(Select * from emp order by sal desc) where $\sigma = 5$

R ENAME SAL. σ en sal

5	BLAKE	2850.
---	-------	-------

Analytical Functions used in SQL views :- (A.6)

Oracle '8i' introduced analytical fun's then all

(1) row-number()

(2) rank()

(3) dense_rank()

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

Syntax:-

Analytical function name() over

(partition by Columnname order by Columnname [asc/dsc])

	denerrank	rank
2000	1	1 - 2000
2000	1	2 - 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, denerrank analytical fun's automatically

assigns same rank numbers when values

are same. ($\frac{2000}{2000}$)

- Al(4.6) → Rank skips next consecutive rank no's
 2e → whereas dense rank does not skip next consecutive 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 (partition by deptno order by sal desc) of from
 emp) where r <= 10;

interview ^{rows - rank}
 dept

interview to display 2nd highest salary employee in each.

Optment from emp table using analytical fun.

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

where r = 2

O.P :-

Deptno	ename	sal	r
10	clark	2450	2
20	Jones	2975	2
30	Allen	1600	2.

WAP to display 5th highest sal employee from emp table using analytical fun.

- Select * from (Select deptno, ename, sal,

dense_rank() over (Partition by ~~Deptno~~ Order by ~~Deptno~~
sal asc) r from emp) where r=5; ~~overall table by~~
~~this query 3 department's 5th highest~~

O.P
10 — 2
20 — 5
30 — 5

but we want only 5th highest overall
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-We.

AAB MXAAA AB. now add

AAB
AB

P row

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

→ 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.

→ Using row id we are retrieving data very fastly from the database.

emp table has 1 no. col

row id has 3 ... certain

changes in rowid

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

→ Select rownum, rowid from emp;

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

→ We can also use max, min fun's in rowid

Ex: ① Select min(rowid) from emp;

② Select max(rowid) from emp;

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

W.A.Q to delete duplicate rows except 1 row in each group from a table.

➤ Select * from test

of 20
Interview
@

➤ delete from test where
from test group by Sno;

➤ Select * from test;

SNO
10
20
30
40
50.

~~10
10
10
20
20
30
30
40
50.~~

rowid not in (Select max(rowid)
not correlated
min(rowid)
(00)

Multiple Column Subqueries

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'

Parent where clause compulsory exists '()'

multiple,

Parent query \rightarrow where '()'

When we are using multiple Column Subqueries when we must specify parent query where Conditional

each
Glimpses within "Panther's U"

> Select * from tablename
where (col1, col2...) in (Select col1, col2 from tablename
where Condition);

WAP to display the employees whose job, mgr match
with the job, mgr of the employee '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	

WAP to display the employees who are getting max
Salary in each department from emp table using
multiple 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

- Was 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	2800	Blake
20	3000	Scott
10	5000	King

Was to display Senior most employees in each job from emp table using multiple column Subquery.

- Select job, hiredate, ename from emp ~~where (job, hiredate)~~ (Job, ~~min(hiredate)~~) where (job, hiredate) in (Select job, min(hiredate) from emp group by job).

job	hiredate	ename.

WAP 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-W.

Correlated Subquery:- Generally in non-correlated subqueries child query is executed first. Then only parent query is executed. Whereas in Correlated Subqueries parent query is executed ^{1st} then only child query is executed.
→ when ever we are submitting Correlated Subquery then database servers 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.

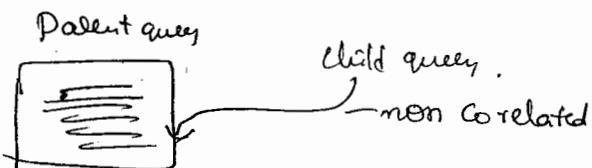
→ In Correlated Subqueries we must create alias name for the parent query table in parent query and use this alias name in child query 'where' Condition.

Syntax :-

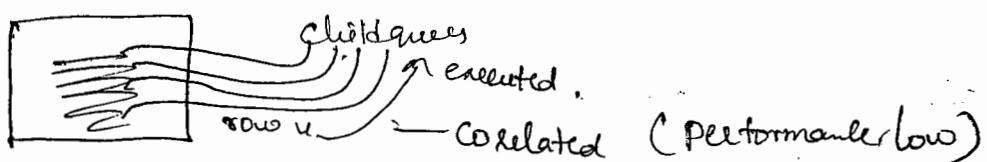
Select * from tablename aliasname1

Where Columnname operator (Select * from tablename

where columnname = aliasname1. Columnname)



In non correlated Subqueries child query is executed only 1e per parent query table.



→ whereas in correlated Subqueries child query is executed for each and every row for parent query table.

→ Generally Correlated Subqueries are used in de-normalization process. (combined)

In this process they are using Correlated updates.

→ Generally If we want to modify 1 table Column values based on another table Columns then we are using Correlated update.

date tablename1 aliasname1 set

Column name = (select Columnname from tablename2

aliasname1 where aliasname1. Common Colname =
aliasname2. Common Colname);

Ex:-
in

SQL> alter table emp add dname VARCHAR2(10);

SQL> update emp e set dname = (select dname from
dept d where e.deptno = d.deptno);

SQL> Select * from emp;

ename	eno	sal	mg	dname

Ques - to display the employees who are getting more
than the avg salaries of their jobs from emp table
using Correlated Subquery.

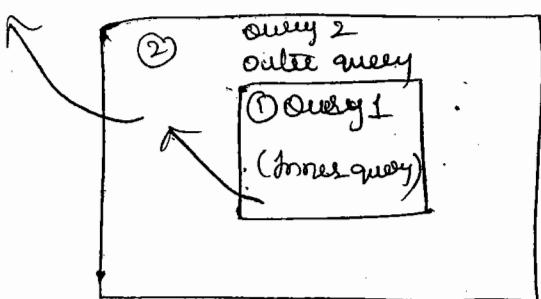
SQL> Select * from emp e where sal > (Select avg(sal)
from emp where job = e.job)

empno ename job mbr hiredate sal comm deptno.

- In non Correlated Subqueries always inner query is independent from the outer query, whereas in
- In Correlated Subquery always inner query depends on outer query. i.e., inner query, outer queries are Correlated.

non Correlated Subquery

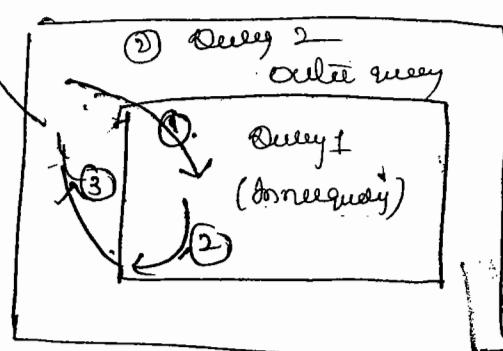
final output



uni direction

Correlated Subquery

final P.



Bi-direction

Q. to display 1st highest salary employee from using Correlated Sub query.

➤ Select * from emp e₁ where 1 = (Select count(*) from emp e₂ where e₂.sal >= e₁.sal).

O.P.:

KING \$100

How to display 2nd highest Salary employee from the following table using Correlated Subquery.

➤ Select * from test;

0.1 ENAME	SAL
abc	100
xyz	150
bbb	200
ccc	300

Select * from test e_1 where $2 = (\text{Select Count(*)} \text{ from test } e_2 \text{ where } e_2.\text{Sal} \geq e_1.\text{Sal})$;

0.2 ENAME	SAL
bbb	200

Execution process.

Phase 1 :-

Step 1 :- get a Candidate row (1st row) \rightarrow abc 100

Step 2 :- \rightarrow Select Count(*) from test e_2 with 100
where $e_2.\text{Sal} \geq e_1.\text{Sal}$

Step 3 :- \rightarrow Select * from test e_1 where $2 = 4$; (False).

Phase 2 :-

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

Step 2 :- Select Count(*) from test e_2 with 150
where $e_2.\text{Sal} \geq e_1.\text{Sal}$

Step 3 :- Select * from test e_1 where $2 = 4$; (False).

$$\begin{array}{r} 100 \\ 150 \\ 200 \\ 300 \end{array} \begin{array}{l} \underline{1} \\ \underline{2} \\ \underline{3} \\ \underline{4} \end{array}$$

$$\begin{array}{r} 150 \\ 200 \\ 300 \end{array} \begin{array}{l} \underline{1} \\ \underline{2} \\ \underline{3} \end{array}$$

Phase 3 :-

Step 1 :- (bbb, 200)

Step 2 :- $e2.sal >= 200$.

2

Step 3 :- $2=2$ (true)

Retrieves O.P is

200	bbb
-----	-----

7-8-14.

Note :- when our resource table contains duplicate data then above query does not return values to outcome this problem we must use "distinct clause".

Ex :-
1) > 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

$2 = (\text{select count}(\text{distinct}(sal)) \text{ from test e2 when}$
 $e2.sal >= e1.sal)$ duplicate values will display.

ename	sal
bbb	200
zzz	200

WAP to display 2nd highest salary employee from the above table using Correlated 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

bob 200.
zzz 200.

④

don't use '='
in (n-1)

WAP to display n^{th} highest salary employee from emp table using Correlated Subquery.

> Select * from emp e₁ where &n = (Select count (distinct (Sal)) from emp e₂

where e₂.Sal > e₁.Sal.

= is there in n

> Enter value for n: 1

KING 5000.

EXISTS operator or (in Correlated only)

- Exists operator used in Correlated 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 allowed to use colname

→ We are not allowed to use Colname.

Syntax:

Select * From tablename aliasname

where exists (Select * From tablename

where Colname = aliasname • Colname);

→ exists {} \Rightarrow False. {} set means
↳ empty Child query gives values.

→ exists {1,2,3} \Rightarrow True.
↳ nonempty

→ 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 exists {1,2,3} \Rightarrow True
exists {} = False.

→ Generally if we want to test 1 table Col values
are available in another table then we are using
Exists operator.

WAP to display those departments from dept table
those departments ^(employees) are available in emp table using

Correlated Subquery exists

Parent query alias name mandatory, child query alias name

> 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, for..., char 'a' works)

WAP 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
110	Operations	Boston

WAP to display those departments from dept table
does not have employees in emp table using not

Correlated Subquery.

> Select * from emp;

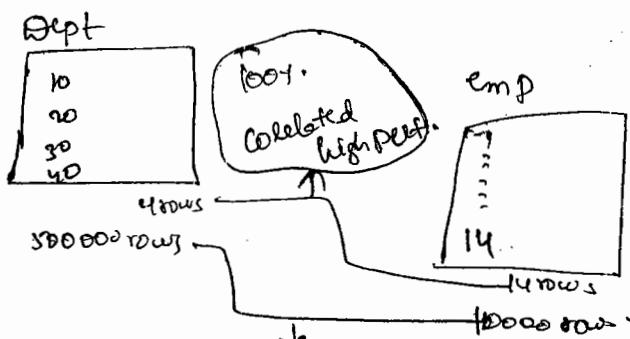
> Select * from dept where deptno not in (select deptno from emp);

Q.P. :-

40	Operations	Boston
----	------------	--------

→

co
of 40 → none
→ 40



This query is correlated also

Q.P. :- 40. bt Correlated
is very high perf.

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 Correlated 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);

40	Operations	Boston
----	------------	--------

Select

In where condition is null, is not null.

in correlated

related sub

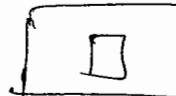
Index does not ^{Search scenes} working.

8-8-14.

bt Correlated
very high perf.

Correlated Sub Query:-

non-correlated



all
any.

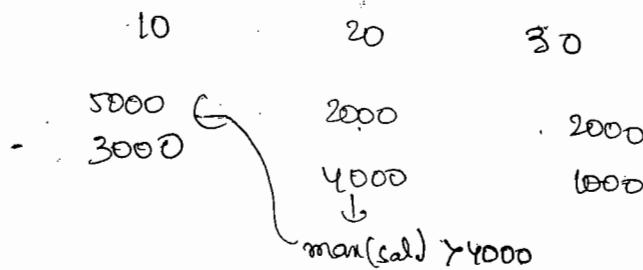
Correlated



exists.

all, any Subquery Special operators used in non correlated Subquery:-

➤ ~~WAP~~ to display the employees who are getting more than the highest paid employee in 20th dept. from emp table.



➤ Select * from emp where Sal > (Select max(sal) from emp where deptno = 20) O.P.

WAP 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. \equiv Grows

— 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 retrieve data from db.

This process automatically degrades performance of the application.

To overcome this problem for improve performance of a query, ansi/iso sql introduced Subquery special operators. there are "all, any".

These operators are used along with relational opr's ($<$, $>$, $=$, \neq , \leq , \geq)

➤ Select * from emp where

min/max = all.
max/min = any

Sal > all (select Sal from emp where
Deptno=20);

➤ Select * from emp where

Sal > any (select Sal from emp where
Deptno=10);

amount

2 or min

using

clauses very

performance

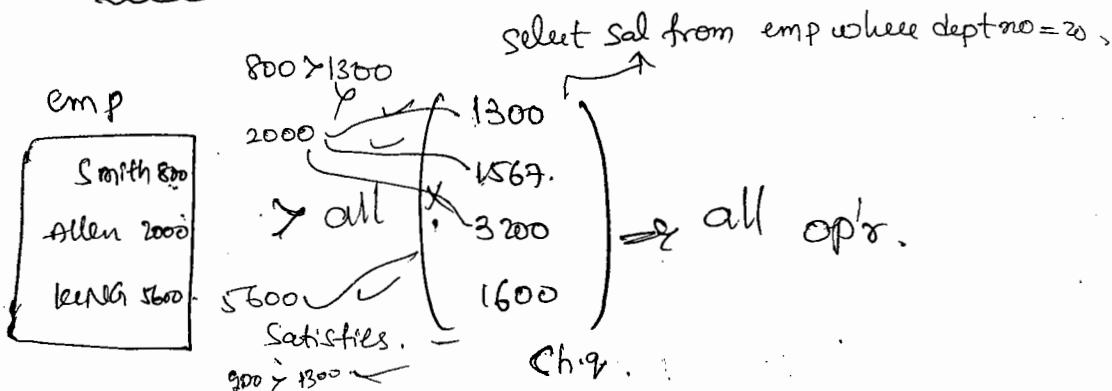
ie preformance

very special

with

all, any operators and in multiple row

Sub queries :-



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

All (\geq) :- It satisfies all values in the list

Any ($>$) :- It satisfies any value in the list

more preformance than 'In' \leftarrow = Any is nothing but internally In op's

Q10 To display the employees who are getting more than the all salaries of the 'clerks' from emp table using subquery special operator (All)

Order by clause \Rightarrow not in child query It won't work
that's why we use inline views.

after child query possible
closing

\Rightarrow Select * from emp where sal > all (select sal from emp where job = 'clerk') order by sal desc.

O.P :- 3100
3200
5000

without
O.P 3200
5000
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 oprs "OR"

All = AND
Any = OR.

Ex:- Select * from emp where

deptno > all (10,20);

O.P:- 30.

① deptno > 10 and

deptno > 20.

→ Select * from emp where

deptno > any (10,20);

O.P:-
20.
30.

② deptno > 10 or
deptno > 20.

③

Note:- "any" is also same as "in" operator.

Bt "any" not same as not in operator.

any = in
any ≠ not in

Pa
not
in

real opr
 ! opr "AND"
 Y Select * from emp where deptno not in (10,20); \leftrightarrow any
O.P :- 30.

real opr
 ! opr "OR"
 Y Select * from emp where deptno
 \leftrightarrow any (10,20);
O.P :- 10,20,30

\leftrightarrow any
 not equal or
 $\neq 10$ means - 20,30
 $\neq 20$ & - 10,30
 SD 10,20,30

Views.

and
 Table, cluster, d.b. objects.

View is a 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 "Windows of table".

user 1				user 2	
emp					
a	b	c	d		

Particular col. only give to another user not all cols. Then restrict cols.

From one user to another user we use views

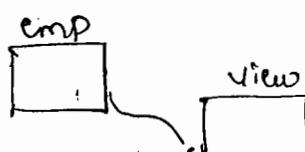
→ Generally if we want restrict table columns from 1 use to another use also we are using 'Views'.

→ Based on the base tables views are categorised into 2 views.

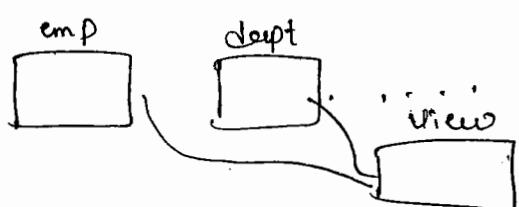
(1) Simple View

(2) Complex View. (or) Join View.

→ Simple View is a view which is created from only 1 base table.



→ Complex View is a view which is created from no. of base tables.



Simple View:-

Syntax:-

Create or replace view Viewname as

Select Stmt;

e Colms

are using

Categorised

(Select stmts are internally stored)
~~view~~-views

9-8-14

- In all d.b. Systems when we are creating views then automatically view definitions are permanently stored 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 bat.

- ↳ Create a view.
- ↳ Create or replace view V1

as

`Select * from emp where
deptno=10;`

- ↳ select * from V1;

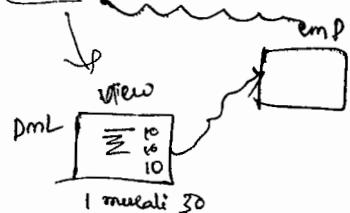
DML Operations on Simple View (to base-table):-

Rule 1: If a Simple View contains group fun's, group by clause, rownum, distinct, set operators, Joins then we cannot perform DML op's 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.

Note
or e
an o

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)

effect in
base table
Here for V_1

Values (1, 'murali', 30);

1 row created.

➤ Select * from emp;

We can not see
this record bcz

In where condn deptno
is '10' so

We can see in base
table (emp) only

This 1 murali 30 is available.

Example 2:

➤ Create or replace view V_2

as

Select * from ename, sal, deptno

from where deptno=10;

➤ Select * from V_2 ;

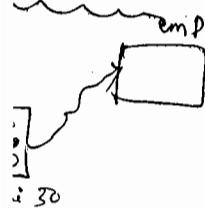
~~there is no~~
emp no.

➤ Insert into V_2 (ename, sal, deptno)

Values ('abc', 2000, 20);

Error: Cannot insert Null into empno.

if null
in perform
x-table.



Notes: 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;

Deptno maximum

10 5600.
20 3200
30 1525

or

→ alias name creation is 2 types

① with or

② after select of view create alias name

like V3 (deptno, maximum)

is no
empno.

→ Create or replace view V3 (deptno, max(sal))

as

Select deptno, max(sal) from emp

group by deptno; OP same

→ desc V3;
Name
Deptno
max(sal)

→ Rec
→

→ Create or replace view V4

as

Select rownum num, ename from emp;
alias

then

→ Select * from V4 where num > 5;

O.P.	rownum	ename
6	1	
7	1	
8	1	
9	1	
10	1	
11	1	
12	1	
13	1	
14	1	
15	1	parimali

W.A.Q to create a view from emp table to display each deptmt wise total salary (sum) from emp, dept tables & also display alias names as dname, total salary.

Ex:

→ Create or replace view V5 (dname, totalsal)

→ test

as

Select dname, sum(sal) from emp, dept d

error

where e.deptno = d.deptno

group by dname;

➤ Select * from V4;

DName	TotalSalary
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 can
those
Syntax:-

Ex:-
C

error

Sol:-
a

o

a

s

Warning

> create

> alter

> drop

to provide
so then we
check option
'Condition'

forced View or force View (Interview Purpose)
we can also create view without base table
those views are also called as forced views

Syntax:-

Create or replace force view Viewname
as
Select * from any name;

Ex:-

Create or replace View V₁
as
Select * from Sunday;

Err:-

table or view does not exist.

Sol:-

Create or replace^{force} View V₁
as
Select * from Sunday;

Warning:- View created with compilation errors.

- Y create table Sunday (sno number(10));
- Y alter View V₁ Compile;
- Y desc V₁;

Learn

11-8-14.

To

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';

↑ now updated.

➤ update V5 Set dname='xyz' where dname='SALES';

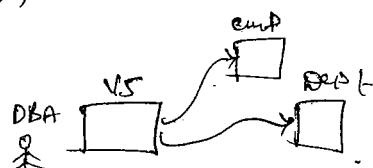
Error:-

In all d.b. Systems generally we can not perform

SQL operations through Complex view to Base tables

i.e., Here Some table cols are effected some

Other table cols are not effected.



mon

"ex

Ex:-

➤ S

-ble.

TR

Trig

pt

oprtn

To overcome this probm oracle introduced
"Instead of triggers".

By default these triggers are low level
triggers.

In oracle if u want to view effectable or
non effectable cols then we are using
"user-updatable-columns" data dictionary.

Ex:- DESC user-updatable-columns;

>Select COLUMN-NAME, UPDATABLE from user-updatable-
columns where TABLE-NAME = 'W6';

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 Statement level Triggers :- Trigger body is executed only 1st per DML Starts.



② row level Triggers :- Trigger body is executed for each row per DML Starts.



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; } compulsory

Difference b/w Stmt level, row level triggers :-

Ex:- row level Triggers :-

>Create table test (col1 datatype);

triggers

Start 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 update on emp for each row

begin insert into test values (sysdate);

end;

/

say.

triggers :-

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 trigger

Trigger body is executed for each row for

DML Stmt - That why we are using for each row

class in trigger specification.

And also DML transaction values are automatically stored in two rollback start qualities there are "old, new".

These buffers are used in either in triggers specification or in trigger body.

When we are using this qualities in trigger body then we must use ';' in front of the

we

Sy

:ol

Sy

:n

⋮
⋮

WA

when

auto,

Ex:

Y

Y

Y

Y

Y

Y

we must use `:` in front of the qualities name

Syntax :-

:old . column name

Syntax :-

:new . column name

	insert	update	delete
:new	✓	✓	✗
:old	✗	✓	✓

WFA 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

Y Create table backup as
Select * from emp where 1=2;

Y Select * from backup;
no rows Selected.

Y desc backup;
Y Create or replace trigger thi
after delete on emp

for each row begin

| |
| | merge

Create or replace trigger th,

after delete on emp

for each row

begin

insert into backup

values (:old.empno, :old.ename, ?old.job, ?old.sal,

----- :old.deptno);

end;

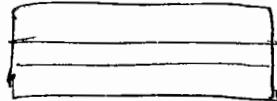
/

testing:-

Y delete from emp where sal > 2000;

Y select * from backup;

rows



Instead of triggers

generally we can not perform DML operations

through Complex views to base table.

To overcome this problem Oracle 8.0

introduced instead of triggers.

Instead of triggers are created on views.

→ By default instead of triggers are low level triggers.

Syntax :-

create or replace trigger trigger name

instead of insert / update / delete on view name

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 -

on views,

triggers.

> desc user_updatable_Columns;

Y Select COLUMN_NAME, UPDATABLE from

user_updatable_Columns

where TABLE_NAME = 'V5';

COLUMN_NAME

ENAME

SAL

DNAME

LOC

UPDATABLE

YES.

YES.

YES.

YES.

Materialized Views :-

Oracle 8i introduced materialized views these views are handle by db. administrator.

materialized views are used in d.b.t.h. applications.

generally views does not 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 views or Synchronize
data based on "base table".
mat'zed view only Administered

Syntax:-

```
Create materialized View Viewname  
as  
Select Statement ;
```

* Before we are creating materialized view DBA's
must give create any materialized view privilege to
(Permission)
user.

Syntax:-

```
grant create any materialized View to Username ;
```

then views

Ex:- Create materialized View mv1 ,
as
Select * from emp;

note view creates
only Administered
so when we create
error occurs
so we will use
priv to grant

> Conn Sys as sysdba;
Enter password: sys
> Conn scott / tiger;
> Create materialized View mv1 ,
as
Select * from emp;

```
Materialized View Created.
```

1.6.11.

state where

are used to

negotiable

exp,
the remote

Note: In oracle sometimes 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;

Y create or replace view v1

as

select * from emp;

error: insufficient privileges.

Y Conn sys as sysdba;

Enter Pwd : sys

Y grant create any view to scott;

Y Conn scott/tiger;

Y create or replace view v1,

as

select * from emp;

View created

Notes In oracle 1 of the materialized base table must contain primary key otherwise oracle server returns an error.

Ex:- Create table test (sno number(10));

>Create materialized view mn,
as

Select * from test;

Error:- table 'test' does not contain a primary key

Constraint

Diff b/w View, materialized view:-

>Create table base (sno number(10) primary key,
name Vaishnav(10));

Insert into ^{base} values (.....);

Select * from base;

Create view

SNO	NAME
1	a
2	b
3	c
4	d

Create or replace view v1

as

Select * from base;

View created.

Create mat'd view.

➤ Create materialized view mK1
as

Select * from book;

⇒ In this case materialized view is also same as
the
view.

— But when ever we are creating mat'd view also
mat'd view definitions automatically stored in dbase
same like a view definitions.

— For oracle if u want to view these materialized view
definitions then we are using user_mviews data dictionary

➤ desc user_mviews;

➤ Select query from

user_mviews where

VIEWNAME = 'mK1';

➤ Select rowid, sno, name from book.

rowid sno name

➤ Select rowid, sno, name from V1;

For this case View rowid's all same as book table
rowid's

— That's why view does not store data

— Through the view we can alter book-table data

That's why Views are also called as
"Virtual tables".

➤ Select rowid, Sno, Sname from mki;

→ Same as
Here matchd view rowid's are diff from base table
rowid's.

→ View also
That's why mate'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 mki;

SNO	Name
1	a
2	b
3	c
4	d

If it is not effected

→ Mate'd View also stores data same like a-table
but when we are refreshing mate'd view it
synchronizes data based on base-tables.

→ In oracle If u want to refresh matchd view then
we are using refresh method from

dbms_mview package.

Syntax:

dbms_mview.refresh('materialized Viewname');

method name

Package name:

exec dbms_mview.refresh('mri');

SNO	Param
1	A
2	B
3	C
4	D

Ex:

1

2

②

13-8-14.

The

Oracle having 2 types of Mat'd Views:-

Oracle

(1) Complete refresh mat'd Views.

(2) Fast Refresh mat'd Views.

high

(1) Complete Refresh materialized Views:-

- In oracle by default mat'd views are complete refresh mat'd views

- When we are refreshing then mat'd views then refresh

internally rowid's are deleted if u are not modified
base table data also.

Syst

Syntax:

Create materialized View Viewname

Refresh complete

as

Select Statement;

13-8-14.

8m² > Select rowid, sno, name from mkt;

> exec

dbms_mview.refresh ('mkt');

> Select rowid, sno, name from mkt;

(Other rowids are changed)

② Fast refresh materialized 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. For this

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. \Rightarrow

Syntax:-

Create materialized view log on basetablename;

(1) M

- Example:-
 - Select * from base;
- 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

(2) Au

using

on con

Sy

mat'd view
bar tables

On demand, on Commit or (with out PL/SQL)

→ generally we are refreshing mat'd view in 2 days

(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:-

Create materialized view viewname

Refresh Complete / refresh fast ondemand / oncommit
as by default without writing is also by default.
Select Statement; no problem

>Create materialized view mks

refresh fast oncommit

as

Select * from bar;

Y Select * from bar;

1	xy
2	yy
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	yy
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

ACL (Data Control Language) :-

- (1) grant (giving permission).
 - (2) revoke (cancel permission).

Creating a User :-

> Conn Sys as Sysdba ; ← > Conn system / manager
Enter Psw: sys; ← (off)

Syntax :-

- ① Create user username identified by password;

Syntax

- ② Grant Connect, Resource to user name;
or
DBA

Syntaxe:

SQL> Conn Username / Password;

 conn sys as sysdba;

Enter psw1:sys.

➤ grant Connect, Resource to India;

→ Conn India / India
4.1.1 P.S.W

➤ Select * from emp; In India even there is no date.

Error: Table or View does not exists.

→ from scott@scott we transfer emp and all data to india@india

→ Conn scott/tiger;

→ grant all on emp to india;

→ Conn india/india;

→ Select * from emp;

error: table or view does not 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, ---

Syntax :-

grant System privileges to username1, username2, ...;

object).

oms having

Conn sys as sysdba;

Enter password : sys.

grant Create procedure Create any materialized view, Create any index to Scott, murali, India;

Note:-

In Oracle If we want to Connect any user to the Server then we are using "Create session privileges"

Create session is used to Connect to Server

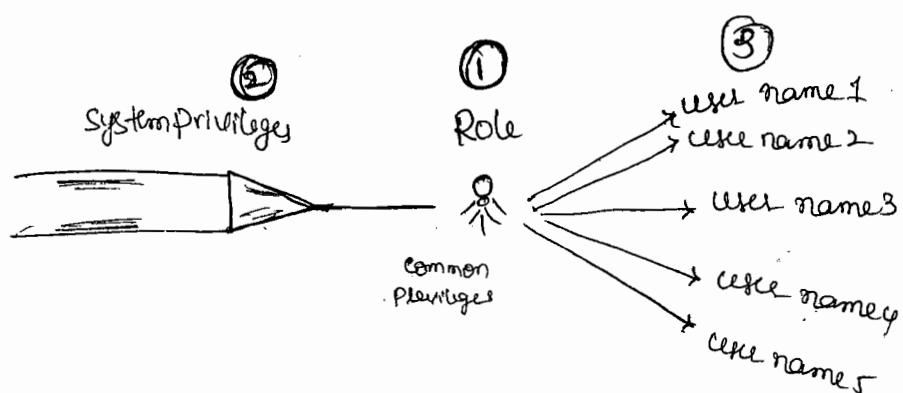
Role :-
privileges given by only admin user

Role is nothing but collection of either System or Collection of object privileges.

User +

- 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's creating a user defined Role & assigns common Set of privileges into Role & then only that role gives 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, ...)

9. > Conn sys as sysdba;

works on: Enter password — sys.

> create role ab;
 ^{filename}

7 (Same) > grant create procedure Create any materialized view,
 creating
 of
 users to
 Create trigger to ab;

> grant ab to

Scott, minali, Pndia;

⇒ 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 then
automatically 3 predefined roles are created.

These are (1) Connect designed for end user purpose
 (2) Resource → developer
 (3) DBA → database 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 material
 ↴
 generate
 ↴
 tables
 ↴
 d.d.
 create table

Object privileges:-

Object privileges are used to perform some operations of the object.

These privileges are given by either d.b. Developer or DBA.

Oracle having following object privileges
 these are {Insert, Delete, Update, Select}

These 4 are also called as all in oracle side
 execute, read, write, (PL/SQL 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 in roles)

who receives with grant option clause those users giving same object privileges 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 grant option

Notes:-

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-prive d.d. `SQL> desc user-tab-prive;`

Revoke:-

It is used to cancel either system or object privileges from the user

Syntax:-

`Revoke System privileges from username, username...;`

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
username, username...;`

➤ Create or replace view V_1

as

Select empno, ename, sal from emp

where deptno=10;

➤ Select * from V_1 ;

➤ Grant all on V_1 to india;

➤ Conn india/ india;

➤ Select * from scott. V_1 ;

→ We can also drop view using drop view view name;

→ When ever we are dropping a base table we can't access a view.

But view definitions are still exists in

d.b.

Ex- ➤ Create or replace view V_1 .

as

Select * from base;

➤ Select * from V_1 ;

➤ drop table base;

➤ desc user-views;

➤ Select Text from user-views where

VIEW-NAME = 'V1';

Text

- Select "SNO", "Name" from bat.

MERGE Stmt (AML) 9i

oracle 9i introduced "Merge Stmt".

- This Stmt is used to transfer data from source table into target table.
- But here these 2 tables structures must be same.
- merge Stmt used in data warehousing applications.
- In merge Stmt we are using update, insert Stmt that's why this Stmt is also called as :

"UPSERT" Stmt.

- In merge Stmt we must create different 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);
- Select * from dept;

10
20
30
40
- 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	Dname	Loc
10		
20		
30		
40		
1	a	b

If
then

Note:- In merge Stmt we can not update

On clause Columns:

d.deptno = s.deptno — error.

SEQUENCE (Independent Obj)

Syntax

→ Sequence is a d.b. object which is used to generate sequence no's automatically.

Syntax

→ generally sequences are used to generate primary key values automatically.

These

→ Sequence is an independent d.b. object once a sequence has been created no. of users simultaneously access that sequence.

Select

→ 1

using

Syntax

Create Sequence Sequence name

Start with n

increment by n

min value n

max value n

cycle / no cycle

cache / no cache;

]

Syntax

SQL

→

update

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 cols are used in insert, update, delete,
select statements (Ans)

Final-

→ 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 deal;

error

Sequence $s1.$ currval is not yet defined for this session.

➤ Select $s1.$ nextval from deal;

5

➤ Select $s1.$ nextval from deal;

7

— In all d.b.sys' if u want to generate 1st sequence no. then we must use next val pseudo column.

— Bcz currval pseudo column 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 deal;

Nextval	currval
5	5

Note :- If u want to view default values for a sequence then we are using user_sequences d.d.

Ex

➤

➤

➤

Not

only

➤

➤

➤

➤

= S

➤

Ex:- Create Sequence S₁

Start with 1;

➤ Select S₁. nextval from deal;

1.

➤ desc user_sequences;

➤ Delete 1 from user_sequences;

Note:- we can alter all sequence parameter values except "Start with"

1st sequence

columns:

1 value

by leaving

Syntax:-

alter sequence S₁

Start with 7

Increment by 1

min value 5

max value 20;

➤ Select S₁. nextval from deal;

7.

➤ Select S₁. nextval from deal;

8.

➤ alter sequence S₁

Incremented by -1;

= Sequence altered.

➤ Select S₁. nextval from deal;

7

→ alter sequence S_1

Start with 8;

error: Cannot alter starting sequence number.

20-8-14

Note: Start with cannot be less than min value.

→ Create sequence S_1

Start with 4

incremented by 1

minValue 5;

error: START WITH cannot be less than minValue

Cycle / no cycle:-

No cycle:-

→ Create sequence S_1

Start with 5

incremented by 1

minValue 4

maxValue 8

cycle

no cache;

→ Create sequence S_1

Start with 5

incremented by 1

minValue 4

maxValue 8,

no cycle ~~no cache~~

→ Select S_1 nextval from deal;

O.P:- 5.

O.P:- 5

6

7

8

→ 1 6 4 4
... 5 5
... 6 6
... 8 8
... 1 1

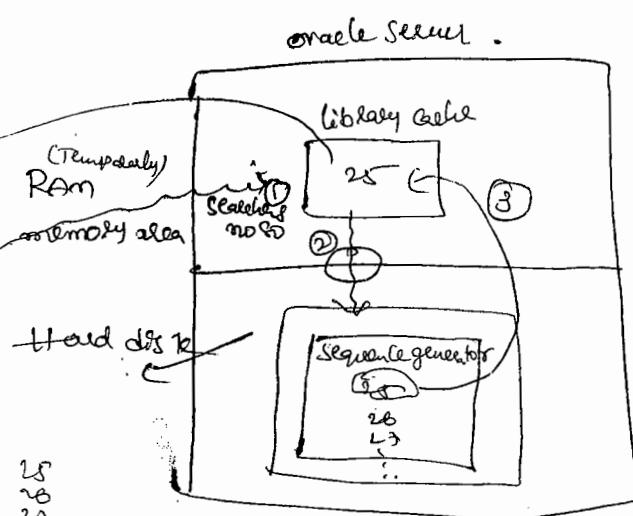
error

➤ Error: Sequence S_1. NEXTVAL exceeds MAXVALUE
and cannot be instantiated.

➤ Select S_1.nextval

from dual; ④
25.

Cache 5. → 25
26
27
28
29
30.



Cache Cache is a memory area which is used to access Sequence values very fastly.

→ generally if we want to improve 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 is fetching cache values from disk & then only those values are transferred into cache memory area.

→ This process automatically (reduces) degrades performance of the apps.

→ To overcome this problem we are defining set of sequence nos using cache option.

In oracle by default Cache values are 20 and also cache min value is 2.

Note :- When ever Server is shut down or system crashes then automatically Cache values are "lost".

> Create sequence sr

- 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

available

Ex:

Create table test (sno number (10) primary key,
name varchar(10));

- process

only those

area,

for performance

Set of

values 20

System

"losted"

e

values

only

>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 Statement only.

→ In Oracle Row Level Locks are introduced in Oracle 6.0.

Syntax:-

```
Select * from tablename  
Where Condition For update [nowait];  
↳ optional
```

→ rows are locked but it does not display any msgs but internally rows are locked. Bt Am2 operations working bcz authorised person allowed bcz unauthorised users can not perform Am2 oprns. Select ^{only} working.

⇒ when ever we are performing locks another user query (select stat) the data but they can not perform Am2 operations. & also

In all d.b. Systems when ever we are using Commit then only automatically locks are released.

Scott/Tiger

Unauthorised → Select * from emp where deptno = 10 for update;
...) → Commit; (or) → rollback;
↳ [For releasing locks].

meval/mevali

→ update Scott.emp set sal = sal + 100 where deptno = 10;
[We cannot perform DMLs]

21-8-14

by using

NoWait :-

Oracle 6.0. → 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.

3 Commit

meval/mevali

→ 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 db. developers uses 2 types of table level locks.

- (1) Share lock.
- (2) Exclusive lock.

(1) Share lock :- (more than 1 user)

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)

Syntax:- Still waiting all the users release the locks.

```
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 lock} in share mode; (✓)

> update scott.emp set sal = sal + 100;

— (we cannot perform DMLs)

(2)

whi

que

ope

Syn

[

lock

:

Not

then

Excl

> 1

er

wt

② Exclusive locks (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;

ie query operations

e)

lock table emp

in exclusive mode;

Scott/Tiger

meali/meali

> select * from Scott.emp(✓)

> lock table Scott.emp

In Share mode : (X)

Exclusive mode : (X) any locks are working
Stuckies to commit

Note: In oracle when ever we are using DML statements

then automatically Oracle Server Internal uses

Exclusive locks [some data performs then only exclusive locks applied]

Scott/Tiger

in same user

Scott/Tiger

> update emp set

ename = 'abc'

where ename = 'SMITH';

> 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 (dupl.) and also automatically avoids insertion, updation, deletion problems.
- In Design phase of the SDLC db. designer uses (1) E "normalization" process i.e.,
 - In Design phase of the SDLC D.b. designer designs logical model of the database.
 - In this logical model only designer 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 2nd Normal Forms.

Candidate
No

Normal Forms :-

- (1) first normal form
- (2) Second normal form
- (3) third normal form.
- (4) BCNF
- (5) Fourth normal form.
- (6) Fifth normal form.

(1) First Normal Form :- (Domain Key NF)

Item table. → multiple values

Item name	Color	price	tax.
marker	black, red	20	0.02
Pen	blue, green	30	0.03

Not in 1st NF

↓ 1NF

Item table.

Item name	Color	price	tax
marker	black	20	0.02
marker	red	20	0.02
Pen	blue	30	0.03
Pen	green	30	0.03

→ duplicates

Candidate key

→ A table is in 1NF in that table every cell having an atomic value and also.

② Identifying a record uniquely using a key

(newly married couple) (couple for electrical shop)
 (B. straw) (cool down)
 Order Form 1 F 2
 Order no

Order date

Customer name

Address

Ph. no

Item name

Amount

Submit

① Child Table

ordno	o.d.	cus name	add	ph	I	A

②

2nd Form

o.d.	o.d.	cus name	add	ph	I	A	I	A	I	A

not possible in d.b.

Order Form 3

Order no

Order date

Customer name

Address

Ph. no.

Item name

Refrigerator
LED TV

add another item

Amount

Submit

P.K.

Order master Table

Order no	Order date	Customer name	Address	Ph. no
1				

Order detailed Table

F.K.

1NF Table

Order no	Item name	Amount
1		
1		
1		

(child Table)

bcz duplicates are there

Every Cell

Process:-

Identifying repeating groups & putting into separate table in more atomic form.

2 Key

data table		
add	phy	I A

→ By default 1NF process table is a child table. Bcz in this table 1 col having duplicate data (must)

22-8-16.

2nd Normal Form:-

S2	A	I3	D3

possible in 1NF.

1.6.

20

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

1.6.

→ The Separate Table is called 2NF Table.

→ By default this is a master table. In this table only - all Nonkey attributes are Fully Functionally dependent on total Candidate Key.

Diagram illustrating a 2NF table structure:

Table Structure:

Item name	Color	Price	Tax
marker	black	20	0.2
marker	Red	20	0.2
Pen	blue	30	0.3
Pen	green	30	0.3

Annotations:

- Candidate key:** Points to the primary key 'Item name'.
- key attributes:** Points to the columns 'Item name' and 'Color'.
- Non key attributes:** Points to the columns 'Price' and 'Tax'.
- ↑ P.** and **↑ P.** indicate that both 'Price' and 'Tax' are partially dependent on the primary key 'Item name'.
- Partial bc+ Price only**: Partially dependent on 'Item name' only, not 'Color'.
- Partial Tax also.**: Partially dependent on 'Tax' also.
- not P in 2NF**: Not a primary key in 2NF.

Table Structure:

Item name	Color
marker	black
marker	Red
Pen	blue
Pen	green

Annotations:

- F.K.**: Foreign Key, points to the 'Item name' column.

Table Structure:

Item name	Price	Tax
Marker	20	0.2
Pen	30	0.3

Annotations:

- P.K.**: Primary Key, points to the 'Item name' column.
- Partial to partial**: Partially dependent on both 'Price' and 'Tax'.

(master-table) 2NF.

Table Structure:

Stno	Stname
101	Shre
102	marker
103	poor
104	Tyo

Annotations:

- P.K.**: Primary Key, points to the 'Stno' column.

Table Structure:

Stno	Activity
101	Cricket
101	Football
102	Cricket
102	Hockey
103	Football
103	Swimming
104	Swimming

Annotations:

- F.K.**: Foreign Key, points to the 'Stno' column.

Table Structure:

Activity	Cost
Cricket	\$10
Football	\$20
Hockey	\$30
Swimming	\$40

Annotations:

- P.K.**: Primary Key, points to the 'Activity' column.
- N.K. attr.**: Non-key attributes.
- Partial key attr.**: Partially dependent on 'Activity'.

2NF (masterTable)

Student Sports Activity table.

ble
this table

smallly

buter

Price only

name only

at color fo

to. Partial

R

Partial
to
partial

1NF

Non
key
attri.

at key attri:

ble

Sno	Sname	Acti 1	Cost 1	Act 2	Cost 2
101	Shreethi	Cricket	\$10	Football	\$20
102	madhu	Cricket	\$10	Hockey	\$30
103	Pooja	Football	\$20	Swimming	\$40
104	Jyothsna	Swimming	\$40		



table

Sno	Sname
101	Shreethi
102	madhu
103	Pooja
104	Jyo

Sno	Acti 1	Cost 1	Act 2	Cost 2
101	Cricket	\$10	Football	\$20
102	Cricket	\$10	Hockey	\$30
103	Football	\$20	Swimming	\$40
104	Swimming	\$40		

Sno	Sname
101	Shreethi
102	madhu
103	Pooja
104	Jyo

Key by attri
other

Sno	Activity	Cost
101	Cricket	\$10
101	Football	\$20
102	Cricket	\$10
102	Hockey	\$30
103	Football	\$20
103	Swimming	\$40
104	Swimming	\$40

candidate key
↓ 1NF
P
Non key attribute
not in
2NF
depends on
activity

2NF

2NF

Convert to 2NF

key attribute
Candidate key

PK
PK
non-key attributes

③ F → F 10

F

Ecode	Proj Code	Dept only code	Dept head only code.	Hours Ecode + Proj code	F
E101	P1	Systems	D1	3	not F 10 2NF.
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	

insertion error

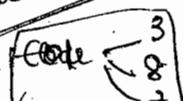
PK

Phase 1:-

FD

① Ecode \rightarrow hours (X)

functional dependency
removable



② Proj code \rightarrow hours (X)

hours

③ Ecode + proj code \rightarrow hours (✓)

insert

in

insert

meest

the

up

dept

part

Phase 2:-

FD

① Ecode \rightarrow Dept (✓)

E101 \rightarrow systems

② Proj code \rightarrow Dept (X)

③ F - F 10

Phase 3:-

not in
2NF.

① Ecode \rightarrow Depthead (✓)
FD

② Proj code \rightarrow Depthead (✗)

Ecode	Dept	Depthead
E101	Systems	D ₁
E102	Sales	D ₂
E103	Research	D ₃

2NF master table

Ecode	Proj code	Hours
E101	P ₁	3
E101	P ₂	8
E101	P ₃	7
E102	P ₁	5
E102	P ₃	10
E103	P ₂	9

* (child table)

23-8-11.

→ Before Normalization process above resource table having insertion, update, deletion problems

Insertion problems :-

For the above resource table when we are trying to insert particular department employee details then we meet assign project code. (1st day ^{working} we do not assign project) (80)

This is called insertion problem.

② update problem → In the above resource table Ecode, dept, depthead attribute values are repeated whenever particular deptmt employee transferred 1 dept in another deptmt

then we must modify all these 3 attribute values correctly. (Update sales dept to eng-~~dept~~).

→ otherwise inconsistency problem occurred. This is called "update problem".

③ Deletion problem: In the above resource table when we are trying to delete a particular employee then automatically deptmt details also deleted. This is called "deletion problem".

→ when ever we are using normalized process then automatically insertion, update, deletion problems are avoided (void).

3NF

→ If a table is in 2NF & also all Non-key attributes are only dependents on Candidate (P.k) key

→ If a table is in 2NF & also Non-key attributes which ~~also~~ depends on another Non-key attributes then that table "not in 3NF".

attribute

This is

table

employee then

is called

its items

elements

non-key

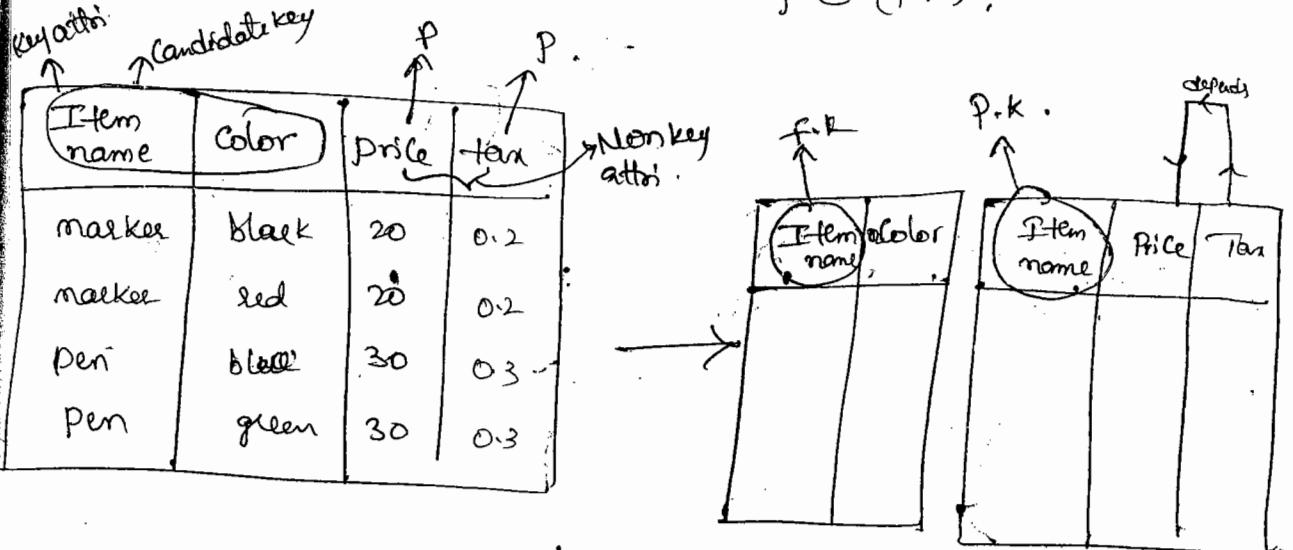
or
(P.K) key

attribute

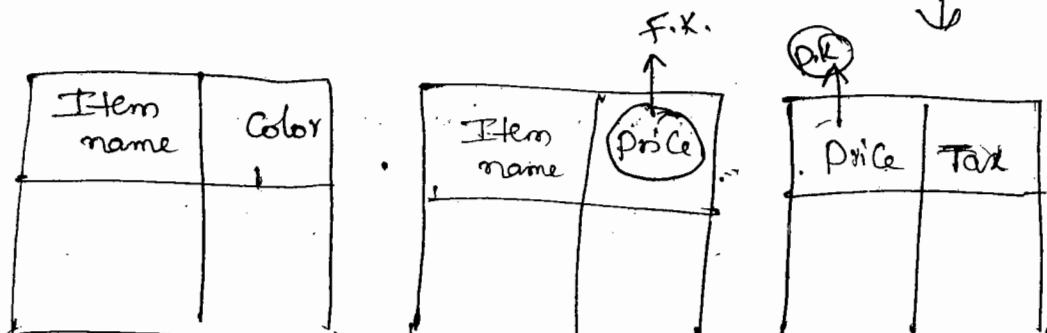
that

Process :-

- Identify Non key attributes, which depends on another Non-key attributes put into separate table.
- This table is called "3NF table". By default this table is master table.
- In this table only all non-key attributes are only dependents on Candidate key \oplus (P.K).



in 2NF master table
but not in 3NF



master table (3NF table).

Order Form.

Order no

Order date

Cust no

Qty Discount

Item name Item no

Amount add another item

Customer Enquiry Form

cust.no	<input type="text"/>
Cust.name	<input type="text"/>
address	<input type="text"/>
Ph.no	<input type="text"/>
<input type="button" value="Submit"/>	

Key α
S

P.K customer master table

(3NF)

Custno	Cust name	Address	Ph.no

P.K order master table

(1C)

Orderno	Orderdate	Custno

Item master

Item no	Item name	Amount

order detailed master table

Order no	Item no.	Qty	Discount

Form

key attributes → Candidate key

25-8-14.

2NF

Not in 3NF (N-key FD on N-key)

Non-key attributes

Stname	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.

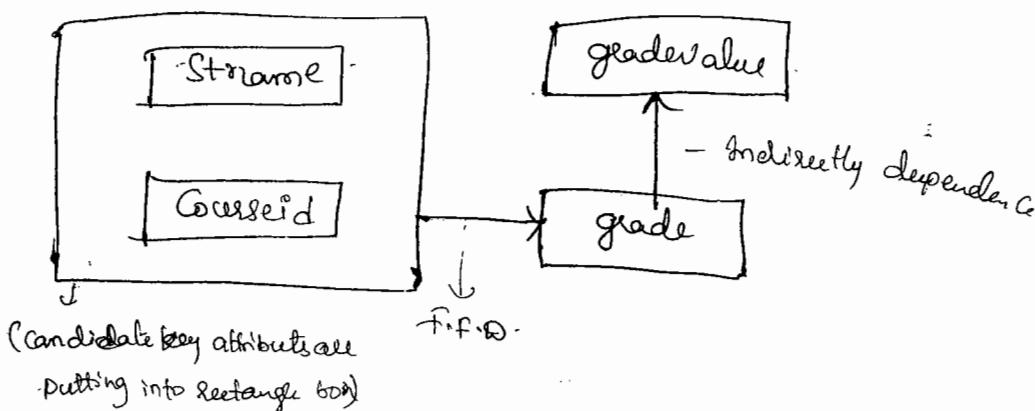
Child table

Master table

3NF

Stname	Courseid	Grade	Gradevalue
abc	CS111	A	4.00
xyz	CS111	B	3.00
zzz	CS222	C	2.00
aaa	CS222	A	4.00

logical diagram.



Functional Dependency :- (Uniqueness) :- (FD) :-

If any given 2 tuples in a relation 'x' 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 $(x \rightarrow y)$

attribute		
Stno	Sname	Hall
t_1 $\rightarrow S_1$	abc	H1
t_1 $\rightarrow S_2$	xyz	H1
S_3	aaa	H2

Relation 'x'.

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.

en
UNF
- Cowl
- Gowl
- tutu
- tur
- Stm
- Stm
- @.o
- Gowl
- laste

1st & 2nd & 3rd NF.

Course id []

Course name []

Tutor id []

Tutor name []

Stno	Sname	D.O.B	Gender	Last attendence
[]	[]	[]	[]	[]

en

UNF

- CourseId
- Course name
- Tutor id
- Tutor name
- Stno
- Sname
- D.O.B
- Gender
- last.attende

1NF

Course id
Course name

Tutor name

1NF
not in
2NF

Course id
Stno
Sname+P
D.O.B.+P
Gender+P
last.attende+P

Some not in 3NF

- Courseid
- Course name
- Tutorid (M.R)
- Tutor name (M.A)

2NF

Stno
Sname
D.O.B
Gender

Courseid
Stno
last.attende

ident

try >

Course id
Course name
Tutor id

3NF
Tutor id
Tutor name

2NF
Stno
Sname
D.O.B.
Gender
last.attende

Cluster :-

Cluster is a d.b. object which contains group of tables together & also cluster 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. servers checks From clause tables are available in clusters or not.

→ If those tables are available in cluster then d.b. servers very fastly derive data from cluster tables.

→ This process automatically improves performance of the application.

Cre

Step

Step

Sy

Step 2

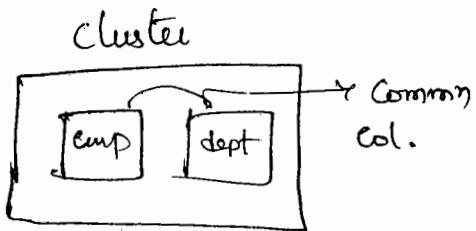
sy

C

Synt

Step 3

C



group of

2 data

1. Use

2. Column

key.

we all

in d.b.

table

2. Then

in cluster

name

→ Select dname, ename, sal, d.deptno, d.dname, loc

- 1 From emp e, dept d
- 2 where e.deptno = d.deptno; → Join
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.deptno (size));`

Step 2) Create an index on cluster:-

Syntax:

`Create index indexname on cluster clustername;`

Syntax: Create actual table:-

Step 3:-

`Create table tablename (col1 d.deptno (size), col2 d.deptno (size)...),
cluster
clustername (Common Colname);`

➤ Create cluster

emp-dept (deptno number (10));

➤ Create index idx 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, 'S/w', 'americpet');

➤ Select * from emp1;

➤ Select * from dept1;

➤ Select rowid from emp1;

➤ Select rowid from dept1;

Note:- Generally two rowid's are not same in table

-Bt cluster tables having same rowid's.

No

cl

inclus

table

Synt

Ex

er

89 or

Ex

dict

Ex

su

Sup

whic

Sup

Note:- Generally we cannot drop cluster if cluster having tables.

→ To overcome 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" dictionary.

Ex:- desc user-clusters;

Super key, Candidate key :-

Super key :- A column or combination of columns which uniquely identifying a record \downarrow is called "Super key".
in a table

Candidate Key

A minimal Super key which uniquely identifying a record in a table is called "Candidate Key".

Note: A Super key which subset of another S.Key (Combination) is not a Candidate key.

empno	ename	skillid	SKPL	voteid
1	murali	1	Oracle	v1
1	murali	2	Sybase	v1
1	murali	3	Ingresa	v1
2	abc	4	DB2	v2
2	abc	1	Oracle	v2
3	xyz	5	Informix	v3
4	zzz	.	.	v4
5	aaa	6	SQLserver	v5
5	aaa	4	DB2	v5

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

this

Can

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 candidate key nonkey attribute put into separate Table this table is called "BCNF Table".

In this table only Every determinant is a candidate key.

BCNF table does not have nonkey attributes

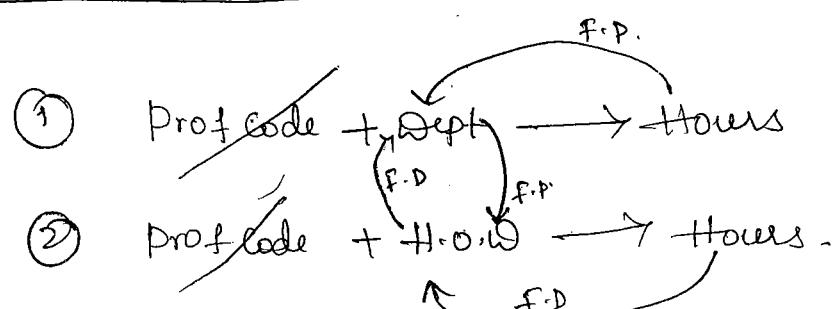
→ Generally if a table contains only 1 Candidate key then BCNF is also same as 3NF.

→ Generally when a table contains multiple Candidate keys then deletion problem occurs.

→ To avoid deletion problem d.b. designer uses BCNF process.

→ Generally in BCNF we are specifying relations ship b/w Candidate keys itself.

Prof Code	Dept	H.O.D	Hours	Non key attributes.
P ₁	Physics	H ₁	3	
P ₁	Computer	H ₂	5	
P ₂	Chemistry	H ₃	7	
P ₂	Botany	H ₄	9	



1 Candidate

1 ple Candidate

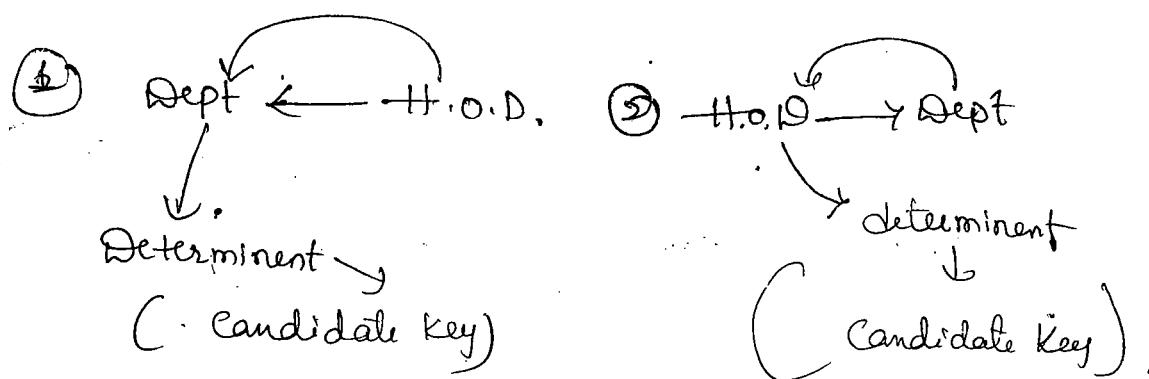
is BCNF

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

relation

non key
Attributes.



27-8-11.

4NF

4NF:-

If a table is in 4NF, then that table does not contain more than one independent multi-valued attribute.

→ If a table contains more than two attributes & also identifying a record uniquely using combination of all these attributes.

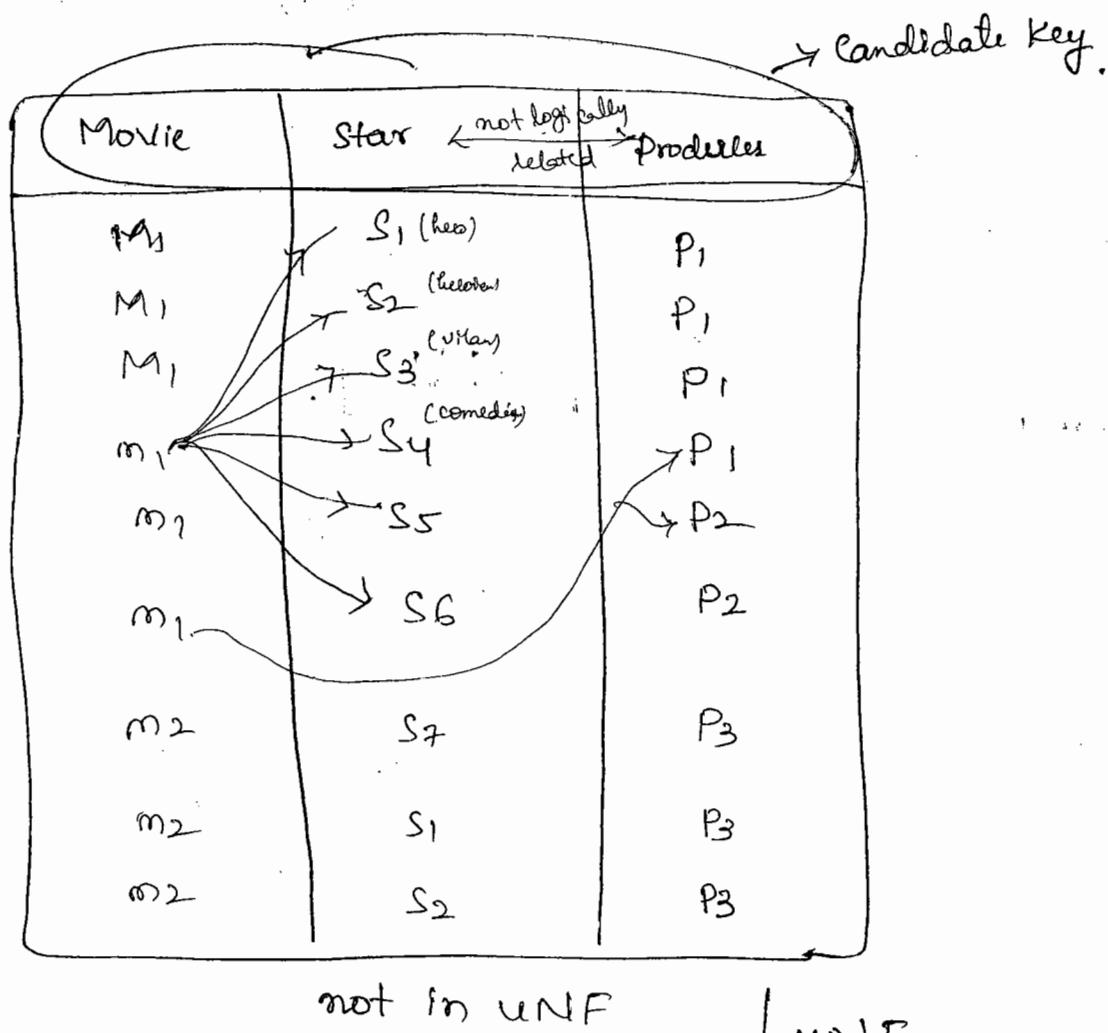
→ & also if attribute set of values which depends on another attribute set of values & also same attribute set of values which depends on another set of values & also some attributes are not logically related then only we are using 4NF.

For 4th NF recall table also there is no non-key attributes.

A
~
proj

Process:-

- Identify a column which depends on another independent multi-valued attribute column. Put onto separate table.
- These tables are called 4NF tables



Movie | Star

U3P

movie | producer

U4F

for
→ 4
then
in

on-key

Assumption:- Each employee having multiple projects & each employee having multiple skills.

Empid	Projectid	Skills	Key
1111	Railway reservation	JSP	
1111	Railway reservation	Asp.net	
1111	Railway reservation	oracle	
1111	Library management	php	
1111	Library management	SQLserver	
1111	SQL	Null	
1111	Null	SAP	

(not in 4NF)

if we insert
null value problem
will occur,

↓ 4NF

Empid.	projectid
1111	Rail.Ren
1111	Librarian
1111	SQL

problem

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 resource table.

→ & also when ever an employee got a new skills then we need to supply null values for the project in the above resource table.

→ These problems are automatically avoided when we are using 4NF tables.

Projection Joined NF (5NF):

→ If a table is in 5NF there are no ① Cyclic dependencies

→ when a table contains more than 2 attributes and also identifying a record uniquely using combination of all attributes (or 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 respective table.

Note:-

→ In 4NF Some field are not logically related whereas in 5NF all fields are logically related

Buyer	Itemname	Company
B1	Shirt	Pepe
B1	Tshirt	levis
B1	Jeans	denim

not in 5NF

Buyer table

when

Buyer	Itemname
Itemname	Company
Buyer	Company

0

SET OPERATORS

→ SET operators are used to retrieve data from
single or multiple tables.

→ These operators are also called "Vertical Joins".

These 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 (male) (in) (minus)

Select job from emp where deptno=20;

Note:- whenever we are using set operators always corresponding expressions must belongs to same data type

& also set operator between 1st query column names as col headings.

→ we can also retrieve data from set operators if corresponding expressions not belongs to same data type also, in this case we must use appropriate type conversion func.

Ex Select deptno "deptno", to-char(mvll) "deptname"
from emp , union

select to-number (mvll), dname from dept.

deptno	dname
10	
20	
30	

Account
Fleck
Allen

Conversions:- Converting 1 dty. into another

dty. is called "Conversions".

28-8-14.

relations

to same 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 Server only automatically converts 1 d.t.y. into another d.t.y.

Note 1:

In Oracle when ever string representing pure number then Oracle Server automatically converts string type to no-type.

Ex: Select Sal + '100' from emp;

Sal + 100 \downarrow
Convert into pure no.

2) Note 2: In Oracle when ever we are putting nos into character fun's (length, upper, lower, initcap) then Oracle Server automatically converts no-type into character type.

Ex: Select length(1111) from ideal;

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 format.

Ex :- Select last_day ('17-dec-05') from dual.

Out :- 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 developer only explicitly convert 1 d.t.y. into another d.t.y. using Sole

Explicit Conversion fun's.

wing

value

pe but

to format

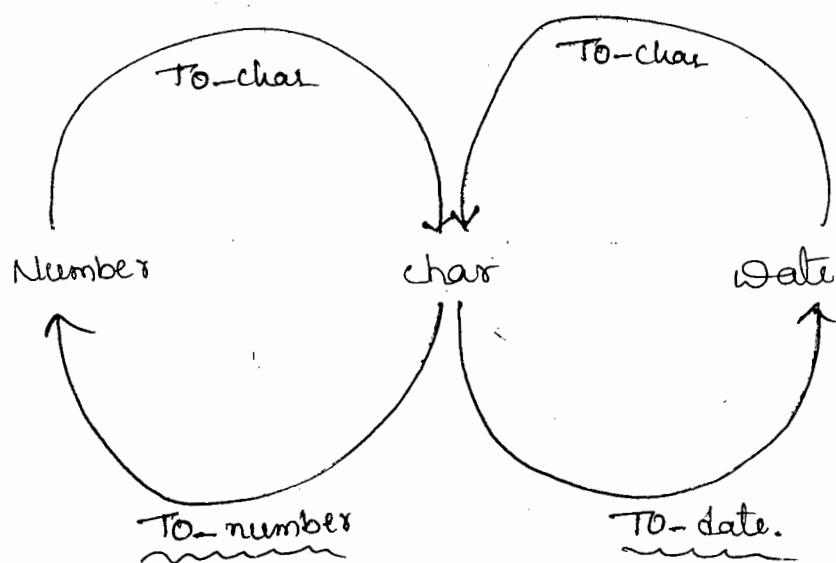
l

→ 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

Output: 488, error: invalid no.

➤ Select

to-number ('\$45.6', '\$99.9') + 3 from dual;

O.P:- 48.6.

On 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 :-

(.) g → group separator.

(.) d → decimal indicator.

→ Select to-char(20000, '9999') from dual.

O.P:- ######

→ Select to-char(20000, '99999') from dual

20000.

→ Select to-char(20000, '999.99999999d9') from dual

20000.00000000

2,00,00.00

To-

→ Select to-char(20000, '\$999999d9') from dual

20000.00

→ Select

Sal, to-char (sal, 'L999,999', 'NLS_CURRENCY=RS')

from emp

Sal

800

1600

1250

TO-CHAR

RS 800

RS 1,600

RS 1,250.

→ Select

Sal, to-char (sal, 'L999,999') from emp.

SAL

TO-CHAR.

800

\$ 800

}
L-local currency

1600

\$ 1,600.

1250

\$ 1,250

→ Select

ename, nvl (to-char (comm), 'no commission')

O.P:

Comm

2000

from emp

no commission (null replaced by)

no commission

\$ 00
200

→ Select

ename, nvl (to-char (mgr), 'no manager') from emp;

→ Select

to-char (sysdate, 'DD/MM/YYYY') from deal

28/08/2014.

To-date: It is used to convert date stored into date type

→ Select

to_date ('16/June/05') from deal;

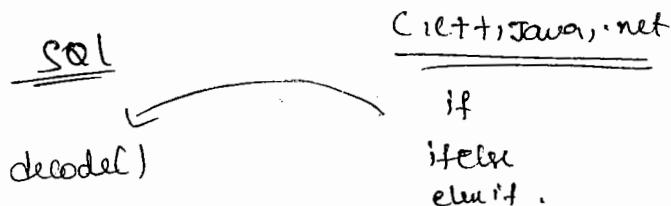
16-JUN-05.

decode(): (Conversion fun) :-

decode() is a conversion fun which is used to convert 1 d.t.y. into another d.t.y.

→ It is used to convert m.type into String type & also converts single character into string.

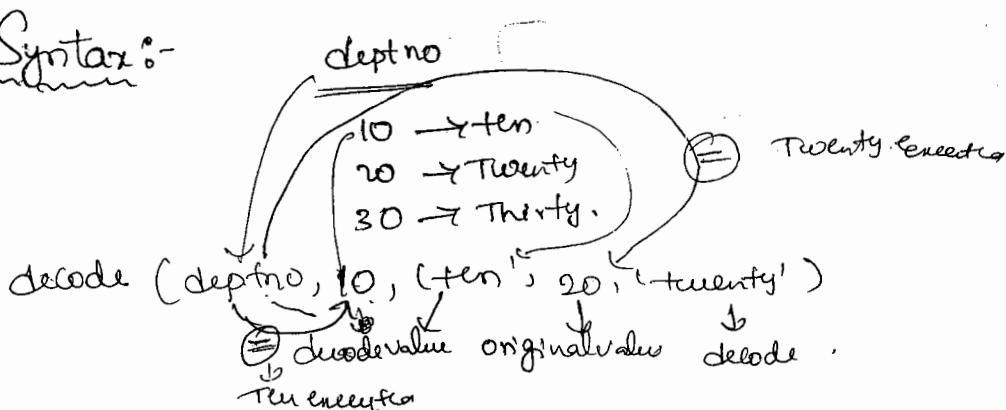
'a' → 'ab' 123 → '(123)'.



It is also same as `if -- then -- elseif` construct in PL/SQL.

→ decode() internally uses equality operator (=)

Syntax:-



`decode (Columnname, Value1, Stmt1, Value2, Stmt2, ... Stmtn)`

➤ Select ename, sal, deptno, decode (deptno, 10, 'ten', 20, 'Twenty', 'Others') from emp

29-8-14.

to connect.

Note:- We can also use decode() for modifying data in a table conditionally. In this case, we are using

type &

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 \rightarrow 10% of sal.
- (2) if job = 'SALESMAN' then Comm \rightarrow 20% of sal.
- (3) if job = 'ANALYST' then Comm \rightarrow 30% of salary.
- (4) else Comm \rightarrow 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:-	<u>JOB</u>	<u>Deptno 10</u>	<u>Deptno 20</u>	<u>Deptno 30</u>
20,	Analyst		6000	
	Clerk	1300	1900	980
	Manager	2450	2975	2880
	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)) "Clerk", sum(decode(job, 'SALESMAN', 1)) "Salesman", sum(decode(job, 'ANALYST', 1)) "Analyst", where e.deptno = d.deptno group by dname;

	clerk	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 advantages

Note:- decode() internally uses equality(=) operator where as in case Stmt we can also use all SQL operators explicitly.

for
also
of
one
to

"clerk",

onwards)

to

changes
in can

Method 1 (B.O):- Syntax:-

case column name
when value1 then stmt1
when value2 then stmt2

else stmt end

Method (2):- Case Conditional Stmt (Oracle 8i):-

Syntax:- Case

when condition1 then stmt1
when cond2 then stmt2

else stmts end

Ex:- > Select ename, sal, deptno
case deptno
when 10 then ('ten')
when 20 then ('twenty')
else 'Others' end
from emp;

Ex:- Select ename, sal, case when sal
1000 then 'low salary' when sal
bw 1000 and 2000 then 'medium sal'
when sal in (2300, 2500, 2800) 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).

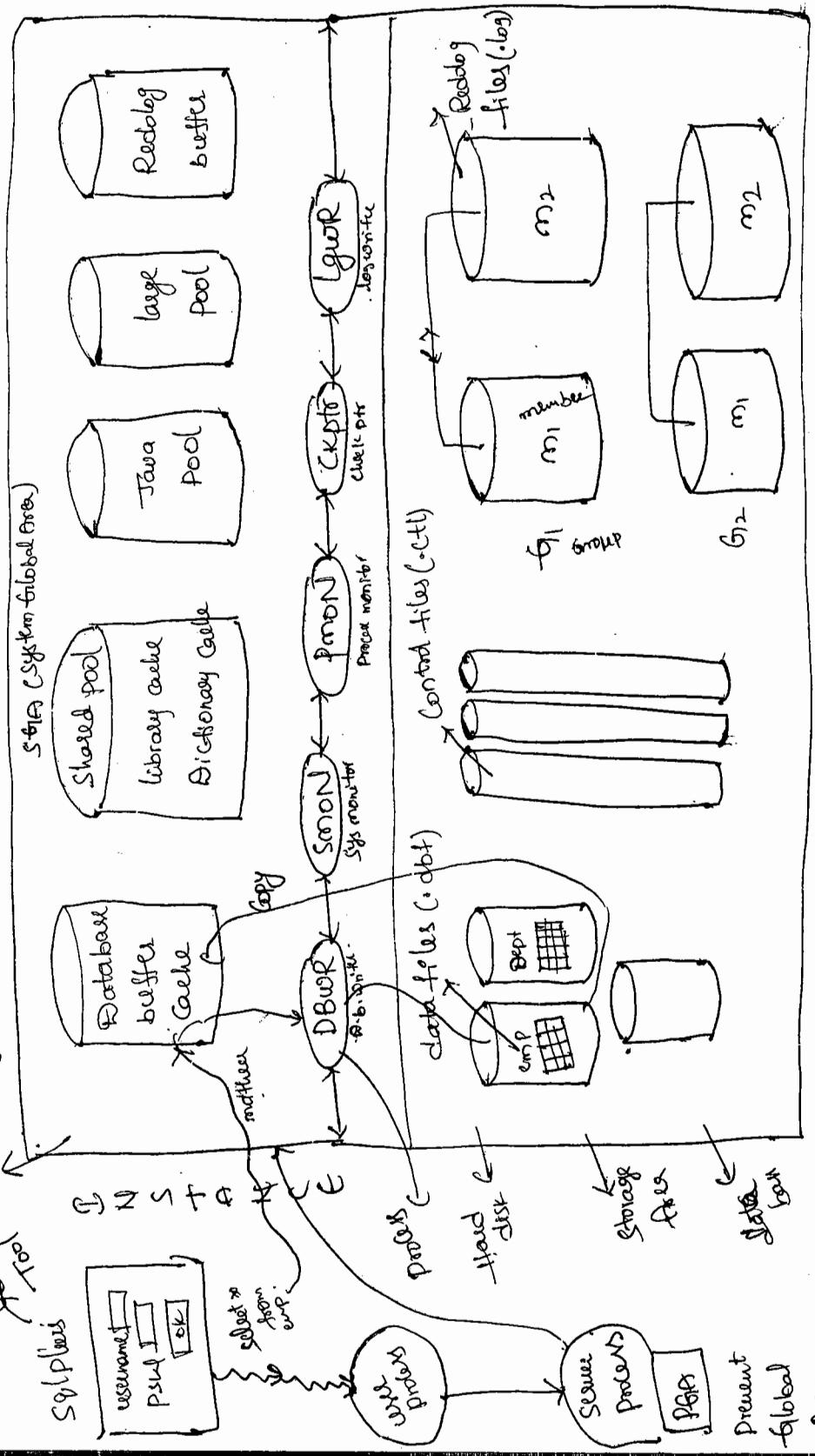
(2) Control files (.ctl).

(3) Redolog files (.log).

These 3 types of files are also called as "Storage Area"
or physical database.

Oracle Server Architektur.

Oracle Sequel.



- Data files stores all database objects physically
- Data files extension is .dbf
- In oracle if we want to view path of the data files then we are using "dba_data_files" date dictionary

Ex:

> Conn sys as sysdba;

Enter 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" date dictionary.

Ex: > Conn sys as sysdba;

Enter 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 Redolog files information stored under "V\$logfile" date 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-GA.

→ SGA Contains Set of buffers. These are
database buffer cache, shared pool, Java pool, large
pool, Redolog buffer.

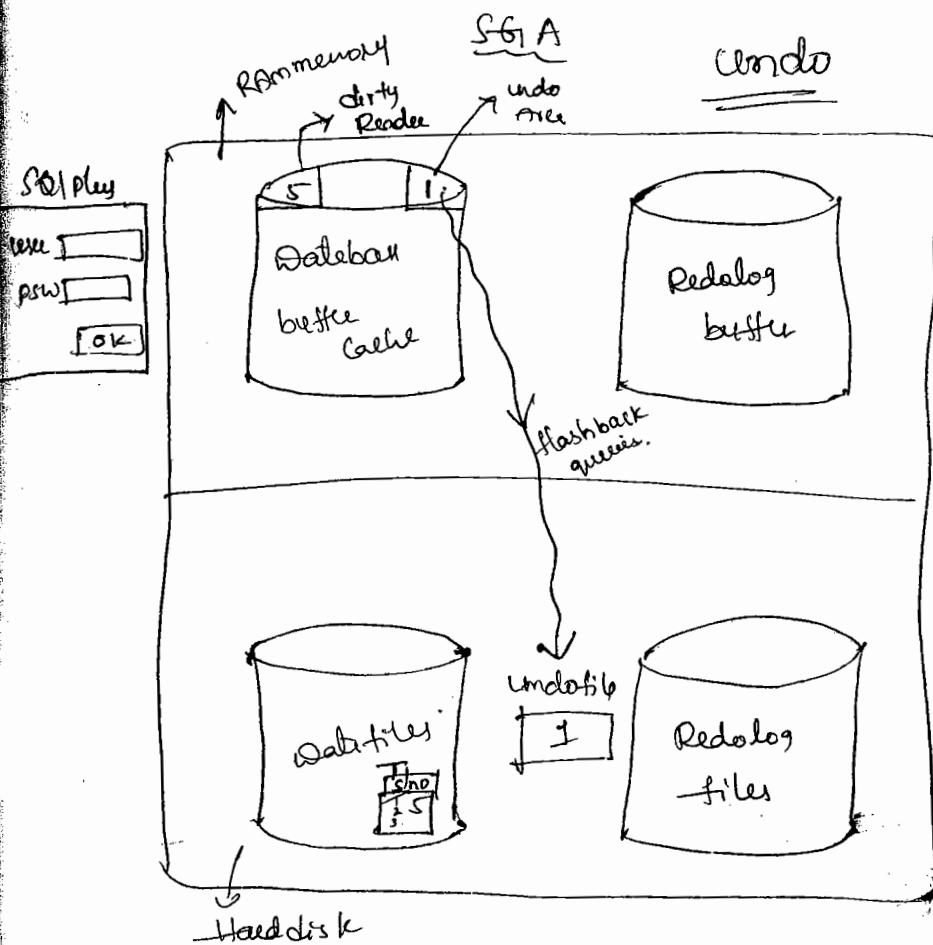
→ when ever we are submitting SQL, PL/SQL Code
that Code is stored in library cache within SGA
pool.

→ Library Cache always checks Syntaxes Semantics

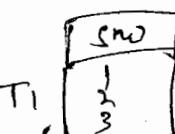
→ Library Cache reduces parsing by using parsing
algorithms.

→ 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 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 T_1 set $Sno=3$ where $Sno=1$,
- Commit;

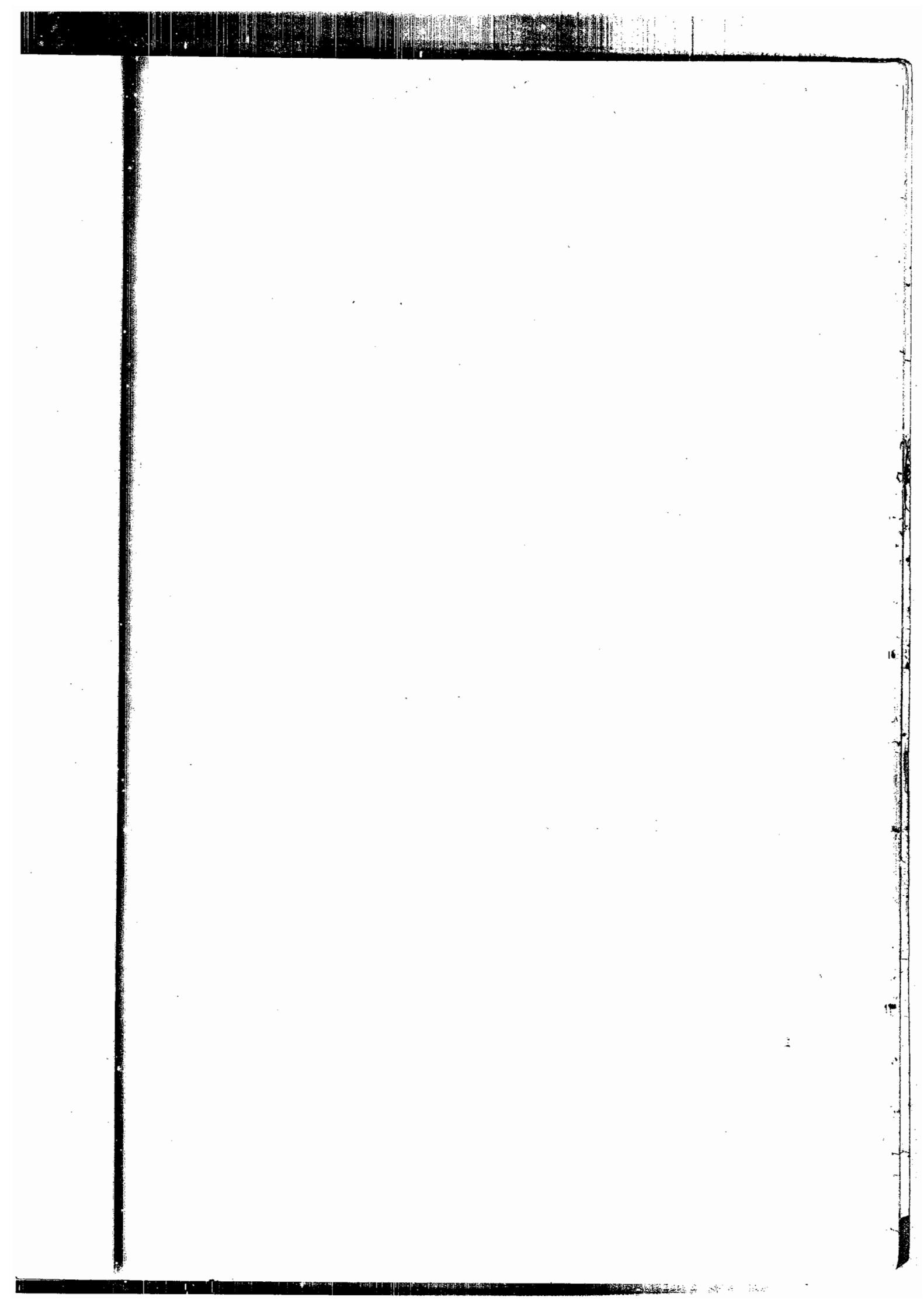


→ for oracle when ever we are performing DML tran
sp's new values for the transaction automatically
stores in dirty ledger & old value for the transaction
stored in undo area within d.b. buffer cache.

→ when ever user gives Commit dirty ledger value
automatically permanently stored in data files and also
undo area data automatically stored in undo file
with in data base.

→ oracle 9i introduced flashback queries to retrieve
old data after committing the transaction also

→ These flashback queries are works based on undotable



Indexes:-

Indexes (use-ind-columns)

- Index is a d.b. object which is used to retrieve 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. sys.
 - 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. Servers Search for indexes.
- Note :- In oracle whenever where clause contains 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 privilege.

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;

⇒ 14.

$$\text{empno Cardinality} = \frac{14}{14} = 1 \text{ (high cardinality) (btree)}$$

② Job cardinality.

→ Select distinct(job) from emp;

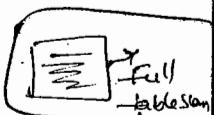
→ 5.

$$\text{Job Cardinality} = \frac{5}{14} = 0.357 \text{ (low cardinality)}$$

bitmap index

btree indexes:- In oracle by default indexes are

btree indexes: (unique, non unique)
default when index is created very index.



→ 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 dictionary.

→ 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 rowids. Based on the

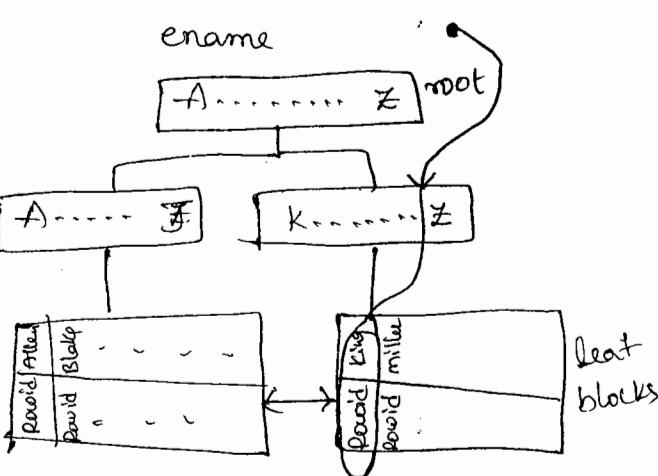
indexes scan mechanism oracle server search data

very fastly in btree structure & also retrieve data very fastly from the leaf blocks.

→ This process automatically improves performance of the application.

Developer

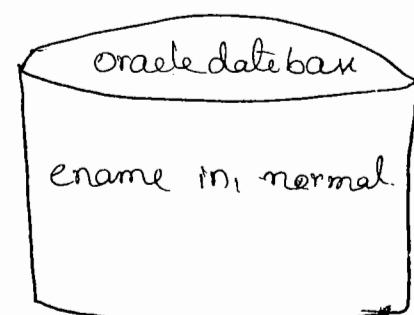
→ Select * from emp (ename created as index to where ename='KING'; btreeformed)



indexes scan mechanism

DBA

(manual creation)
Create index in,
on emp(ename);



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 (-1, CPU)
3.

Using indexes :-

➤ Create Index int 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" 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

** User-IND-Columns :-

→ In oracle if u want to view column names along with index names then we are 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.

to read

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 function-based indexes
which creates indexes on fun's along with columns

→ Syntax:-

create index index_name on
table_name (function_name (column_name));
or
expression

Ex:-

Create index idx on

emp(upper(ename));

→ Select * from emp where upper(ename)='KINTI';
→ Select index-name, index-type from
user_indexes where
table-name = 'emp';

INDEX-NAME

IN1

INDEX-TYPE

Function-Based Normal

~~After~~ 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, function 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 (columnname)) [virtual].
(or)
expression

- > Create table t1 (a number(10), b number(10),
c number(10) generated always as (a+b) virtual);
↳ stored expression (previous col
values)
↳ Synt
- > 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.

Stores

Ex: `> desc user-tab-columns;`

`> select column-name, data-default from`

and stored

`user-tab-columns where table-name = 'T1';`

rows,

<u>COLUMN-NAME</u>	<u>DATA-DEFAULT</u>
C	"A+B"

h Stores

Bitmap Indexes

class

Oracle 7.3 introduced Bitmap Indexes

→ Generally Bitmap indexes are created on "low cardinality" columns

→ Bitmap indexes are created on low cardinality.

and in "duplicating applications"

→ Generally we are not allowed 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 (columnname);`

Create unique index in_1 on $emp(ename)$;

Index created

Create unique index in_2 on $emp(job)$;

error: Can not create unique syntax. Index: duplicate value.
 80

Solutions:-

→ Create bitmap index in_2 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 in_2 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
Andy St	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

One can also drop Index using "drop index" index name

Reference name Synonyms :- View (Security purpose) by default private
dropping synonym table also

- 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 we want to create public Synonyms then we must give "Create public Synonym privilege" to user

Grant Create public Synonym to username;

Private Synonym Syntax:-

- Create Synonym synonym name for username • object name @ database link;

Public Synonym:-

Create public Synonym Synonymname

for username.objectname @ database link;

Scott/Tiger	murali/murali ^o	ar/ar
> Grant all on emp to murali ar; > Select * from Scott.emp ; (v) > Create synonym abc _{synonym} for Scott.emp ; > Select * from abc; (v) > Create public synonym xyz for Scott.emp ; > Conn xyz as sysdba; Enter psw: sys > Grant Create public Synonym to murali; > Conn murali^o / murali^o; > Create public Synonym xyz for Scott.emp ; > Select * from xyz; (v)	> Select * from scott.emp ; (v) > Select * from abc; <u>error</u> : Table or view does not exist > Select * from xyz; (v)	2 1 + (1) too
		(2)

21-9-14

→ 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:-

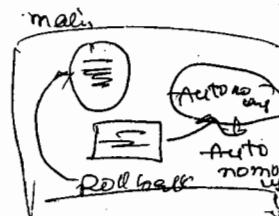
rollback ;

Particular

Rollback to Save Point:-

Syntax:-

rollback to Savepoint name;



Perm

Cre

Step

Step 2

Step 3

Create

This

Differ

But

Synt

Cr

as

Cre

Sy

C.

Step

Sy

[

Commit Completed.

(Create or add type) (undefined type) Not here in db.
Instead Table :- (Index by Table ^{overcome problem} extents is Not) :-

→ Oracle 8.0 introduced instead table basically instead table is a user defined type which is used to store no. of data items in a "single unit".

→ generally if we want to store large amount of data into single unit then we are using Index by Table in PL/SQL.

→ But we are not allowed 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

permanently into d.b. using SQL..

Creating Nested Tables

Step 1:- Create an object type.

Step 2:- Create a nested table type

Step 3:- Create actual table.

Create an 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 datatype (size) ...)

Create an nested table type:-

Create or replace type typename

as table of objectname type;

Step 3:- Create actual table:-

Syntax:- create table tablename (col1 datatype (size),

... coln nested table type) nested table column same

Stores as any name

Ex: Create or replace type e1 as object (book ID number (10), book name Varchar2(10), price number (10));

> Create or replace type xy as table of e1;

/ - complex.

> Create table Student (Stno number(10), sname Varchar2(10),

Col3 xy) instead table Col3 Same Stored as pgm;
Kind address (Copy name its working)

> desc Student;

name	type.
Stno	Number(10)
sname	Varchar2(10)
Col3	xy, <small>2</small> type name

Note: If u want to store data into instead table type then we must use Constructor.

→ here Constructor name is same as type name.

> Insert into student values (1, 'murali') as (e1(101, 'Java', 900));

> Select * from student;

O.P: Col3 (BookID, Bookname, price)

murali

xy (e1(101, 'Java', 900), e1(102, 'PLSQL', 400));

Now created

mes

BOOK Pd

- (10);

value2 (10),

Copy name
its value)

Stno	Sname	Col3
1	murali	

Indirectly Storing
Nested table

Bookid	Bookname	price
101	java	900
102	PLSQL	400

> Select * from table (Select Col3 from student);

Bookid	Bookname	price
101	java	900
102	PLSQL	400

> update table (Select Col3 from student) Set price=1000

where BookId = 101;

ad table

> Select * from table (Select Col3 from student);

101	java	1000
102	PLSQL	400

(101,Java)

→ If u want to view nested table date or modified nested table date then we must use Table operator.

→ & also along with Table operator we must select nested table columns within subquery

Syntax:

- Select * from table (select col3 from student);
- Select * from table (Select nested table colname from actual table name);

Q.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 data warehousing applications.
- generally partitions are created based on a key column
- This col is also called as partition key
- when database contains large amount of data 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.
from

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 (maxvalue)

retrieve particular 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 (maxvalue))

- > 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 introduce list partition in this method data base administrator partitioning the table by using list of values.

Syntax: create table tablename (col ,daty (size),....)

partition by list (key columnname)

(partition partition name1 values (value1, value2, ...),

partition partition name2 values (value1, value2, ...),

partition particular name values (Default));

Ex:- Create table test1 (sno number(10), name varchar(10))
Partition by list (name)
(partition P1 values ('india', 'Pakistan'),
partition P2
values ('us', 'uk', 'canada'),
partition others
values (default));

alter
table
 > insert into test1 values (....);
 > select * from test1 partition (others);

<u>SNO</u>	<u>NAME</u>
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 date dictionary.

Ex: SQL> desc user-tab-partitions;

→ Select partition-name from user-tab-partitions
where table-name = 'Test 3';

Hierarchical queries:- In relational date bases we can also stored hierarchical data

→ In oracle if we want to retrieve hierarchical data then we are using level pseudo column along with following clauses.

(1) Start with

(2) Connect by.

→ If we want to store hierarchical data in relational table minimum 2 column are required and also these 2 column must belong to same datatype & also having some relation.

→ level is a Pseudo Column which assigns no's to each level in Tree structure

→ Start with:-

This clause specify searching condition with in hierarchy

ab-

Syntax:- Start with Condition

Connect by:- This clause is used to specify deletion
row in

ship b/w parent, child Column.

— In this clause we must used prior operator

Syntax's Select Col1, Col2

from tablename

where Condition

Start with Condition

Connect by prior

parent column name = child column name;

➤ Write a hierarchical query to display employee who
are working under another employees root node
onwards from emp table using employee number,
mfor Columns.

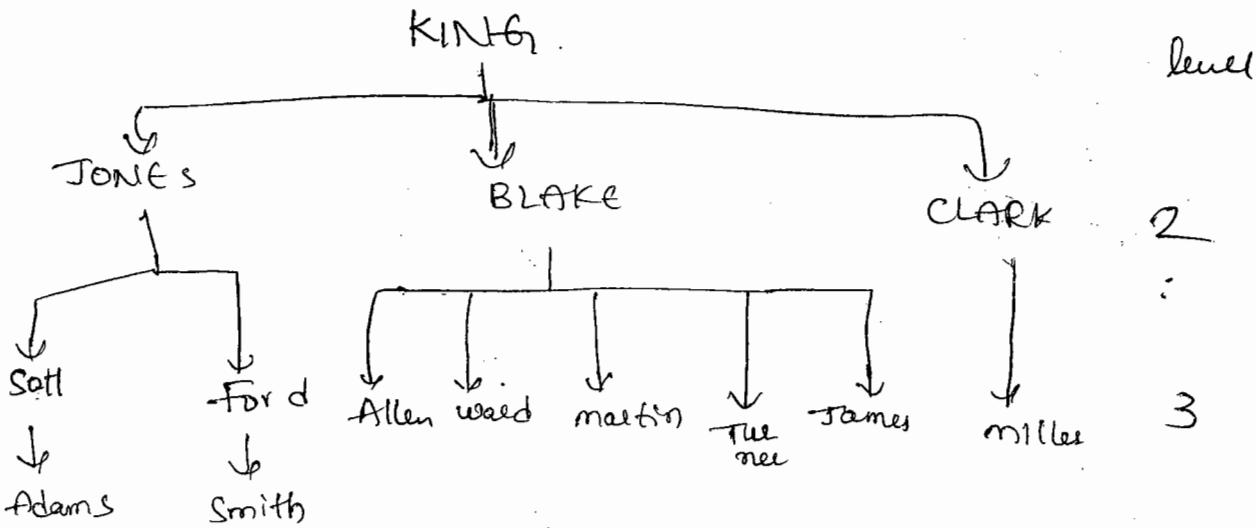
SELECT level, fname

from emp

level

Start with mfor is null

Connect By prior empno=mfor.



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, 'delimitername')

Ex:-

Select

level . `SYS_CONNECT_BY_PATH(ename, '→')`

From emp

Start with mgr is null

Connect by prior `empno = mgr`

1.

level

Q) write a hierarchical query to display the employee who are working under Blake from emp table using emp no, mgr columns.

2

Select level,

3

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

Plus is an unary operator and along with
Connect by clause always prior operator between
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

up to

Interview questions Flashback Queries.

- Oracle 9i introduced flashback queries using flashback queries data base administrator to retrieve accidental date
- flashback queries always allows content of a table can be queried with respect to a specific point of time using As of clause
- Flashback queries works based on undo file using flashback query we can retrieve previous data after committing date also
- To retrieve accidental date then we are using 2 methods
 - (1) using timestamp datatype
 - (2) using SCN (System Changed Number)

Method - 1:- using timestamp datatype

Syntax:

Select * from tablename

as of timestamp(specific point of time);

Ex: > Create table t1 (Sno number (10));

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

> Commit;

> Select * from t1;

O.P.:	SNO
1	1
2	2
3	3
4	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 seq)

→ In oracle whenever we are creating transaction then automatically a unique identification no. is created in the date base.

→ This unique identification number is also called as system changed number and on this number also data base administrator can do any operation.

Syntax :-

Select * from tablename

as of scn System changenumber;

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.

Question

10 → 25

